Torstein Martinsheimen Egge
Åsmund Hunderi Stemland
Omer Jonuzi

Cloud Backup Architectures Resistant to Ransomware Attacks

Bacheloroppgave i Digital infrastruktur og cybersikkerhet
Veileder: Helge Hafting
Medveileder: Gleb Sizov

Mai 2022

Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Torstein Martinsheimen Egge
Åsmund Hunderi Stemland
Omer Jonuzi

Cloud Backup Architectures Resistant to Ransomware Attacks

Bacheloroppgave i Digital infrastruktur og cybersikkerhet
Veileder: Helge Hafting
Medveileder: Gleb Sizov
Mai 2022

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Norwegian University of
Science and Technology

Cloud Backup Architectures Resistant to Ransomware Attacks

Torstein Martinsheimen Egge Åsmund Hunderi Stemland
Omer Jonuzi

May 20, 2022

Abstract

Ransomware attacks have been on the rise globally the last few years. They pose a serious threat to any business's data, reputation and operational capability. Backups are an essential part of an effective strategy against ransomware attacks, functioning as the last line of defense. How do you prevent a hacker from corrupting or outright deleting your backups before unleashing their ransomware?

In this thesis, we attempt to find the trends and characteristics that define modern ransomware, as well as how to recover from modern ransomware attacks. We explore various backup solutions for managed and unmanaged databases in Azure, the cloud service platform operated by Microsoft, and evaluate their effectiveness against modern ransomware attacks. In addition, we look at security mechanisms and features in Azure that can help defend backups from malicious actions.

The results address several security features which block a number of attack vectors. Furthermore, issues with the existing backup solutions are highlighted and in some cases suggestions for improvement proposed.

Sammendrag

Løsepengevirusangrep har sett en økning globalt de siste årene. De representerer en alvorlig trussel mot enhver bedrifts data, omdømme og operasjonelle kapabiliteter. Sikkerhetskopier er en essensiell del av en effektiv strategi for å motså løsepengevirus-angrep, med funksjonen til en siste forsvarslinje. Hvordan skal en hindre en hacker fra å infisere eller slette sikkerhetskopier før de setter i gang med løsepengevirusangrepet?

I denne oppgaven forsøker vi å se på de trender og karakteristiske preg som definerer moderne løsepengevirus, i tillegg til hvordan gjenopprettelse etter et moderne løsepengevirusangrep utspiller seg. Vi utforsker løsninger for sikkerhetskopiering av både administrerte og ikke-administrerte databaser på Azure, Microsoft sin plattform for skytjenester, og sammenligner hvor effektive de løsningene er på å motstå moderne løsepengevirus. I tillegg ser vi på sikkerhetsmekanismer og funksjoner i Azure som bidrar til sikring av backup mot ondsinnede handlinger.

Resultatene tar for seg flere sikkerhetsfunksjoner som setter en stopper for en rekke angrepsvektorer. Videre fremhever vi problemer knyttet til eksisterende backupløsninger og i noen tilfeller foreslår vi forbedringer.

Contents

Figures

Glossary

backup Copy of data that can be used to restore the original in case something happens to it.. 2, 11

malware Malicious software. General term for software that does damage to computers, IT systems, etc.. 7

managed service A cloud service that is operated and maintained by the cloud provider.. 2

privilege escalation Attack where the attacker obtains unauthorized privileges/permissions, often by exploiting a vulnerability.. 11, 17

ransomware Type of malware that encrypts files, in order for the attacker to demand a ransom from their victims in exchange for the files being decrypted.. 1

security control A security control is any action taken to secure something in a system.. 5, 6

unmanaged service A cloud service that is operated and maintained by the customer, and not by the cloud provider.. 2

virtual machine A virtual machine (VM) is a computer emulated in software. One of the building blocks of cloud infrastructures.. 3

zero-day Computer security vulnerability that is unknown or unpatched.. 8

Preface

This thesis was written as the final assignment of our bachelors degree in digital infrastructure and cybersecurity at the Norwegian University of Science and Technology (NTNU). It explores the cross section between infrastructure and security: Security controls against ransomware in databases hosted in a public cloud environment.

We were motivated to write this thesis because we felt it was both relevant and important to organisations facing the growing threat of ransomware. No other theme has been as prevalent in cybersecurity news while studying for our degree. The choice to focus on cloud environments is relevant for today, but will also remain relevant for years to come as well.

We hope this thesis is relevant to organizations that are unsure of how to assess the myriad of security features in cloud environments, and help them stay secure.

We would like to thank TrønderEnergi for their help with the project, NTNU for the opportunity to study relevant and important subjects. We would also like to thank our advisors: Helge Hafting, Gleb Sizov, and Joakim Klemets. All your advice helped us greatly.

Chapter 1

Introduction

1.1 Background

Ransomware is one of the biggest cyber threats businesses and organisations face today [1]. Ransomware has been around since the early days of computers, and was first delivered via floppy disks, targeting AIDS researchers in 1989 [2]. With the advent of asymmetric encryption schemes, the creation and growth of cryptocurrencies, and an ever growing reliance on digital workspaces, ransomware is bigger than ever [3].

As more and more organizations go through a digital transformation, in particular because of the Covid-19 pandemic, more and more work is done digitally, and over the internet. The risk surface is greater than ever, and ever interconnected systems prove to be great targets for ransomware. Nowadays ransomware gangs operate as legitimate software companies, delivering ransomware-as-a-service to less sophisticated cybercriminals who launch attacks as their affiliates – splitting the profits [3].

Ransomware is no longer spread widely and randomly to any system that will run it. Ransomware gangs are now looking to buy access to corporate networks, and partner with affiliates that are willing to perform the attack on that network. Weeks of research go into a single attack to ensure it is as effective as possible, and to ensure that the ransom is set at an amount that maximizes the likelihood of a payout [4].

Many businesses are moving to the cloud, and rely on cloud platforms to maintain security. A big advantage of this is that cloud platform providers are inherently interested in maintaining a secure platform – or their reputation will suffer. At the same time they are always in development and driven by a desire to make as large a profit as possible by delivering as much value as possible, for the smallest cost.

This is the background for our thesis topic. In a digital landscape where ransomware is one of the biggest threats, and data is an organisations greatest asset, can organisations rely on the security solutions in the cloud? Despite the fancy words and fantastical features of security solutions – do they hold up?

1.2 Thesis Topic

Our thesis topic is to analyze cloud backup architectures with regards to their resistance to ransomware attacks, and recommend a set of features to aid implementation in a secure manner.

We have chosen to focus on the cloud platform, the second largest cloud platform by revenue [5]. Within Azure we will look at two different databases, one that is managed service and one that is unmanaged service, and analyze different backup options for both. To aid us in this search, we have developed the following research questions that we will answer based on our analysis.

1.2.1 Research questions

Research question 1

What are some best practices for securing backups against ransomware and other malware, and how can they be implemented in Azure?

Research question 2

Are Azure's security mechanisms effective against a modern ransomware attack?

1.2.2 Partner organisation

This project was completed in cooperation with the Norwegian energy company TrønderEnergi, who suggested the topic, and who have been providing guidance throughout the project. Because of this cooperation, certain deliminations and choices regarding the technologies used have been made.

We tested two databases in our analysis. The first is a ClickHouse database running in a VM in Azure. The other is a single server PostgreSQL database hosted on Azure. These were both chosen because TrønderEnergi uses them in their production environment, and because they were different enough to provide generalized results for managing backups in Azure. As becomes apparent throughout the report, they are supported by a very different set of security features within Azure.

To analyze the backup solutions, a number of scenarios were created. The scenarios were made in cooperation with TrønderEnergi, which was done in order to ensure our tests matched the threat landscape the an organisation such as theirs is facing.

1.3 Thesis outline

Chapter 1: Introduction In the introductory chapter, we provide the background for the project, as well as the topic and scope of our report.

Chapter 2: Theory In the theory chapter, we examine the current ransomware threat landscape and how ransomware works in order to be able to simulate it in our analysis. We also use this chapter to present any external sources that provide the theoretical background for our later deliminations in terms of our method and analysis. This includes sources that define backups, the relevant resources and services in Azure, as well information about ClickHouse and PostgreSQL. Finally we also look at some relevant security best practises that will become relevant when discussing the research questions toward the end of the report.

Chapter 3: Method In this chapter, we define the criteria which our security implementation will have to pass in order to prove effective. This includes not only criteria in terms of how secure it is, but also non-functional requirements such as performance, cost and ease of use. The chapter also outlines how experiments were performed, as well as how our test environments were set up.

Chapter 4: Results In this chapter, the results of our experiments are presented. These results are discussed briefly for each experiment.

Chapter 5: Discussion In this chapter, we answer the research questions from 1.2.1, based on our discussion of the experiments and their results in chapter 4. We evaluate the backup solutions we analyzed, and look at the results in a greater context.

Chapter 6: Conclusions In the conclusion, we summarize our findings. We will also look at future work on the topic, as well as this project's place in the greater picture.

1.4 Scope and delimitation

The goal of this project was to assess backup solutions in Azure, and how effectively they mitigate the risk of data loss when attacked by ransomware. The backup solutions back up two different databases hosted on the public cloud platform Azure. The databases were Azure Database for PostgreSQL, a managed database server in Azure, and ClickHouse, an unmanaged database running in an Azure virtual machine.

While the project delves into the current ransomware threat landscape, the analysis did not use real ransomware, as it seemed an unnecessary complication of our analysis without providing any significant insights. The goal was rather to determine how well we could protect our backups with the security features in Azure, not how ransomware operates in a system or network.

The methodology was as follows: First we determined our scope, and began researching our chosen technologies, ransomware and security more in general. Once we had a good idea of the the threat landscape, we decided to define some scenarios which outline potential attacks that a good backup solution should be able to withstand. Finally we created test infrastructures and performed experiments based on the scenarios. The results were used to evaluate the different backup solutions with regards to certain criteria.

Outside the scope of this project are most of the other ways of increasing security in an organisation that is also necessary to prevent ransomware attacks from occurring. Be it cultural, technical or organizational aspects of security, they are beyond our scope. We will only focus on the security features of backup solutions in Azure, and considerations on how to implement it.

Chapter 2

Theory

This chapter of the report is where we lay the theoretical groundwork for the discussions and choices in the rest of the report. The choice of technologies in chapter 3, is based on the information in this chapter. The chapter starts with some definitions and concepts and a concise summary of the ransomware threat in sections 2.2 and 2.3. The technologies we utilized in the project are described in sections 2.4 to 2.7. The last section (2.8) presents some best practices for security controls.

2.1 Definitions and concepts

2.1.1 Public cloud

Any search for a definition of the cloud will yield a tremendous number of results, and just about as many different interpretations of what the cloud actually is. In a 2010 article, Hofmann and Wood [6] provide the following definition: "At its core, cloud computing means providing computing services via the Internet. The 'cloud' idea is tightly connected with the 'as-a-service' idea." They go on to describe the public cloud as a set of standard resources that can be combined into applications or services. While this article is over ten years old, the fundamentals are still the same.

The public cloud is available to organisations and consumers through cloud platforms such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP), which are the biggest providers on the market [5]. Cloud platforms generally provide resources to the user. Resources are manageable items that are available through that cloud platform [7]. A common example is virtual machines, that are compute resources made available to costumers as a server over the internet, but it is in fact a virtual server on top of a physical server.

2.1.2 CIA-triad

An important aspect of Security design is the concept of the CIA-triad. CIA is short for: Confidentiality, Integrity, and Availability [8]. Any security control must address either the confidentiality of the thing you are trying to secure, its integrity or its availability. A security control is any action taken to secure something in an system.

- Confidentiality
  This involves keeping data secret, and preventing unauthorized disclosure of data.
- Integrity
  This ensures that data is not modified by unauthorized persons, or unauthorized modifications are not made to data no matter by whom. It also ensures data consistency.
- Availability
  This aspect is about making sure that data and resources are available to authorized personnel in a timely manner.

2.1.3 Ransomware

Ransomware is a central aspect of this project and has a dedicated chapter (2.2) Ransomware is a type of malware that encrypts files on an infected system and holds the files as ransom for typically large sums of money – typically in the form of cryptocurrency. The malicious actor that spread the ransomware in question will usually have the decryption key and only share it with the victim once the ransom has been paid [1].

2.1.4 Databases

A database is a type of system that stores and organizes data. Databases are used for many different purposes including storing application data, analytics, or any information that can be stored as data. Structured Query Language (SQL) is a commonly used language for interacting with databases. Databases that do not use SQL exist as well. These are commonly referred to as "NoSQL" databases.

2.1.5 Cryptography concepts

One of the essential building blocks of any ransomware is the concept of cryptography. Sjaak Laan defines cryptography in his book as "the practice of hiding information using encryption and decryption techniques" [8]. Only those who know how to decrypt the information can read it. This is usually done using a cipher, which are a pair of algorithms, that along with a key – the secret only known by the encrypter and those they share it with – controls the encryption or decryption process.

A number of different methods of encryption exists, some more advanced than others. For security purposes, it is generally advised to use a known cipher, with a secret key. As the cipher will be subject to scrutiny by security professionals at large, the odds of gaining any security by obfuscation in this space is limited. Given that the key can remain secret however, a tried and tested, open-source cipher method with a secret key is by far the most secure.

2.1.6 RPO and RTO

The efficiency of a data protection or disaster recovery plan can be measured by two specifications, namely; Recovery Point Objective (RPO) and Recovery Time Objective (RTO) [9].

RPO The Recovery point objective (RPO) is the data loss after disaster recovery, measured by time, upon where data loss exceeds the threshold of what an organization can accept.

RTO Recovery Time Objective (RTO) is a metric that defines both the time duration within which a service must be restored to after a disaster in order to avoid detrimental damage caused to the business.

2.2 Ransomware

2.2.1 Definition

Ransomware is a type of malware that encrypts files on an infected system and holds the files as ransom for large sums of money – typically in the form of cryptocurrency. Access to data is usually blocked by encrypting the data (crypto ranomsware) or by locking the computer (locker ransomware) [1].

Recovering encrypted files is generally exceedingly difficult without access to the encryption key that was used. Paying the ransom is also not a reliable recovery option, as there is no guarantee the attacker will keep their end of the bargain. In addition, by paying the ransom the victim is likely to be marked as a target in future attacks, as well as funding the ransomware threat actor. According to reserach from Cybereason 80% of victim organisations that paid the ransom were attacked again [10].

2.2.2 Ransomware trends

Ransomware has been steadily on the rise and has become highly relevant for the cybersecurity industry during the late 2010s. Malware actors have become more organized, and more goal-oriented, which according to cybersecurity research group Cisco Talos means that they now primarily try to follow the money [11]. Cisco Talos pose that threat actors do not want to risk getting discovered by installing low-profit malware, when they instead can sell the access, or use ransomware to make a larger profit.

This explains why ransomware has grown in popularity by threat actors in recent years. According to a report by the security firm Check Point, the first half of 2021 saw twice as many known ransomware attacks compared to 2020 [12]. The report also highlights some trends that further Cisco Talos' point: Threat actors keep searching for ways to employ double- or triple-extortion techniques to maximize profits. Double- and triple-extortion involves not only extorting the victim organisation for ransom, but also threatening the organisations customers or clients and threaten to release their data unless they also pay up.

The energy and utilities sector was also the second-most targeted, which is part of the reason for this project focusing on the needs of the Norwegian energy sector [12]. Prominent examples of this are the ransomware attacks on Volue, a Norwegian company that provides technology to the energy and infrastructure sector, as well as the prolific Colonial Pipeline attack. In the latter the pipeline that supply the southeastern United States with gasoline had to stop all operation because of a ransomware attack on its computerized equipment. Both attacks occured in May 2021 [13]. The latter has been named "one of the most disruptive ransomware attacks ever on US critical infrastructure" [14].

The goal for any business has to be the prevention of ransomware attacks, but that may not always be possible, as with the attack on the Norwegian parliament in March 2021 [15], where a zero-day vulnerability in Microsoft Exchange proved fatal enough for data to be exfiltrated. Similar examples are prevalent, and it is impossible for an organisation to guarantee prevention.

When prevention is not possible, the mitigation of the consequences of ransomware attacks become vital. One of the most important risk mitigation measures that an organisation can do is to set up backup solutions for their data. For critical infrastructure this backup has to both be frequent, and consistent enough to let the company return to normal operations quickly, even when they are the victim of a ransomware attack.

Human-operated ransomware

In a human-operated ransomware attack, the attacker targets a specific organization, manually breaks in and deploys ransomware. This is done in order to increase their chance of success and a potential payout. Before deploying the ransomware, they might make preparations in order to decrease the organization's ability to defend themselves, such as destroying backups. This is unlike more traditional, undirected ransomware attack, where the ransomware spreads to random computers by replicating itself across a network.

Being struck by a human-operated ransomware attack can be very expensive, even if the ransom is not paid, as rooting out the infection can be difficult [16]. The attacker might leave backdoors, entry points in the infrastructure for the attacker to reuse later, or compromise the organization's security in other ways.

Ransomware as a service, access as a service

Ransomware-as-a-service (RaaS) is a business model for ransomware developers, in which the developers license their ransomware to other threat actors. According to an article published by Threatpost, ransomware gangs look more and more like legitimate businesses, with IT-support, and customer service. These ransomware gangs offer affiliate programs to whom they provide RaaS-offerings [4].

RaaS is a driving factor in today's ransomware landscape, and most ransomware gangs operate on the RaaS model, according to research from security firm Abnormal [3]. This shows an overarching trend in the way that ransomware gangs pick their targets. Gone are the days of spreading malware to as many systems and networks as possible, hoping something will stick. Nowadays, an attack is preceded by weeks or months of reconnaissance. Not only do the attackers explore the systems they've penetrated, but they also explore the businesses financial situation and internal communications in order to set a ransom that has the highest chance of resulting in a payout, whilst also being as large a sum as possible [4].

The model for choosing affiliates is also highly selective, and hopefuls will have to go through a rigorous application process, submitting a resume and impressing the ransomware gang in an interview. The ransomware gangs are on their hand worried about poor-performing affiliates which may compromise the brand's reputation, or infiltrators from western law enforcement agencies. As a result many ransomware gangs are quite picky both in terms of technical abilities, as well as language and knowledge of Russian history and culture that is not easily researched [4].

2.2.3 Ransomware gangs

According to research made by Abnormal [3], a few ransomware gangs have dominated the ransomware scene in the last two years. These are Conti, LockBit, Pysa, REvil, and Maze/Egregor. They were responsible for more than half of all ransomware attacks in this time period. Conti, LockBit and Pysa are still active as of the publication of their research, and all have a large number of victims.

The aforementioned prolific attacks on Volue and the Colonial Pipeline (see 2.2.2) were conducted by Darkside and Ryuk respectively [14]. Both are known for attacking the energy sector.

2.2.4 How ransomware works

The ransomware field is ever evolving, and any analysis is only snapshot of ransomware at the time of publishing. Despite this it is useful to assess the present situation to understand the threat at hand.

An analysis of encryption schemes in modern ransomware was published in 2020 by Plozek, Svec, and Debnar [17]. In the paper 10 different ransomware samples were analysed, but only one of those, LockBit, overlaps with the dominating groups as assessed by Abnormal. The samples were collected between 2019 and 2020, so this might show the rate of evolution in this field.

In the analysis the researchers identified four different encryption schemes in the ransomware malware samples. The differences mainly concerned where the keys were generated, and how they are distributed. Prevalent in their findings was that different keys are usually generated for each victim, indicating that the goal no longer is to affect as many victims as possible, but to make as much profit per victim that they can [17].

Another major finding in their research was that malware authors are getting better at cryptography. Generally compared to older research the researchers found that modern ransomware used more secure encryption algorithms than older ransomware. This trend toward more advanced encryption schemes align with the general trend of ransomware gangs getting more professional [17].

This makes brute force decryption less feasible or in many cases not feasible at all, and any files encrypted by ransomware should be considered lost, as decryption of those files is not likely to have effect, especially in the time necessary for an organisation to not suffer great losses. In addition the paper discovered that several of the ransomware strains created a separate key for each encrypted file, so the brute-force method would be even less feasible [17].

2.3 The anatomy of an attack

Cyberattacks generally follow a similar pattern, with the same main steps or phases. The number of steps, and the names they are assigned differ, but they are generally similar. A brief overview of the anatomy of an attack is described below. Figure 2.1 illustrates these steps visually, as a flowchart, showing possible paths an attacker may take.

Reconnaissance

An attacker can spend a long time before an attack gathering information, in order to evaluate which organization they should target to earn the most money, as well as looking for vulnerabilities that can be used to gain a foothold in the organization's network.

Initial access

After gathering information about the organization, the hacker will try to gain access to the network. According to BankInfoSecurity [18], the following three attacks are the most common types of attacks used for initial access:

- RDP compromise
  By exploiting weak Remote Desktop Protocol (RDP) endpoints, a hacker can get full access to a user account on the network. RDP compromise is currently the most common vulnerability used for initial access [18, 19]

- Phishing
  Phishing is a type of social engineering attack where the attacker sends a link or email attachment containing malware, pretending that it is legitimate. Through the information gathered in the reconnaissance step, the attacker may be able to craft believable emails.
- Software vulnerabilities
  Vulnerabilities or bugs in software can expose the organization in a way that an attacker can take advantage of.

Expansion

After getting a foothold in the organization's network, the attacker will move through the network (lateral movement) to gain information about resources and assets, and attempt to expand their privileges (privilege escalation).

Exploitation

After gaining access to enough privileges and resources, the main attack can begin. Before deploying the ransomware, the attacker might first attempt to exfiltrate (steal) valuable data. The attacker might also try to destroy backups before proceeding to deploy the ransomware itself, encrypting the data.

After this, the attacker typically demands a ransom in order to decrypt the data. If the data has been exfiltrated, the attacker might try to perform a double extortion attack, in which they make a threat to release the exfiltrated data publicly if the organization does not pay the ransom.

2.3.1 Attacks against backups

Since backups are an essential part of being able to recover from a ransomware attack, hackers will often try to destroy their victim's backups before encrypting files. Backups are explored in detail in section 2.4 In this section, we will outline a few possible attack vectors against the backups themselves that could try to make them useless.

The simplest way of destroying backups is to simply delete them. By deleting an organization's backup, hackers can deny recovery, which can increase the chance that the organization pays the ransom.

A potential method of corrupting an organization's backups could be to slowly encrypt the data that is being backed up. Over time, the backups would be overwritten with invalid data, making them useless by the time the ransomware is deployed.

A hacker could potentially exfiltrate data from backups. By encrypting data before backing it up, this risk can be mitigated. If a hacker exfiltrates encrypted data, they will not be able to read it without having access to the encryption key used.

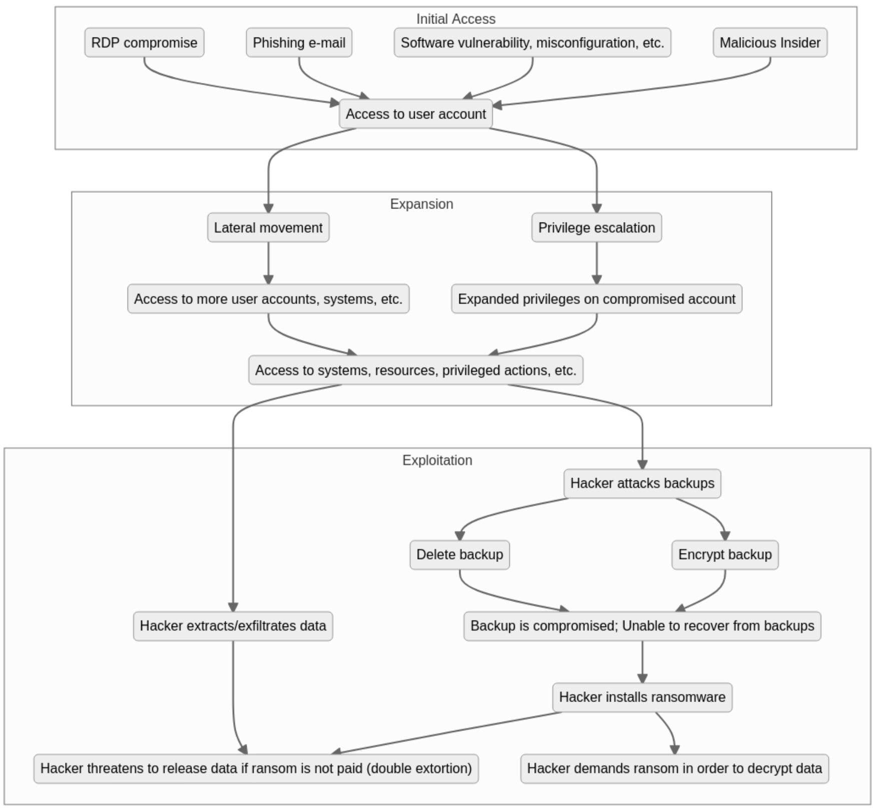

Initial Access
RDP compromise
Phishing e-mail
Software vulnerability, misconfiguration, etc.
Malicious Insider
Access to user account
Expansion
Lateral movement
Privilege escalation
Access to more user accounts, systems, etc.
Expanded privileges on compromised account
Access to systems, resources, privileged actions, etc.
Exploitation
Hacker attacks backups
Delete backup
Encrypt backup
Hacker extracts/exfiltrates data
Backup is compromised; Unable to recover from backups
Hacker installs ransomware
Hacker threatens to release data if ransom is not paid (double extortion)
Hacker demands ransom in order to decrypt data

Figure 2.1: Illustration of the anatomy of a cyber attack

2.4 Backup

Faced with ransomware attacks and malicious actors that target the data of organisations, securing that data is essential. It is a fact of living in a digital, interconnected world that malicious actors will attempt to target any data that is available. Face with that risk keeping backups is an important security control.

According to Sjaak Laan, "Managing security is all about managing risks" [8]. Faced with any type of risk, senior management has the option between risk acceptance, avoidance, transfer, and mitigation. While having no data stored digitally, or completely disconnecting all systems from the outside world would be considered avoidance, and some sort of insurance would be considered risk transfer (although almost half of victim organisations that has cyber insurance only got a portion of the losses covered [10].) An effective backup solution would be an effective risk mitigation technique.

Sjaak Laan defines a backup as copies of data, used to restore data to a previous state in case of data loss, data corruption, or a disaster recovery situation. Laan argues that a backup older than a few weeks is of little use, as old, outdated data is probably not very useful for a company in an emergency. The archiving of older data for compliance should be done separately from backups (but should be backed up also) [8].

Recovery is the process of restoring files from a backup, and is an essential part of any plan to mitigate data loss risk. A backup without a good plan for recovery loses a lot of its use. Recovery should also include procedures for spinning up new VMs as in case of a ransomware attack it would be fair to assume the network to be infected.

In his book "Infrastructure as Code," Kief Morris [20] argues that in the cloud age organisations should plan for disaster recovery continuously, for example by using the same tools to recover from a disastrous event that the organisation uses to provision and change infrastructure. Recovering from complete disaster should according to him be as easy as anything else. Restoring from a backup in the cloud, and accessing the necessary data on fresh, secure VMs is principal in cloud infrastructure design.

2.4.1 Types of backup

Depending on the type of data, and the organization's needs, different types of backups are used. The three main types of backups are are full, differential and incremental backups [21] [22]. A brief description of each type of backup follows below.

Full backup

A full backup is one of the most straightforward types of backup. In a full backup, all the data is backed up at once. Depending on the amount of data, this can take a long time, and may require a lot of storage. Recovering from a full backup is usually very simple. All one needs to do is load the disk containing the backup, and start the recovery process.

Full backups only require one backup medium (unless the amount of data is very large, in which case it needs to be distributed).

Differential backup

Differential backups are built upon an existing full backup. They work by only backing up the changes made since the last full backup, instead of backing up all the data. Since the amount of data changed is generally much lower than the total amount of data, this is usually faster than doing a full backup every time.

Differential backups require at least two disks, one for the full backup and one for the changes. Recovery can be done by loading the full backup first, and then the changes. Recovery from a differential backup is generally slower than from a full backup, but faster than from an incremental backup.

Incremental backup

Incremental backups are similar to differential backups, in that only changes since the last full backup are backed up. However, unlike differential backups, where all changes since the last full backup are stored, incremental backups store only changes since the previous backup activity (full or incremental).

Recovery from incremental backups consist of first loading the full backup, then all the incremental backups in order. This can be a time consuming and complicated process, especially if done using physical drives. Some cloud platforms, like Azure, automate this process, making it both fast and easy [23].

2.4.2 Database backup

There are several ways to back up databases. Many databases support a way to to generate a series of SQL statements that recreate the database. This is called a database dump. To recover from a dump, the SQL statements are simply run by the database engine.

Another way to back up a database is to make a backup of the entire virtual machine the database is running on. This consumes more storage than a dump, but can potentially be faster to recover from, since software and configuration files are backed up alongside the data.

2.5 The Azure cloud platform

Microsoft Azure is one of the largest cloud platforms on the market [5]. The platform provides a number of services to organisations that wish to move to the cloud.

2.5.1 Azure Backup

Azure Backup is the go-to backup service in Azure. It aims to "provide simple, secure, and cost-effective solutions" for backing up, and recovering data from the Microsoft Azure cloud [24]. The service supports a number of services and storage solutions, and the list appears to be expanding. One month into this project for example, it was announced that Azure Backup now supports the managed Azure Database for PostgreSQL-service that is in the scope of our project, with the release of Azure PostgreSQL backup with long-term retention in February 2022 [25].

Azure Backup offers a number of benefits, according to the Azure documentation [24]. The data stored in Azure backup is not directly available to the attacker [26]. Protecting backup vaults with Role Based Access Controls (RBAC) is also an important part of protecting the data from ransomware [27]. RBAC will be discussed in more detail shortly, in section 2.5.2.

Azure Backup has support for backing up a number of different resources. There is support for backing up files, folders, and system state of on-premise systems by using Microsoft Azure Recovery Services (MARS) agent [28].

Backup items are stored in vaults, either a Recovery Services vault, or a Backup vault. Vaults are online storage entities in Azure. According to Microsoft they are designed to make it easier to manage the backed up data. Vaults offer monitoring of the backup data, and configuration of things such as redundancy. Each vault holds data, "such as backup copies, recovery points, and backup policies." [29]

Recovery Services Vault

A Recovery Services vault (RSV) is a type of management entity for backups in Azure. It is a newer version of Backup vaults with a number of additional features. Since its release in 2016, it has been supported with security features like soft delete, Cross Region Restore, and Multi-user authorization [30]. The Multi-user authorization and soft delete features are discussed in sections 2.5.3 and 2.5.4 respectively.

Vaults also support a variety of data redundancy tiers (2.5.1), which provide different ways to replicate data across data centers.

The services supported by Recovery Services vaults, according to Microsoft [30], are:

- Azure Virtual machines
- SQL in Azure VMs
- Azure Files (Azure Storage)
- SAP HANA databases in Azure VMs
- Azure Backup server
- Azure Backup Agent
- DPM

Backup items are held locally, in a storage account, for a while before being transferred to the vault. Transferring items to a vault can take several hours, but it happens automatically.

Backup Vault

Backup vaults are the older version of recovery services vaults, and therefore offer less functionality. There is no support for soft delete, Multi-user authorization, or other newer security features. In 2017 Microsoft announced in a blog post that they would support "seamless upgrade of classic Backup or Site Recovery vaults to ARM based Recovery Services vaults. " [31]

Backup vaults are still supported in Azure backup to this day, but its primary use is to be "a storage entity in Azure that houses backup data for certain newer workloads that Azure Backup supports" [32]. This includes Azure Database for PostgreSQL servers, and other newer workloads:

- Azure Database for PostgreSQL servers
- Azure Blobs (Azure Storage)
- Azure Disks

- Kubernetes Service (In preview)
- AVS Virtual machines (In preview)

Azure snapshots

A snapshot typically refers to a read-only point-in-time copy of something, for example a virtual machine or a hard drive. Unlike file based backups which store only data files, snapshots store state the state of the object.

Azure snapshots are read-only copies of a managed virtual hard disk (VHD). These snapshots can be used for backup and recovery purposes. The snapshots can then be recovered to a new VHD. If the snapshotted disk was an operating system disk, it's possible to create a new VM directly from the snapshot.

Azure provides two types of snapshots, full and incremental. These are essentially implementations of full and incremental backups (see 2.4.1). A full snapshot makes a copy of the entire disk, while an incremental snapshot only makes a copy of the changes since the last snapshot. Traditionally, incremental snapshots provide faster backup speeds, at the cost of a slower and more complex recovery process. This is not the case with Azure's snapshots. According to Microsoft blog, recovery is almost as fast from incremental snapshots as with full snapshots, and recovery time is constant no matter how many snapshots are used [23]. Incremental snapshots in Azure can be deleted without invalidating newer snapshots.

Azure data redundancy tiers

Azure provides several tiers of redundancy for data stored in certain Azure services. These tiers provide different ways of replicating data, the goal being to prevent data loss in case an incident occurs. The tiers are described below in order of increasing price [33].

Locally redundant storage (LRS) stores 3 copies of the data, in the same data center. Zone-redundant storage (ZRS) stores 3 copies of the data, in different data centers in the same region (one copy per data center). Geo-redundant storage (GRS) replicates LRS from the primary region to a different secondary region (2×3 copies). Geo-zone-redundant storage (GZRS) combines ZRS and LRS. Data in the primary region is stored in ZRS, while data in the secondary region is stored in LRS.

2.5.2 Role based access control (RBAC)

In Azure, access to resources is only granted when an identity can be authenticated, and it has been assigned the correct permissions. Identites are essentially what seperates different users and resources from each other. Each user has a different identity that is used to grant them the access they need. According to Modi, in Azure for Architects, this is known as authorization, or more commonly as Role-Based Access Control, or RBAC [34]. "RBAC in Azure refers to the assigning of permissions to identities within a scope. The scope could be a subscription, a resource group, or individual resources."

RBAC helps control which users have access to which resources, with as much granularity as is needed to segregate permissions within an organisation, or within a single team. In general it is considered best-practice to only provide the least level of access necessary to perform a task. Modi argues that separating access between team-members helps ensure both security, as well as keeping team-members feeling responsible for their jobs as they may be the only ones with a necessary level of access to perform some tasks.

RBAC has been an important concept of security administrations since its formalization in 1992, by by David Ferraiolo and Rick Kuhn [35]. It is the predominant model of advanced access control according to NIST. One of the advantages of RBAC is that security is managed at a level to closely corresponds to an organization's structure. Many important security features, such as multi-user authentication, and the zero-trust model builds upon RBAC.

2.5.3 Multi-User Authorization (MUA)

While RBAC is useful for administrating access to Azure resources, it may be bypassed if an attacker is able to perform a privilege escalation attack. Destructive actions will still trigger alerts for system administrators, but by the time these are initiated, it may be too late.

Multi-user authorization is a relatively new security feature in Azure which can mitigate this risk. MUA allows the use of a Resource Guard to protect Recovery Services Vaults in Azure (2.5.1). This Resource Guard is owned by a different user (the security administrator) in Azure. The owner of the resource needs to request permission from the security administrator, in order to to perform destructive actions on the resource [36]. As a result, no person is solely in control of any task that may be destructive to the backup data. This can greatly reduce the risk of data loss in case one of the administrator accounts is compromised.

The actions supported by MUA as of this report is:

- Disable soft delete
- Disable MUA protection
- Modify backup policy
- Modify protection
- Stop protection
- Change MARS security PIN

2.5.4 Soft delete

Soft delete is a security feature for Azure Backup or Azure Storage Blobs. When soft delete is enabled, data that is deleted is not removed immediately. Instead it is retained for 14 days before being permanently deleted. During this time, it is possible to "undelete" the data. Soft delete is enabled for Recovery Services Vaults by default[37].

2.5.5 Azure Monitor

Azure Monitor is a service that monitors resources and services in Azure. If certain actions or events are detected, Azure Monitor can issue alerts to the relevant administrators. An example is the Delete Backup Data alert, which is sent out if any backup data is deleted. Alerts are either sent to the Azure Monitor view or via email, if it is serious enough. According to the documentation, certain alerts relating to Azure Backup are sent via email automatically [38]. Custom alert rules can also be configured.

2.5.6 Interacting with Azure

Azure provides three main methods of interaction:

- Azure Portal
  The Azure Portal is a graphical interface for managing an Azure environment. The Portal is generally quite user friendly.
- Azure CLI
  The Azure CLI is a set of commands which can be used to manage an Azure Environment. The CLI can be accessed through the Azure Cloud Shell, which is a type of console in the Azure Portal. The CLI makes it possible to make scripts to automate processes.
- Infrastructure as Code (IaC)
  Infrastructure as Code is an approach to infrastructure management where resources are declared and configured programmatically. This can improve stability and reduce maintenance. IaC can be implemented using Azure Resource Manager templates, or by using third-party tools like Terraform [39].

2.5.7 Azure Blob Storage

Azure Blob Storage is a storage solution in Azure. It is optimized for large amounts of unstructured data [40]. This makes it usable for a variety of workloads, including as a low cost storage solution for backups. Like many other storage options in Azure, it supports data redundancy tiers [33] (see 2.5.1), and security features like soft delete [41] (see 2.5.4).

2.6 ClickHouse

2.6.1 Introduction to ClickHouse

ClickHouse is a database used for Online Analytical Processing (OLAP). It is highly optimized for certain types of workloads, and is able to process large amounts of data in an efficient manner [42].

ClickHouse runs as a server on a virtual machine, which can be accessed using the program `clickhouse-client`. ClickHouse uses SQL as its query language.

2.6.2 Backup solutions for ClickHouse

According to the ClickHouse Documentation, there is no universal backup solution for ClickHouse, but the documentation contains a few different suggested solutions [43]. In this section, we will provide a short summary of the solutions listed in the documentation.

- Duplicating Source Data Somewhere Else
  A program such as Apache Kafka can be used to serve a stream of data as input to ClickHouse. This stream can be forked and diverted to send data to backup storage at the same time as it is sent to ClickHouse.
- Filesystem Snapshots
  Some filesystems like ZFS support making snapshots of the filesystem's state. These snapshots can then be sent to an external storage solution.
- clickhouse-copier
  clickhouse-copier is a tool that can be used to copy data between ClickHouse clusters. Originally intended for petabyte-sized databases.
- Manipulations with Parts
  A local copy of a table can be made with ClickHouse's built-in `ALTER TABLE FREEZE` query. The copy can be exported with a program like `rsync`.
- clickhouse-backup
  clickhouse-backup is a tool that automates the manipulations with parts approach. The tool is installed on the same server as ClickHouse. It can store backups locally or on a remote server, using various storage methods, like Azure Blob Storage or AWS S3 object storage.

2.6.3 Backup solutions for ClickHouse in Azure

Azure provides tools for backing up data. These are not mentioned in the ClickHouse documentation, since they are not specific to ClickHouse. Azure Backup supports backing up individual virtual machines as described in 2.5.1. This feature can be used to back up the VM running ClickHouse.

2.7 PostgreSQL

2.7.1 Introduction to PostgreSQL

PostgreSQL is an open-source relational database management system (RDBMS). It is suited for operations requiring complex SQL queries, and provides functionality enabling data analysis. Azure has its own hosted solution called Azure Database for PostgreSQL, which automates maintenance, patching, and updates [44]. It also allows effortlessly scaling the database size and computing.

There are only a few backup options that are relevant to look into when working with a managed PostgreSQL instance hosted in Azure. This short introduction will also include frequently seen practices for unmanaged instances of PostgreSQL before narrowing the focus into managed ones hosted on Azure.

2.7.2 Backup solutions for PostgreSQL

One recovery option for protecting backup could be removable storage. This will however not be the focus when exploring options for a ransomware-resistant backup architecture.

Some options for backing up a PostgreSQL database include extraction of data using the pg_dump command. This way one could back up data off-site for increased security by air-gapping, although setting this up in such a way that RTO and RPO are met can prove challenging. Moreover, a backup restore from a remote server takes longer time than a local copy.

2.7.3 Backup solutions for PostgreSQL in Azure

Azure Backup

Azure Database for PostgreSQL backup has long-term retention in Backup Vaults. A backup vault is a storage entity housing backup data in Azure [45]. It requires connecting to a Key Vault to integrate with the database instance that will be backed up. Azure Key Vault stores and allows accessing secrets [46]. Examples of secrets could be API keys, passwords, certificates, or cryptographic keys. In this case, it is the database connection string.

The Backup Vault has long-term retention of up to 10 years. It also allows configuration and granular control allowing for a backup policy that complies with the organisations needs.

Point-in-time-Restore

Point-in-time Restore (PITR) is a solution that automatically backs up PostgreSQL data in either local- or georedundant storage.

In contrast to the Backup Vault, the built-in PITR-solution has a default retention of only 7 days, which can be set to a maximum of 35 days if using Standard or Premium tier. Different tiers vary in cost and capabilities. Full backups are taken every week, differential backups every day, and log backups every 5 minutes. These do not need to be scheduled, and no maintenance downtime is associated as they run in the background. PITR is a security measure that one need not opt-in to and does not cost anything unless it gets used. It always restores to an new database which is configured in the same tier as the original database.

The built-in backup solution and the Azure Database for PostgreSQL backup can operate simultaneously.

2.8 Security Best Practices

2.8.1 Principle of least privilege

When assigning permissions to users and processes to access resources in an organisation, it is considered best practice according to the principle of least privilege to only give the minimum level of access needed at any given time. For example doing menial tasks like checking emails and browsing the internet should not be done while accessing a system as administrator. In modern systems architecture the role of system administrator should not be used instead, access should be provided only to the resources a user needs at at given time [47]. In Azure, this is implemented with RBAC (2.5.2).

2.8.2 Zero trust

One of the modern mainstays of cybersecurity and infrastructure security is the Zero Trust model. The National institute of standards (NIST) special publication 800-207 provides the following definition: "Zero trust provides a collection of concepts and ideas designed to minimize uncertainty in enforcing accurate, least privilege per-request access decisions in information systems and services in the face of a network viewed as compromised." [48]

This definition says that one should view the network as compromised, and as such bear that in mind for every decision made when designing the network. The alternative, assuming a network or system is not compromised, has disastrous consequences if this assumption is incorrect. This can be exemplified by Ramel's retelling of a ransomware attack where the cause of the breach was an administrator logging in to a compromised system to perform mundane tasks with admin privileges [47]. The attacker would not have been able to gain admin rights if it hadn't been for the unnecessary use of the administrator account.

Zero Trust architecture is described by CISA as fundamentally different from prior models, and one that will require a holistic change in the way an organisation thinks about its cybersecurity controls, but also its philosophy and culture around cybersecurity.

The core concept of a Zero Trust architecture is that security is no longer perimeter-based, and that there is an implicit assumed threat in any network. Therefore the security model focuses on data and service protection, and only providing the minimal level of access to resources and data when, and only when, that access is needed. Trust and access must always be continually evaluated, and an attacker won't have free reign of all resources once inside a network. The model aims to hinder the unauthorized lateral movement of attackers inside networks once the perimeter is breached [49].

The same source describes a number of factors that may enable a system to grant or deny access to a resource. A trust algorithm uses these factors to make a decision on whether or not it is likely that the request is legitimate, and denies it if it is not likely. The trust algorithm can be run multiple times during a session to continually confirm the legitimacy of the access given to a subject.

Chapter 3

Method

3.1 Chapter outline

To answer the research questions presented in section 1.2.1, we conducted a number of experiments on two different databases with two different backup solutions in place for each of them. The goal was to use our findings in the experiments to get a good overview of security features in Azure, in order to be able to answer our research questions. The basis for our experiments was three scenarios that represent different attack vectors that the databases must be secured against.

In the discussion chapter (5), we compare and contrast the different backup solutions used in the experiments. This is so that we can determine which security features are worth implementing, how secure they are, and how they impact the overall backup architecture. This is also used to help answer the research questions. More general requirements for backup solutions, such as performance and cost, were also considered. Because of this, a performance test was performed for each backup solution in order to aid in the comparison.

In this chapter we present the method for our analysis, and describe in detail how we worked to find the results presented in chapter 4. We start off with describing the criteria used in our analysis in section 3.2. In section 3.3, we present the two databases we implemented backup solutions for, and their backup solutions. We then describe how they were deployed in section 3.4.

The scenarios are at the center of our analysis. Their results are based on the criteria in 3.2, and are performed in the test environments described. The scenarios are described in section 3.5. For each of the three scenarios, several experiments were performed is described in detail. Finally, in section 3.6 we describe how the performance of the backup solutions were tested.

3.2 Criteria for analysis

The backup solutions we describe in 3.3 were evaluated based on the criteria we describe in this section. These are important criteria to assess the security of the different backup solutions.

3.2.1 Resistance to ransomware attacks

Since the topic of our thesis was to analyse cloud backup architectures with regards to their resistance to ransomware attacks, resistance to ransomware is the most important criteria to evaluate backup solutions by. Though this is a measure that can be hard to quantify, several factors are taken into consideration when evaluating the ransomware resistance of a backup solution.

Specific properties to keep in mind when evaluating the ransomware resistance of a system are:

- Ability to recover encrypted files.
- Ransomware detection.
- Ability for backup to withstand encryption.
- Ability for backup to withstand deletion.

Each backup solution's resistance to ransomware was assessed qualitatively.

3.2.2 Ease of use

There is often a trade-off between security and ease of use. Disconnecting a database server from all networks would essentially render it immune to ransomware attacks over the network, but it would also be incredibly inconvenient. An acceptable backup solution needs to be secure enough to be able to resist ransomware attacks, while at the same time being easy to set up, automate and maintain. Humans are fallible, and if the backup solution is too troublesome to use, the maintainers might neglect to verify that everything is working correctly. If security measures interrupt the operations of those using the services that are being protected, the users might seek out ways to bypass them.

Each backup solution's ease of use was assessed qualitatively throughout the experiments, based on our experience using them. We focused on the ease of setup, ease of maintenance (regular testing), and ease of recovery in the event of an attack.

3.2.3 RPO and RTO

Recovery Point Objective (see 2.1.6) and Recovery Time Objective (see 2.1.6) are two essential parameters to keep in mind when evaluating backup architectures. It is up to each organization to determine their own required RPO and RTO, based on cost-benefit analyses and other considerations.

RPO was analyzed by looking at the options provided by each backup solution with regards to options for specifying the frequency of creating recovery points. RTO was analyzed by conducting performance tests (see 3.6).

3.2.4 Cost

More secure solutions are generally more expensive. Cloud platforms charge for the amount of data stored in their services, and that includes backups. While we can't decide if a backup solution is too expensive or not, we considered the cost of the different solutions in our general comparison.

3.3 Backup solutions to analyze

In our analysis we considered each scenario, and created experiments for backup solutions for two different databases. The two databases were an unmanaged ClickHouse database running on an Azure VM, and an Azure Database for PostgreSQL instance, which is a managed database in Azure. This means that each scenario is analyzed twice; once for each of the two databases. The databases are detailed in sections 2.6, and 2.7 respectively.

3.3.1 Backup solutions for ClickHouse

Our experiments focused on two backup solutions for ClickHouse. These were Azure Backup for VMs, a service provided by Azure, and clickhouse-backup, a third-party tool for backing up ClickHouse databases specifically.

Azure Backup

Azure Backup (see 2.5.1) was chosen as one of the backup solutions to test for ClickHouse. It provides tools for easily configuring automatic backup for Azure VMs. Since it backs up entire VMs, and not just the specific database's data, it can essentially be used to back up any service running in a VM.

Since Azure Backup stores backed up VMs in Recovery Services vaults (see 2.5.1), all the security features of these newer vaults are available for this backup solution.

clickhouse-backup

Of all the backup solutions listed in the ClickHouse documentation (see 2.6.2), clickhouse-backup seemed like the most appropriate one. clickhouse-backup can back up data locally and remotely, and supports integration with different cloud providers. One of the supported remote storage solutions is Azure Blob Storage (see 2.5.7), which is a cheap storage option.

Duplicating Source Data Somewhere Else was ruled out because it requires the use of a persistent queue, which is not relevant to our case. Filesystem snapshots were ruled out because we are not using a filesystem with snapshot functionality. clickhouse-copier was ruled out because it is designed for petabyte-sized databases [43], which seems overkill for our use case.

3.3.2 Backup solutions for Azure Database for PostgreSQL

For PostgreSQL we also tested two different backup solutions. Azure Backup for PostgreSQL, and the Point-in-time-restore (PITR) functionality of the Azure database for PostgreSQL service. These services were able to be used separately or together as part of a holistic solution.

Azure Backup for PostgreSQL

Azure Backup started supporting PostgreSQL in January 2022. Because it is still relatively new it is not supported by Recovery Services vaults. Instead, it uses the older Backup Vaults in Azure. This means that far fewer security features are supported compared to the Azure Backup solution for VMs. Even so, this solution supports 10 years of data retention, scheduled and on-demand backups and backup restoration, either directly to a database or as a file in Azure Blob Storage.

Point-in-time Restore

Point-in-time Restore is a feature of Azure Database for PostgreSQL that allows restoring the database to any point within the last 35 days, depending on settings. It is described in more detail in 2.7.3. This provides a lot of flexibility when it comes to choosing which restore point to restore to.

3.4 Test environments

In this section, we describe how the test environments were deployed and configured in detail.

3.4.1 Illustration of test environment

The test environment is illustrated in Figure 3.1, and shows the databases we used in our analysis, as well as their backup solutions. The red bordered area shows where the backups are stored within the architecture. Each scenario will attack some part of this architecture, and the experiments will show how to recover the lost data, or otherwise mitigate the risk associated with the attack.

3.4.2 Test environments for ClickHouse

Test environment for non-performance sensitive tests

In order to perform the experiments where performance was not being measured, we set up a simple test environment. The test environment consisted of a single Linux VM running Ubuntu 16.04. The VM size selected was "Standard B1s" [50], which is a general purpose VM with a single vCPU, 1GB of RAM, and a 30GB SSD. ClickHouse and clickhouse-backup was installed on the VM and configured. In addition, Azure Backup was set up. A data set of around 700MB (when compressed

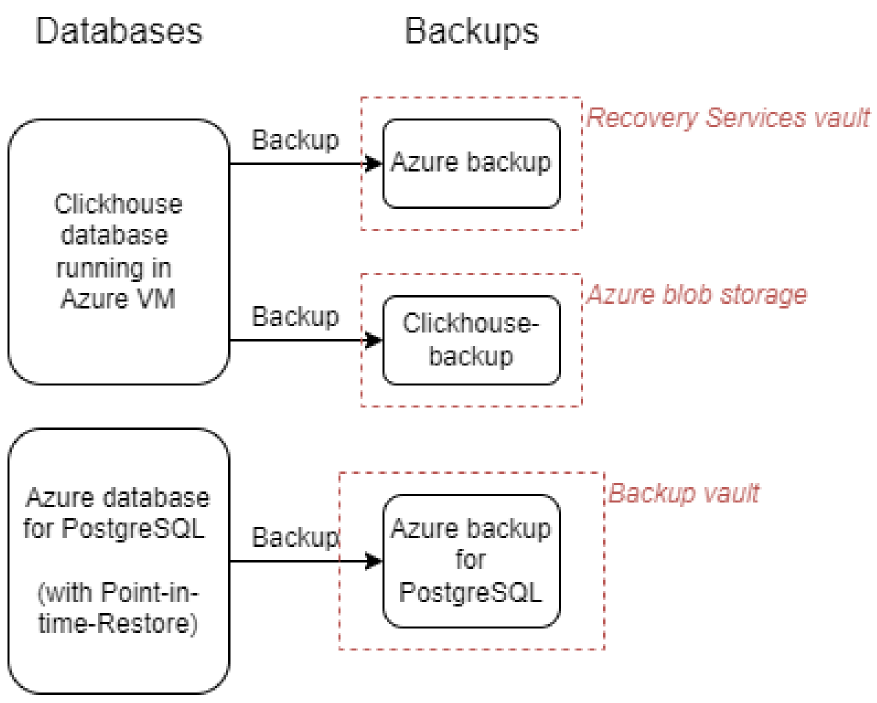

Databases
Backups
Clickhouse database running in Azure VM
Backup
Azure backup
Recovery Services vault
Backup
Clickhouse-backup
Azure blob storage
Azure database for PostgreSQL
(with Point-in-time-Restore)
Backup
Azure backup for PostgreSQL
Backup vault

Figure 3.1: Illustration of the databases and their backup solutions

by ClickHouse) was used. A detailed overview of how the non-performance critical test environment was set up is included in the appendix, see A.1.

Test environment for performance tests

For performance tests, a different test environment was used. A more powerful VM was necessary in order to avoid bottlenecks that could interfere with the performance test results. The VM size selected was "Standard D4s v4" [51]. This is a general purpose VM with 4 vCPUs and 16GB of RAM. 16GB of RAM is less than the amount recommended for ClickHouse [52] but greater than the minimum requirement [53]. The reason we chose this VM size was because it was the largest size available with our Azure subscriptions vCPU quota. This is unfortunate, but we believe it was sufficient for our tests.

The OS disk size selected was 2048GB, which was enough to leave some overhead after loading the test data. The uncompressed size on disk of the data set used was around 5TB. The data set was compressed to a size of 955GB by ClickHouse. We believe this amount of data should be enough to give a realistic overview of the performance of each backup solution.

For a detailed overview of how the performance test environment was set up, see A.2 in the appendix.

3.4.3 Test environment for PostgreSQL

We tested Azure database for PostgreSQL with the `GP_Gen5_2`-SKU, a general purpose configuration that is similar to the needs of most businesses. The setup was done with a PowerShell script for repeatability, and can be seen in appendix A.15.1. Point-in-time-restore is enabled by default (and can not be disabled.)

Connecting to the PostgreSQL instance was done with pgAdmin 4. This is a tool used to manage PostgreSQL databases. Another option here would have been using Azure Data Studio.

We populated the database with data using a script. See appendix A.15.2 for details.

In order to protect PostgreSQL with Azure Backup, we created a Backup vault and deployed a backup instance with an accompanying backup policy for the PostgreSQL server within it. The process is described in detail in appendix A.15.3.

3.5 Scenarios

In order to evaluate the effectiveness of each of the backup solutions, we came up with three practical scenarios. These are imagined attacks against the databases or backups, which the backup architecture has to be able to withstand.

These scenarios were used as a basis to create experiments, which are practical demonstrations of how an attack could be carried out, and how the system would recover from the attack, or otherwise how the risk of the attack may be mitigated. The scenarios are also used to provide a meaningful context for discussing each backup solution's resistance to ransomware.

3.5.1 Scenario 1: Attacker encrypts database data

This first scenario is perhaps the most central scenario for the project. This is essentially a basic ransomware attack, where only the data stored on the database servers are encrypted. This attack is illustrated in figure 3.2.

Conditions

This scenario assumed an attacker has managed to gain access to the database-server, or the server that configures it. The attacker has also managed to elevate their privileges enough to delete or encrypt data. The attacker is not necessarily sophisticated enough to look further into the system, or attempt to counteract the security controls in place.

Consequences

The worst case consequences of this scenario is complete data loss. Given that the attackers are using a modern strain of ransomware with sophisticated encryption schemes, as it is fair to assume, the encrypted data could be completely lost. This

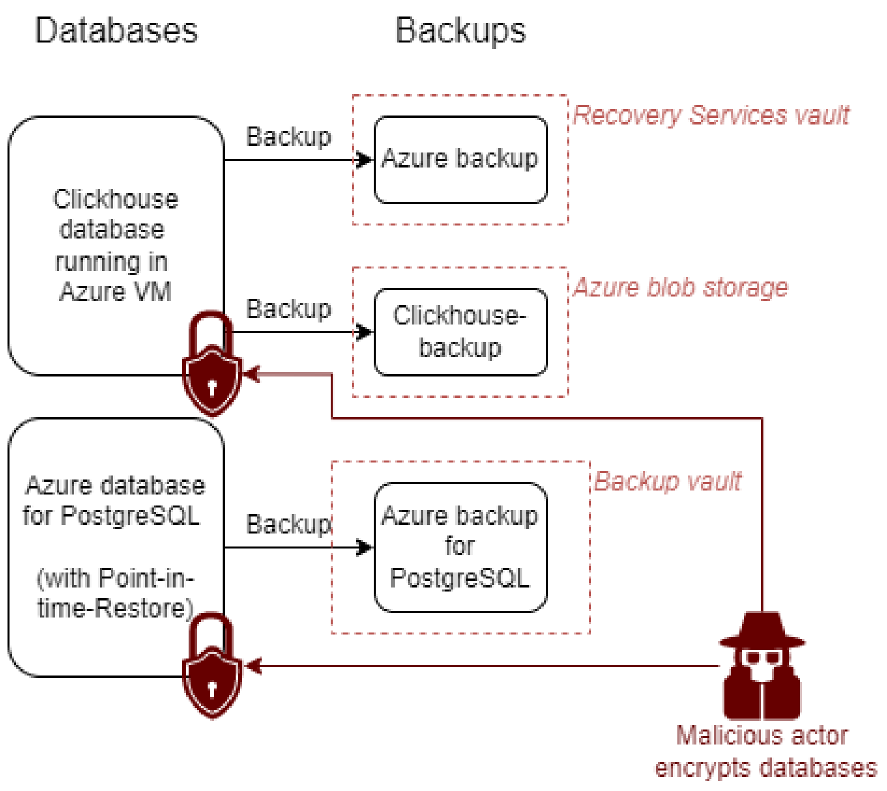

Databases
Backups
Backup
Azure backup
Recovery Services vault
Clickhouse database running in Azure VM
Backup
Clickhouse-backup
Azure blob storage
Azure database for PostgreSQL (with Point-in-time-Restore)
Backup
Azure backup for PostgreSQL
Backup vault
Malicious actor encrypts databases

Figure 3.2: Illustration of Scenario 1, a malicious attacker encrypting databases

attack without a plan for data restoration could force the victim organisation to pay the ransom or risk bankruptcy especially if the data is business-critical.

Risk Mitigation

Recovering from this attack involves restoring the data in the databases from backup. As the goal of the attack is to ensure the organisation loses access to their data, and making the ransom the only viable method of recovery, our goal should be to ensure another method of recovery exists.

In our case, recovery from this scenario involved verifying that the backup solutions reliably backed up data, and that the data restored from them was the same as we backed up.

Testing

To test this scenario we encrypted or deleted data in each database. Each of these has backups that we then restored from. In addition to testing that restoring from these backups works, we noted significant aspects of the recovery process in order to better evaluate the non-functional attributes of the backups.

ClickHouse experiment: To test this scenario with ClickHouse, we ran an experiment where we encrypted certain files in `/var/lib/clickhouse/`. This is the default storage path for ClickHouse. The files were encrypted using an encryption tool called `ccrypt`. The full details of how this experiment was conducted can be found in section A.5 of the appendix.

A short summary follows here.

The files we encrypted were files that correspond to columns in a database table. We tried encrypting both a single file and the whole directory containing the files. The effects the encryption had on the database were observed. Then, the database was recovered using backups from clickhouse-backup and Azure Backup.

The following files were encrypted:

- `/var/lib/clickhouse/store/c28/c283470d-9ab3-4be8-bd81-132274c9f9b0/all_1_35_2/radio.bin`
- `/var/lib/clickhouse/store/`

(See section 4.1.1 for results.)

PostgreSQL experiments: We find it unlikely that a normal ransomware virus would be able to encrypt data in a managed PostgreSQL database. It is however not unlikely that future threat actors target managed cloud services to a greater degree than is normal now, as more organisations move to the cloud. To test this scenario we simulated an attack where the tables in the database were dropped, instead of encrypting them. This gave us an opportunity to test how recovery is performed with each backup solution.

The first backup method that was tested was Point-in-time restore (PITR). The table was dropped and then restored with PITR. The experiment is described in detail in the appendix (A.16).

To test recovery with Azure Backup for PostgreSQL, we once again dropped the table and started recovery. The Backup vault GUI for Azure Backup for PostgreSQL was used to restore to a new database from the latest backup available. This is described in detail in appendix A.17.

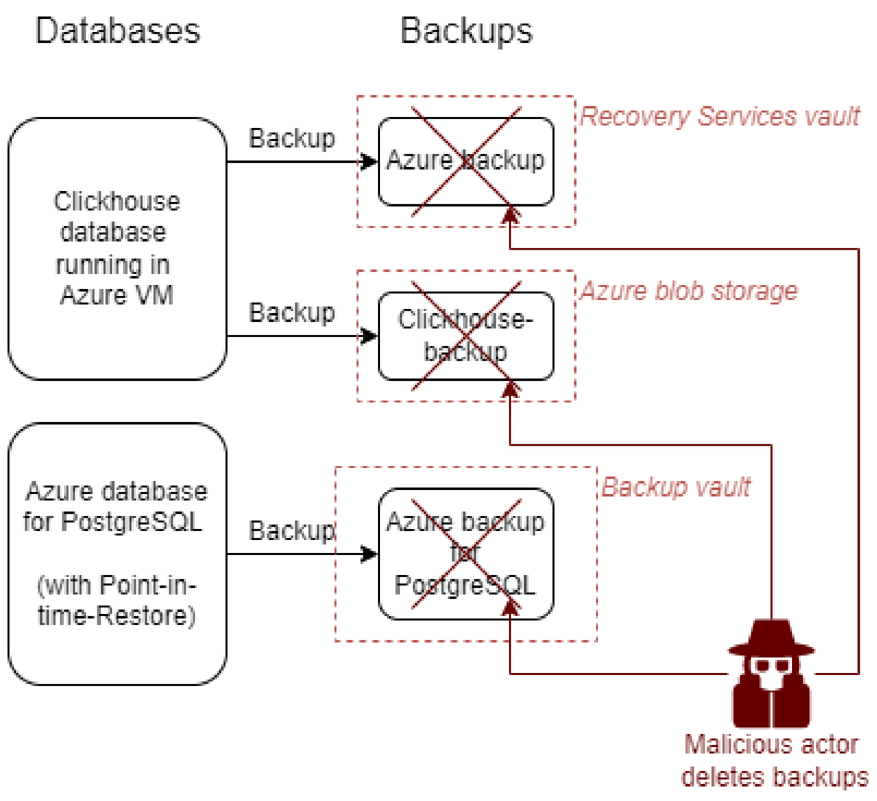
(See section 4.2.1 for results.)

3.5.2 Scenario 2: Attacker deletes backups

In a human-operated ransomware attack, it is likely that the attacker would attempt to delete backups before deploying the ransomware itself. Figure 3.3 shows the attacker deleting the backups in our test environment.

Conditions

The attacker would need access rights at a high enough level to threaten the backups. Whereas in scenario 1 the attacker only needed access to the specific data


Databases
Backups
Backup
Recovery Services vault
Azure backup
Clickhouse database running in Azure VM
Backup
Azure blob storage
Clickhouse-backup
Azure database for PostgreSQL

(with Point-in-time-Restore)
Backup
Backup vault
Azure backup for PostgreSQL
Malicious actor deletes backups

Figure 3.3: Illustration of Scenario 2, a malicious attacker deleting backups

they attempted to ransom, the attacker would in this scenario also require access to the backup administrator's Azure account, or an account with equivalent privileges over the backups.

Consequences

This attack is unlikely to be devastating on its own, but if the attacker is able to delete or encrypt the data in the databases as well, this could be crippling for an organisation. The ultimate consequence of such an attack would be total data loss.

Risk Mitigation

Preventing this scenario from even occurring is critical to any backup implementation. If for example a compromised administrator has access to both the database and the backup, they become a single point of failure for the whole system.

A number of attacks may be avoided if there are routines to monitor logs and alerts that may tell the organisation that something is wrong. Even if an attacker is able to bypass the security features protecting the backups, there ought to still be someone monitoring any log entries or alerts that this is happening. This would let the organisation preemptively try to mitigate the attack, and ideally lock the attackers out of the network.

Even if the backups are deleted, they may still be recoverable. By default Recovery service vaults has soft delete (see 2.5.4) enabled, which would allow the organisation to recover their backups, and then recover their data from them. Note that Backup vaults do not supports soft delete. Soft delete can also be enabled for Azure Storage Blobs.

Testing

In order to test this scenario, we attempted to delete backups made with the different backup solutions. After this we attempted to restore them and recover data from the now undeleted backups. Where that was not possible we also looked into other security mitigation possibilities.

ClickHouse experiment: For ClickHouse, four experiments were performed. These all involved destroying backups in different ways.

In the first experiment, local backups made with clickhouse-backup were encrypted using the tool `ccrypt`, and the results were observed. We attempted to list and restore the encrypted backup using `clickhouse backup list` and `clickhouse backup restore` respectively. See the appendix (A.6) for full details regarding how the experiment was carried out.

Clickhouse-backup has commands for deleting local and remote backups (`clickhouse-backup delete [local/remote]`). In the second experiment, backups made using clickhouse-backup were deleted using these commands. This was done in order to simulate an attack where the attacker has root access to the VM, but not to the Azure environment itself. Afterwards, the deleted backups were recovered by undeleting the Blobs via the Azure CLI, and then restoring the database with `clickhouse-backup restore_remote`. See the appendix (A.7) for the full details.

The third experiment was similar to the second, in that remote backups made using clickhouse-backup were deleted. This time, they were deleted using the Azure CLI, in order to simulate an attack where the attack has access to an Azure Cloud Shell session. Afterwards, the backups were once again recovered by undeleting the Blobs using the Azure CLI and then restoring the database with `clickhouse-backup restore_remote`. See the appendix (A.8) for the full details.

In the fourth experiment, backups made using Azure Backup were deleted, and then restored from their soft deleted state. Both deletion and restoration was performed using the Azure CLI. See the appendix (A.9) for the full details.

(See section 4.1.2 for results.)

PostgreSQL experiments Backups for a managed PostgreSQL database are inherently tied to the database, due to its point-in-time restore (PITR) function. To test this scenario, we therefore deleted the database server instance in an attempt to delete the backup, and then restored it from PITR. When creating a new managed PostgreSQL Azure Database the option is given to restore from a PITR-point. This allows the database to be "undeleted" in practice, as long as it is restored within the restore period. This process is shown in A.18.

As Backup vaults, which Azure Backup for PostgreSQL uses, does not support soft delete functionality, the experiment for the Azure Backup solution was simply to delete the backup instance in the Backup vault. After doing so, there was no option to undelete or restore the backup data.

In addition to restoring the deleted backups, one important risk mitigation effort would be to ensure that the backup administrator is alerted when backup data is deleted. Azure backup does not automatically alert the backup administrator when backup data is deleted from Backup vaults.. Alerts will also have to be monitored elsewhere than the Backup Center, as no alerts have shown up here throughout our testing.

We created an Alert Rule that sent an alert each time the "Microsoft.DataProtection/backupVaults/backupInstances/delete" action succeeded within the scope of the subscription containing the Backup vault. This is the action which deletes backup instances in backup vaults. This rule was linked to an Action Group that sent an email to the backup administrator each time the Alert was delivered. This could also be sent as a phone notification or a text message. This alert rule is described in detail in appendix A.19.

(See section 4.2.3 for results.)

3.5.3 Scenario 3: Backup administrator compromised

The third scenario assumes a complete compromise of the backup administrator account. This account has administrator access to the vaults in Azure, or has other permissions allowing them to modify or delete backups, or change security settings. In the previous scenario we looked at how to mitigate risks associated with a lost or deleted backup. This time the attacker has access to the backup administrator account and can disable whatever security settings or delete whichever resources they wish.

This attack is illustrated in figure 3.4, which shows the backup administrator account controlled by the malicious actor, as well as the domain which the backup admin account controls.

Conditions

The condition for this scenario to occur is simply that an attacker has gained access to the backup administrator account. As discussed in section 2.8.2, this can easily happen through simple human error, or misuse of administrator accounts when performing mundane tasks.

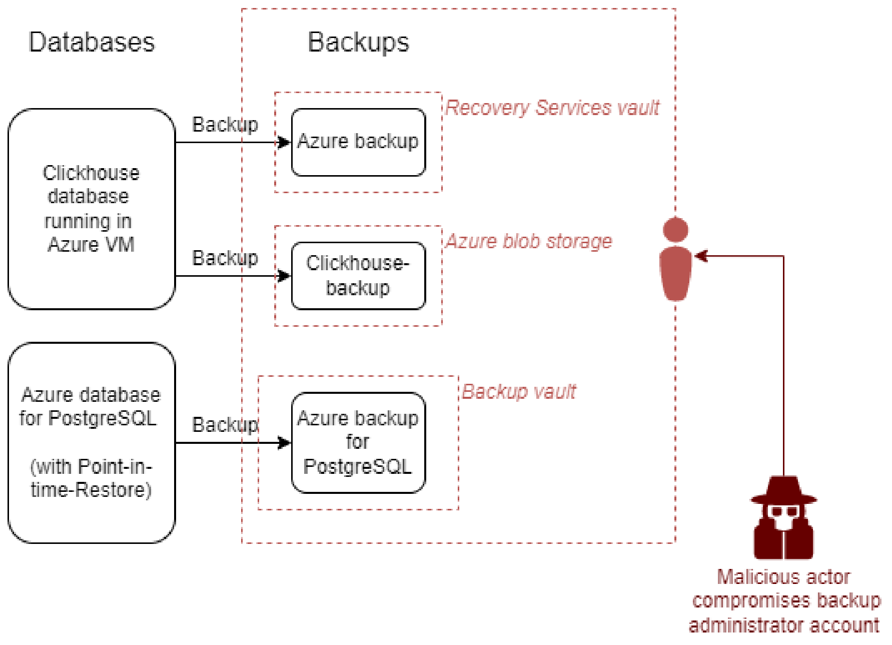

Databases
Backups
Clickhouse database running in Azure VM
Backup
Azure backup
Recovery Services vault
Backup
Clickhouse-backup
Azure blob storage
Azure database for PostgreSQL (with Point-in-time-Restore)
Backup
Azure backup for PostgreSQL
Backup vault
Malicious actor compromises backup administrator account

Figure 3.4: Illustration of Scenario 3, a malicious attacker compromising backup admin

Consequences

One of the consequences of this scenario is the permanent loss of all data, given that the attacker is able to disable soft delete and delete all backups and deploy ransomware shortly thereafter. Ransomware operators spend weeks trying to figure out all there is to know about an organization, and if they see an opportunity to attack both the databases and their backups simultaneously, they are sure to grab it.

Risk mitigation

There are security features in Azure to prevent a compromised administrator account from disabling soft-delete and deleting backups. One of them is Multi-user authorization, as described in 2.5.3. In addition to that feature in particular, well designed access control in general, based on Role-based access control, and Zero-trust features such as Just-in-time access can prevent this scenario from resulting in any major consequences.

Testing

Testing for this scenario involved exploring how much data could be made irrecoverable with full access to the backup resources in Azure,

ClickHouse experiments: We performed 3 experiments for this scenario in Click-house.

In the first experiment, soft delete was disabled for a protected item in Azure Backup before the item was deleted using the Azure CLI. See the appendix (A.10) for the full details.

In the second experiment, the entire Recovery Services vault used to store Azure Backup backups was deleted using a PowerShell script. See the appendix (A.11) for the full details.

In the third experiment, we once again attempted to run the Recovery Services vault deletion script, but before doing so, we enabled Multi-user authorization and observed what happened. See the appendix (A.12.3) for the full details.

(See section 4.1.5 for results.)

PostgreSQL experiments: Since there it is not possible to disable the soft delete feature of PITR, this scenario is less relevant for PostgreSQL compared to Click-House. Backup vaults support neither soft delete nor MUA, so this scenario does not have a direct solution using Azure Backup either.

(See section 4.2.6 for results.)

3.6 Performance testing

In order to determine whether it is realistic that a backup solution is able to recover data within a given RTO, we conducted performance tests for each backup solution. The performance tests measure how long it takes to restore a database of a certain size. This is a useful parameter to consider when comparing the different backup solutions.

3.6.1 Performance tests for ClickHouse

ClickHouse is able to handle large amounts of data, which means a backup solution for ClickHouse should also be able to. We decided to load ClickHouse with 1TB (compressed) of data, which we believe was enough to get a realistic idea of the backup solutions' performance.

For the Azure Backup test, we first prepared the environment by deleting the VM, listing the available restore points, and then selecting which restore point to use. A stopwatch was started the moment we initiated the restore job. The stopwatch was stopped when it was possible to connect to the VM using SSH and run database queries. The time used by the restore job itself is printed of the command `Restore-AzRecoveryServicesBackupItem`. This was noted separately. See A.3 in the appendix for the full details regarding how the tests were performed.

For clickhouse-backup, the plan was to measure the time it took to create a new VM with the necessary software, using a stopwatch. Then, the builtin command `time` was to be used to measure the time used by the `clickhouse-backup restore_remote`. Afterwards, the times were to be summed up to get the total restore time. However, we experienced errors we were not able to resolve, which prevented us from carrying out the test. Appendix A.4 contains details about how the tests were performed, and what went wrong. This is also discussed in chapter 4.

(See section 4.1.8 for results.)

3.6.2 Performance tests for PostgreSQL

The following experiments compared the performance of Azure Database for PostgreSQL single server when restoring using different methods. A general purpose single server with 4 vCores and 100 GB of storage was deployed in the same way as for our other testing, as seen in A.15.1.

We populated the databases with data using a script (see appendix A.15.2). The script generated 7107MB of data.

Delete server and restore using backup vault

After deleting the data in the database we restored the data to a running PostgreSQL-instance using previous backups from Azure backup. The goal was measuring the restore time, and we did this by measuring the time between pressing restore, and having access to the data again. This was sufficient for a baseline test.

To create a backup vault, a key vault was deployed which the backup vault linked to for access to the connection string necessary to work with the PostgreSQL single server.

Delete server and restore using point-in-time restore

A test similar to the previous was performed with point-in-time Restore. PITR creates a new server from the restore point, which is what was deployed in this test.

In this test as well we measured the time it took to gain access to the database after starting the restore-process.

(See section 4.2.7 for results.)

Chapter 4

Results

In this chapter, we present the results of the analysis described in chapter 3. We present these results separately for ClickHouse and PostgreSQL in order to more easily look at them holistically here and in the discussion later.

As this project analyzes the different backup solutions qualitatively, the results are based on our evaluation of how well each experiment highlights the resistance to ransomware of each backup solution, as well as their ease of use.

4.1 ClickHouse

4.1.1 Scenario 1, experiment 1: Encrypting database files

(See section 3.5.1 for method.)

In this experiment, we encrypted files in our ClickHouse database. After encrypting the database files, we expected that either the database would crash entirely, or that database queries would return strange or unreadable output. Instead, we observed that the database continued to run, but that certain queries would fail. Queries that included a column whose corresponding file had been encrypted would fail, while queries that didn't would succeed. Encrypting the `/var/lib/clickhouse/store` directory made all queries against the `cell_towers` table fail. Queries against the `system` table would succeed, though.

This experiment has given us insights about the threat of ransomware attacks against ClickHouse. Since ClickHouse is an unmanaged service that runs as a normal program on a VM, it is possible to encrypt the files used by it. In this regard, ClickHouse can be vulnerable to ransomware. The backup solutions themselves do not provide any ways to detect that the data being backed up is encrypted, as far as we know. However, since queries are likely to fail when all the ClickHouse data is encrypted, we believe it is possible that the attack would be discovered quickly, if the database is queried frequently. If this is not the case, though, the attack might go unnoticed for a long time, resulting in the backups being replaced by encrypted data over time – depending on the chosen retention time. A possible way to detect encryption could simply be to regularly test if all columns are query-able. This could be done with a simple SQL query which selects all columns. This possibility has not been explored by us, so it is hard to say whether it would be sufficient.

The experiment has also shown us how Azure Backup and clickhouse-backup can be used to recover files, which gave us useful insights into their ease of use. Recovery using Azure Backup can be done in two ways, with the Azure CLI or with the Azure Portal. We used the Azure CLI for these experiments for the sake of reproducibility. The CLI does have some advantages over the Portal, like the ability to automate recovery, as well as making it possible to automatically verify that backups are working as intended. If recovery scripts are prepared in advance, it is likely faster than using the Portal, as well as reducing the chance of human errors.

We found recovery using clickhouse-backup to be a bit more involved than recovery using Azure Backup. In order to prevent the spread of ransomware, we suggest rebuilding the VM running ClickHouse before recovering the data. Azure Backup supports replacing the existing VM when recovering, which makes this easy to do. clickhouse-backup only loads data from the backups into the database. The VM has to be rebuilt manually, or by using scripts. Then all the software has to be installed and configured before the backup can be loaded. This is quite simple, but it could be a time consuming process, unless prepared for with scripts or Infrastructure as Code-tools.

4.1.2 Scenario 2, experiment 1: Encrypting local backups made with clickhouse-backup

(See section 3.5.2 for method.)

In this experiment, we encrypted the files corresponding to local backups made with clickhouse-backup. Encrypting local backups causes visual glitches when trying to list them with clickhouse-backup.
Attempts at recovering from an encrypted local backup causes the following error: `error stat /var/lib/clickhouse/backup/local/metadata: no such file or directory`. `clickhouse-client` would not start if `/var/lib/clickhouse/backup` was encrypted.

In a ransomware attack, it is fair to assume that these backups would be encrypted along with other ClickHouse files, given that the ransomware attacks the `/var/lib/clickhouse` directory. As long as the remote backups are still working, recovery should be possible however.

Encryption of local backups could potentially be dangerous if the local backups are encrypted before they are exported to remote storage. It is difficult to say whether it is likely that ransomware could encrypt the local backup before it is exported in this case. It would depend on the sophistication of the attacker that has managed to gain access to the organisations systems.

As long as the `clickhouse-backup create_remote` command is used to create remote backups, we find this unlikely. This command creates a local backup and exports it immediately to remote storage, which we doubt many attackers would be able to circumvent.

4.1.3 Scenario 2, experiments 2 and 3: Deleting backups made with clickhouse-backup

In experiments 2 and 3, we deleted remote backups made with clickhouse-backup. In experiment 2, we used the clickhouse-backup program itself to delete the backups. In experiment 3, we used the Azure CLI to delete the backups, by deleting the Blobs the backups were stored in.

The reason we tried two methods of doing the same thing was to explore different angles of attack. In experiment 2, we assume the attacker has access to the VM, but not the rest of the system. In experiment 3, we assume the attack has access to the Azure environment.

Deleting the remote backups was possible from both angles of attack. In both cases, the backups were retained in a soft deleted state. We were therefore able to recover the backups by undeleting them. Soft delete has to be explicitly enabled for Blobs first; it is not enabled by default.

The experiments showed how soft delete can be a useful defense against backup deletion. Without it the risk of data loss is higher. The process of undeleting before recovery was also quite easy to do, and whilst undeleting increases the time spent, it is not a complicated step to add to a recovery plan.

4.1.4 Scenario 2, experiment 4: Deleting backups in Recovery Services Vaults

This experiment was similar to experiments 2 and 3. In this experiment, backups made with Azure Backup were deleted using the Azure CLI. Like with Blobs (used by clickhouse-backup), soft delete prevents the immediate deletion of the backups. Unlike with Blobs, soft delete is enabled by default for Recovery Services vaults.

Like in experiment 2 and 3, soft delete is a feature that both increases security, but without increasing complexity.

The deletion of backups in Azure Backup triggered an automatic "Delete Backup Data" alert from Azure Monitor. Alerts such as these can help an organization detect that they are under attack. Recieving such an alert should mean that the organisation is able to react much faster to a ransomware attack.

4.1.5 Scenario 3, experiment 1: Disabling soft delete and deleting backups

(See section 3.5.3 for method.)

In this experiment, we disabled soft delete for Azure Backup and then deleted the backups. This resulted in immediate deletion of the backups, with no way of recovering them. While soft delete can be a useful tool for recovering deleted backups, it is not a silver bullet. Backups made using either backup solution can be deleted instantly and permanently by first disabling soft delete. This is expected functionality, as there can be legitimate reasons to delete backups instantly. This shows that stopping malicious actors from disabling soft delete is crucial.

4.1.6 Scenario 3, experiment 2: Deleting a Recovery Services vault

In this experiment, the Recovery Services vault used for Azure Backup was deleted using a PowerShell script.

The goal was to see how fast an attacker can delete an RSV, if they have the right privileges. It turns out that this can be done very quickly. The script only takes a few minutes to complete and permanently deletes everything in the vault, with no way to recover it.

In addition to elevated privileges, access to four variables is required:

- The name of vault
- The name of the resource group
- The name of the subscription
- The subscription ID

A "Soft Delete disabled for Vault" alert was fired by Azure Monitor, but by the time this goes through, it might be to late to do anything about it.

This script can be used to quickly and easily delete all backup data stored in Azure Backup. A script could also be made for deleting a Storage Blob Container in a similar manner. In the case where an attacker obtains the necessary privileges, it seems neither backup solution is able to prevent such an attack without other protections in place.

While these experiments show that data loss is inevitable if an attacker gains access to full administrator privileges, it is not a bad thing: Unless such features are configured and implemented into a system, an administrator account Should be able to delete their own backups. Protecting against compromised administrator accounts is however necessary to avoid having a single point of failure for an entire organisations data.

4.1.7 Scenario 3, experiment 3: Preventing soft delete from being disabled with MUA

In this experiment, Multi-user authorization (MUA) was enabled for a Recovery Services vault before attempting to delete it with the script from experiment 2. With MUA enabled, we were unable to delete it. We were also unable to disable soft delete for the vault without requesting access from the security administrator.

MUA can prevent this attack as long as the attacker does not have access to both the backup administrator and the security administrator account. This greatly reduces the probability that the attacker would be able to threaten an organisations data even with access to a compromised administrator account. With MUA enabled, we believe Azure Backup would be quite resistant to backup deletion.

Unfortunately, MUA is not available for Azure Storage Blobs, which means that this extra layer of security cannot be utilized by clickhouse-backup. The risk can still be mitigated by ensuring that alerts are being properly monitored, and that administrator accounts are only used when absolutely necessary.

4.1.8 Performance tests for ClickHouse

(See section 3.6.1 for method.)

The speed of recovery using both Azure Backup and clickhouse-backup was measured, in order to compare their performance.

Azure Backup

Two successful performance tests were performed. One using the Azure Portal, and one using the Azure CLI. The time for the restore job itself is noted separately. The total time includes the time for us to paste commands, etc.

Test 1 (Azure Portal):

	Time (hh:mm:ss)
Restore job:	00:02:10
Total:	00:03:30

Test 2 (Azure CLI):

	Time (hh:mm:ss)
Restore job:	00:01:06
Total:	00:06:33

This assumes that an Instant Restore point is available. If this is not the case, the backup first has to be retrieved from the Recovery Services vault. This can take several hours, depending on various factors.

Azure Backup supports individual disks up to 32TB [54].

clickhouse-backup

Three unsuccessful attempts were made at recovering from clickhouse-backup. We were unable to solve the issue. Here is an example of an error that occurred during a download:

```
2022/05/12 10:02:43.087969 error one of Download go-routine return
error: one of downloadTableData go-routine return error: handling file:
/all_3441_4218_4/file_time.bin: context deadline exceeded
```

Because of this, we were unable to do a proper performance test for clickhouse-backup.

clickhouse-backup's GitHub page contains a short paragraph about backing up terabytes of data using clickhouse-backup [55]. They recommend making a local backup with clickhouse-backup and then using clickhouse-copier (see 2.6.2

to copy the backup to another VM. This might indicate that clickhouse-backup's remote storage feature does not scale well enough to be used for several terabyte large databases.

Despite our struggles, we still believe we were able to get a good overview of how long recovery could take. Recovery with clickhouse-backup consists of two steps: Downloading the backup and restoring the database. The first step is determined by the time it takes to transfer the files from Blob storage to the VM. The second is determined by how fast the data can be loaded in the database. While preparing the test environment (A.2), the time it took to make and upload the backup was measured. If we assume the upload and the download time for the backups are the same, we essentially know how long it would take to download the backups. What remains is the restore step. This was measured by restoring the database from the local backup of the same 1TB data set. Below is a table showing the theoretical performance of clickhouse-backup with 1TB of data.

Theoretical performance:

	Time (hh:mm:ss)
Upload:	03:18:48
Local restore:	00:00:03
Total:	03:18:51

4.1.9 Cost

The cost of both backup solutions can vary greatly based on several factors.

Prices are calculated in the North Europe region and displayed in USD.

Azure Backup cost

The pricing information in this section is based on the pricing details provided by Microsoft [56].

Azure Backup has a base cost, which is calculated per VM. For instances larger than 500GB, this cost is $10 for each 500 GB increment + storage consumed. In addition, a separate amount is charged based on the redundancy tier used (see 2.5.1 for information on redundancy tiers).

Below are two simple calculations assuming that a single 10TB VM will be backed up, and that the amount of data will not change over time.

Cost of backing up a 10TB VM using LRS:

	Cost per month
Base cost	$200
Redundancy (LRS)	$224
Total	$424

Cost of backing up a 10TB VM using GRS:

	Cost per month
Base cost	\$200
Redundancy (GRS)	\$448
Total	\$648

The number of Instant Restore points can affect the total cost of Azure Backup. According to Microsoft, the cost can be calculated with the following formula: "Snapshot retention period daily churn per VM storage cost per GB". The term churn refers to percentage of data that is changed each day. The minimum number of Instant Restore points one can have with Azure Backup is one.

The number of retention points and the length of the retention period also affect costs. This is dependent on churn, which varies from organization to organization.

clickhouse-backup cost

Pricing details are based on the pricing for Blob storage in Microsoft's documentation [57]

clickhouse-backup used in combination with Azure Storage Blobs can provide a cost effective backup solution. The pricing model is a bit simpler than Azure Backup's. Below are two simple calculations for the monthly price of 10TB of data using different redundancy tiers. The calculations assume that the "hot" storage is used.

LRS: $\$0.022 \times 10000GB = \$220$ per month.

GRS: $\$0.037 \times 10000GB = \$370$ per month.

Like the calculations for Azure Backup, these calculations assume that data will not change over time, which is unrealistic in many cases.

clickhouse-backup supports the use of incremental backups. This would make the cost calculation over time similar to Azure Backup, in that the changes

4.2 PostgreSQL

4.2.1 Scenario 1, experiment 1: Recovery with Point-in-time-Restore

(See section 3.5.1 for method.)

The first experiment of this scenario shows how easy and effective postgres' PITR-functionality is. We could restore to exactly the point in time we needed to, regardless of whether a backup had been made at that time. The lost data was recovered, and we were quickly up and running again, in part due to its ease of use.

One aspect of the restoration process that is a drawback for ease of use is the fact that a new database instance had to be deployed which the data could be restored to. This includes configuration of the new server instance in order to integrate it with the rest of the architecture, and if there is no procedure, practice, or plan in order to do this effectively, RTO may increase considerably. RTO will be looked at more closely when performance testing the backup solutions.

This experiment also shows how highly PITR functionality scores for RPO, as the organisation can restore to the state of the data only seconds before the data breach. This is because the feature uses the change-log in the database to track continuous changes to the data. However, if the breach had long been ongoing and the database filled with corrupted data, the short retention period may be detrimental. This is due to the relatively meager 35 days of retention in PITR. Whether it is likely that encrypted data in this database would not be discovered is dependent on the data type and the organisation in question.

4.2.2 Scenario 1, experiment 2: Recovery with Azure Backup For PostgreSQL

Similarly to the previous experiment, this was just as easy to use, and was also able to meet the requirement of the experiment; restoring the data. Where it falls short however is in RPO. The latest data, or recent changes to the data can only be recovered if the latest backup was made after those changes. This could result in a bigger data loss depending on the time of the attack. Whilst not an uncommon caveat of any backup solution, it's relevant to consider how important any change in data is to the organisation. The fact that Azure Backup data can be stored for up to ten years in a Backup vault is an advantage however, since long-term storage may be necessary for compliance.

This scenario shows how well PITR and Azure Backup complement each other in a holistic implementation. These backup-solutions are compatible, may be used in parallel, and work in consort. Both recovered the database easily and effectively, and were reliable in our testing. In this way an organisation can combine the great RPO of PITR, and still retain long term backups with Azure Backup.

4.2.3 Scenario 2, experiment 1: Deleting database-server and restoring with PITR

(See section 3.5.2 for method.)

This experiment largely yielded the same results as the first experiment in scenario 1. As recovery with PITR has to be made to a new server instance anyway, the difference between the two experiments was negligible.

Deleting the database-server does not delete the restore points, and there is no way of deleting restore points. This means that despite there being no way of restoring the deleted database or backup, the organisation can still recover their data through PITR.

4.2.4 Scenario 2, experiment 2: Deleting backup-instance in Backup vault and attempting undelete

This experiment highlighted some weaknesses of Backup vaults. Azure backup for PostgreSQL is implemented in Backup vaults, instead of the newer Recovery Service vaults, and lacks support for several features that increase security, especially for this scenario. Soft-delete is a great feature of Recovery Services vault, as our Clickhouse-experiments showed. This not being a feature of Backup vault makes PostgreSQL backups far more vulnerable to attacks that target the backup data.

The combination of Azure backup and PITR met the security requirements of our second scenario. The solution would however have been more robust and reliable if the PostgreSQL-database had been backed up in a Recovery Service vault instead of a Backup vault.

4.2.5 Scenario 2, experiment 3: Setting up alerts and deleting backup-instance

This experiment is not about evaluating the restoration process of either backup solution, but rather a related security feature. It was discovered in testing the other experiments, and is a relevant part of our analysis, which is why we included it.

In this experiment we discovered that Azures built in, automatic alerts for deleted backup data does not work with their PostgreSQL backup solution. The cause of this is likely that the Alerts are inherently tied to the soft delete functionality, and as Backup vault does not support soft delete, they do not support those alerts either. The fact that the documentation made it appear as if these alerts were automatic when they were not is something we consider a massive flaw of either the implementation in Azure, or the documentation.

Setting up manual alerts was an effective counter to this problem, and by the end of the experiment we were able to maintain needed functionality and get alerted whenever any backup instance was deleted from a backup vault. After implementing our own alert-rule for deleted backup instances, we consider our PostgreSQL backup architecture to be even more resilient, as administrators can be alerted if an attack is ongoing.

This scores low on ease-of-use however, and we imagine it is far easier to bypass compared to the built-in solution that worked effectively in our Clickhouse experiments.

4.2.6 Scenario 3

As discussed earlier in chapter 3.5.3, there were no experiments needed to be performed here that were not already done for scenario two. There are some remarks however.

As Backup vault does not support Multi-user authorization there was no need to attempt to enable it. Consequently, there was no way to limit the capabilities of a compromised backup admin within the scope of the Backup vault. This means that a compromised account in this role would all but guarantee the loss of all data in the vault. Instead of MUA, designing and implementing an access management design that is based on RBAC and Zero trust proved a worthy replacement. As learned in chapter 2.8.2, administrator accounts become vulnerable to risk if used more often than necessary. A large number of permissions in a single account is a single point of failure. No account should have any permissions beyond the ones they absolutely need, and administrator accounts should not be used beyond the tasks they need those privileges for.

Ensuring that the backup admin does not have permissions to delete the database instance will reduce risk in the system. Whilst not actually being MUA, it will inherently require two administrator accounts to be compromised before risk of data loss, which is more unlikely. Since the backup in Azure Backup is a different resource in Azure than the database itself, dividing responsibilities in the organisation according to RBAC-methology, and making no admin the single source of failure accomplishes some of the protection that lies in MUA.

In addition to MUA, other lacking Recovery Service vault features such as soft-delete and automatic alerts would all be helpful in this implementation. Use of PITR alongside Backup Vault largely remedied the shortcomings of a lack of soft-delete feature. Further, our implementation of alert rules in appendix A.19 was an adequate replacement for the automatic ones in Recovery Service vaults.

Azure supports a lot of fine-grained control over access rules and alerts, and this scenario showed how much an organisations security can improve with the correct implementation, despite the latest security features not being available. Even though Recovery Service vaults would have been nice to have, and would have made implementation easier and less prone to human error, the lack of that support does not mean that security is inherently compromised.

4.2.7 Performance tests for PostgreSQL

(See section 3.6.1 for method.)

A Backup vault has the ability to perform a manual full backup at any given time, which is more effective to restore to than a solution based on a range of differential backups. A full backup in a Backup vault was performed with operation details shown in the appendix (see A.13). The backup instance was then deployed to a new server, giving a recovery time of 1 minute and 54 seconds.

For PITR we used the Azure REST API to deploy a server with PITR creation mode, where the goal was to measure the time it takes to have a functioning database from a given previous state. A database equal in size to the one used in the backup vault experiment took 31 minutes and 27 seconds to deploy, see operation details in the appendix (A.14).

There was an expectation of different performance when recovering from backup vault versus using PITR. Part of this was that PITR recovery requires deploying a new server, which meant we had to count with server creation time as part of the recovery time. Our expectations were however exceeded in regards to the efficiency the vault solution exhibited with a stark difference of 29 minutes and 33 seconds, approximating to 15 times faster recovery using backup vault compared to PITR.

It is worth noting however that the backup instance in the vault was a recent full backup, while PITR most likely used transaction logs and differential backups to reconstruct the database to the specified point in time. Because of this there is a trade-off that the victim organisation must consider: Do they restore as quickly as possible to as quickly as possible resume business operations, or do they restore to the most recent possible recovery point with PITR, but spend much longer doing so? The difference may be more substantial than in our case if the transaction log is much longer than in our experiments.

4.2.8 PostgreSQL cost

Similarly to Clickhouses backup solutions the pricing for PostgreSQLs backup solutions vary depending on several factors. The PostgreSQL-service in Azure is priced depending on performance – the number of virtual cores and memory – as well as the storage used [56]. The Point-in-time-Restore data is part of this, but charges an additional amount depending on redundancy tier and storage amount. If the data changes frequently this extra storage amount will increase.

Azure Backup is priced in a similar manner, with a set price per instance, plus however much storage is used for the backup – again depending on redundany tier. Extra fees are also incurred if archived data is deleted early, or retrieved back to hot storage.

As we did not end up comparing two opposing backup solutions in PostgreSQL, we chose not to perform a detailed cost analysis, as it would be of little value to our analysis or thesis.

Chapter 5

Discussion

Our initial thesis topic was to analyze cloud backup architectures with regards to their resistance to ransomware attacks. In the following discussion, we answer the research questions we have defined 1.2.1. In order to do this we compare and evaluate the different backup solutions we tested in regards to the criteria laid out in the method chapter 3.2.

The research questions were:

Research question 1 What are some best practices for securing backups against ransomware and other malware, and how can they be implemented in Azure?

Research question 2 Are Azure's security mechanisms effective against a modern ransomware attack?

5.1 Discussion of backup solutions for each database

The purpose of discussing and comparing the backup solutions at the core of our analysis is to identify relevant findings in order to best answer our research questions.

5.1.1 ClickHouse

Based on our findings, as presented in chapter 4, we will discuss the two backup solutions we tested for ClickHouse: Azure Backup, and clickhouse-backup. The former is the solution that we found to be the most secure, reliable, and easy to use, all in all.

We found clickhouse-backup to be generally more unreliable and complicated to use in our experience. Unless the organization wishes to back up ClickHouse data outside of Azure, we do not see much reason to choose clickhouse-backup implementation in an Azure environment, as Azure Backup provides more security features and an easier setup. When we tried to recover 1TB of data with clickhouse-backup, the restore operation failed, and we were unable to find the cause. Obscure error messages are not a welcome sight when trying to recover from a ransomware attack. While we were unable to perform a proper performance test, we did get an idea of the potential recovery speed of the solution. If upload and download speeds are similar, we can expect a recovery time of around 3.3 hours for 1TB of data, which is much slower than Azure Backup's 3-7 minutes for the same amount of data. Another downside of clickhouse-backup is the lack of support for Multi-User authorization, or rather, the lack of support for MUA in Azure Blob Storage. Hopefully, Microsoft will implement MUA for Blob storage in the future.

With that being said, the solution was able to withstand two of the three scenarios, and with proper monitoring tools and Role-based access control in place there is nothing to suggest that clickhouse-backup will be an insecure solution. It has support for many of the same security features as Azure Backup, and it is also possible to put the backup in a different cloud environment entirely if needed. Because of the option to use Blob storage, the cost of clickhouse-backup is lower than that of Azure Backup. This may be a decisive factor in a cost-benefit analysis of the different alternatives.

The advantages of Azure Backup on the other hand are its ease of use, and its support for some of the better security features available, like MUA. One of the features that highlight Azure Backup's ease of use is the configuration of backup policies. Azure Backup provides great control over retention, backup frequency and data redundancy through a graphical user interface. In clickhouse-backup, configuration is done by editing a YAML file, which is far less intuitive. Another area where Azure Backup is more intuitive than clickhouse-backup is error messages. We found that Azure Backup provides clear and understandable error messages, both in the CLI and in the Portal, when something goes wrong. The big disadvantage compared to clickhouse-backup is however the price for the service.

Soft delete and Multi-user authorization are two security features that have proven to be very effective. When used together, as they are designed to be, they make it practically impossible to lose data without the attacker compromising two specific administrator accounts – which we find unlikely.

Our experience with restoring from the Azure Backup were also positive. When Instant Restore points were available, the performance was excellent – and far more reliable than clickhouse-backup. After our experiences with restoring from both backup solutions the Azure Backup process was simple, effective and reliable. Exactly what you want from a backup solution.

While Azure Backup locks the organization in to a single cloud service provider, and that may have its disadvantages, there are also advantages to using a single platform. Azure Backup provides central monitoring of backups via the Backup Center service, which also provides a good overview to ensure compliance and security. Security should be easy and automatic, and Azure Backup manages to go a long way to get there.

It seems Azure Monitor has a higher degree of integration with Azure Backup than with Blob storage. After all, Blobs are a general purpose storage solutions, while Azure Backup is more specific.

5.1.2 PostgreSQL

For PostgreSQL, we are not as focused on comparing alternative backup solutions, as both Azure Backup for PostgreSQL and Point-in-time Restore can, and should be, used in combination. These two solutions complemented each other well in a common architecture. Neither service did everything perfectly, but combining the great RPO, soft delete, and ease of use of PITR, with the longer retention and more granular control of Azure backup provided an excellent total package.

The overall process of backing up and restoring a managed PostgreSQL instance in Azure left us impressed with the ease of use and efficiency of the processes. Setting up a single server instance of Azure Database for PostgreSQL and then backing it up in a Backup vault was a quick and intuitive, and with the automatically enabled Point-in-time-Restore (PITR)-feature, restoring to any previous point in time is easy.

By default, there are no automatic alerts for critical actions performed on backup data in a Backup vault. This means that if an attacker has managed to compromise the backup administrator, and then deletes the backup instance, it is possible no one might know, unless they actively and manually monitor the Backup Center. Manually enabling alert rules for backup delete actions is critical when relying on a Backup vault in a production environment. Testing that this functionality is set up correctly is also important, as our experiments showed that the alert system is somewhat fiddly in the way that it has to be implemented.

With no soft delete or Multi-user authorization support, a compromised backup admin can cause great damage to backup instances. Luckily PITR is supported even if a database instance is deleted. So both restoring to an earlier state, and undeleting a database instance is possible, as long as one operates within the configured retention time of up to 35 days.

An important security feature that is supported by Backup vaults is RBAC. As mentioned, the backup administrator becomes a single point of failure for the Backup instance, so separating the backup administrator from accessing administrator privileges for the database instance is critical to security. This ensures no single point of failure for the database data itself, and essentially becomes an air gap between the database and its backup.

To minimize RTO we recommend restoring from the Backup vault data, as its restore times generally were much quicker, compared to PITR. However, depending on the frequency of backups as per the backup policy, it might prove more effective with regards to RPO to choose a specific PITR-point to restore from.

As discussed in the previous chapter, Backup vaults have less support for the newer security features in Azure than Recovery Services vaults, and as such fewer security features are available for Azure Backup for PostgreSQL, than in Azure Backup for VMs. Despite this we found the backup solution we tested to be quite secure.

While a secure and effective backup architecture can be built for Azure database for PostgreSQL, there is no denying that Recovery Services vaults are inherently more secure than Backup vaults. Recovery Services vaults support features such as MUA, soft delete, and automatic alerts. As long as only Backup vaults support PostgreSQL servers, they will be secure as long as the overall level of security in the organisation is high enough. Given that with time, Azure backup for PostgreSQL support is added to Recovery Services vaults, any organisation should upgrade to a Recovery Services vault as soon as it becomes available.

5.2 Research questions

The purpose of this project was to answer the research questions from chapter 1.2.1, using our experience with the backup solutions for the two different databases in Azure to do so.

5.2.1 Research question 1

What are some best practices for securing backups against ransomware and other malware, and how can they be implemented in Azure? Protecting backups from ransomware is different in the cloud age, than in the time of physical infrastructures, but the principles largely remain the same. It is important to protect against unauthorized access to the backups, and against modifications or deletion. The CIA triad was described in section 2.1.2, and all three principles are important for securing backups.

Protecting against unauthorized access is important to ensure both the confidentiality, and integrity of the backups. This can be done outside of cloud platforms with physical barriers separating the servers from unwanted persons. The use of passwords or other access control mechanisms is also standard practice. In the cloud age however, everything is available over the internet, and the security controls must address this.

In Azure unauthorized access is hindered with Role-based access control (see 2.5.2 for details). Through the use of strict permissions and roles for all users, the system can prevent unauthorized access to specified resources or resource groups. This means that only a very limited set of people can access the backup data, and even fewer can perform modifications to it, when implemented properly.

Backups must also be protected from changes to ensure the integrity of the data. It is essential when a backup is restored to a production system that the data can be trusted to be the same as when it was backed up. In Azure Backup there is no way to change backups after they are created. When a backup job runs, it is allowed to write the backup into storage, but after the backup job is finished, the data is read-only. This ensures that an attacker can not encrypt backups directly, or tamper with them by any other means than deleting them.

Naturally it follows that backups must also be available to authorized personnel in a timely manner. This means that they for example should not be able to be deleted by unauthorized users. In Azure Backups Recovery Services vaults, this can be ensured with several features. Soft delete prevents all backup data from being permanently deleted within 14 days of attempting to delete it. During this time, it is possible to recover the data by undeleting it. Multi-user authorization prevents data from being deleted (and soft delete from getting disabled) without the authorization from another administrator account beyond the backup administrator.

In summary there are security features for backup solutions in Azure that address the follow the best practices of security controls. These secure the confidentiality, integrity, and availability of backup data stored in Azure.

5.2.2 Research question 2

Are Azure's security mechanisms effective against a modern ransomware attack? Modern ransomware is created by professional threat actors, and the modern encryption schemes used are not going to be able to be bypassed without the decryption key. Falling victim to ransomware will require data recovery from some other source. Human-operated ransomware is also usually preceded by weeks of reconnaissance, and all the systems the attacker is able to reach will likely be targeted by the ransomware, or otherwise attacked to increase the highest likelihood of the ransom being paid.

The weakness of off-site backups on systems that are still available over the internet is that the ransomware very well could target them as well. This is one of the advantages of Azure Backup – the vaults provide a way to store backup data in a read-only state, which makes it impossible to modify that data, apart from deleting it. New backups in Azure Backup do not overwrite existing ones, until the retention period for the older backups runs out. That means that a compromised system that is getting backed up cannot tamper with old backups by sending bad data.

This means that one of the major vulnerabilities of Azure's backup solutions is going to come from a compromised backup administrator account. Our scenarios addressed this, and the backup solutions for each database was able to resist these scenarios. This indicates a resilience and reliability of the backup solutions which meets the standards of security that organisations should expect.

Separating privileges, which can help eliminate single points of failure, as well as lowering the chance of unauthorized access, is crucial. It also lowers the risk of backups getting deleted or tampered with in some way. We believe Azure's implementation of RBAC is effective, based on our tests. Combined with MUA, RBAC appears to be very secure.

Soft delete was an essential security feature in Azure, and the biggest disadvantage of this feature is that it is not supported by more services in Azure. Throughout our analysis it has been one of the most important features to ensure the availability of backup data. Together with MUA, this made backup data in Recovery Services vaults as secure as we could hope for, by preventing data loss even with a fully compromised backup administrator account.

On the other hand, it is noticeable that the features in cloud platforms are under constant development. Protecting backup data in Backup vaults (used by Azure Backup for PostgreSQL), compared to Recovery Services vaults (used by Azure Backup for VMs) is more difficult. The addition of new features may introduce bugs or security flaws, which puts the customer at risk. The customer has to trust that the cloud provider will fix such vulnerabilities. Part of the responsibility of ensuring security is shifted from the administrators to the cloud services providers, which has both advantages and disadvantages.

In total it is our evaluation that Azure's security mechanisms do protect against modern ransomware attacks, but some more than others. Azure Backup supports many different services and supports a number of different workloads, but the security features supported for each service varies. In the case of Azure Backup for VMs, we believe it is sufficient to prevent modern ransomware attacks, if MUA is configured. Azure Backup for PostgreSQL is still lacking some features like MUA, making it less resilient against ransomware. The existence of Point-in-time restore mitigates the shortcomings of Azure Backup in this specific instance, though. In general, Azure provides the tools needed to protect against a modern ransomware attack, and in the case of services that use Recovery Services vaults, it does most of this right out of the box.

5.3 Future work/Limitations

This thesis explored several different technologies that all could have been explored in much more depth if we had chosen to do so. As we limited the scope of the thesis to a less detailed view of each of the databases backup possibilities, we list some of the areas we chose not to explore further.

PostgreSQL Infrastructure Double encryption effect: Future research could measure whether there is a noticable difference with and without the use of infrasctructure double encryption feature when creating the PostgreSQL-server instance. To avoid expected margin of error the database should be scaled up to 40 GB, and ought to be coupled with more thorough performance tests to evaluate how different features affect performance

PostgreSQL Flexible server, being burstable effect on restore time It is also possible to test out whether having a burstable flexible server affects restore time where adding to vCores when restoring will decrease the restoration time compared to operating on the normal amount used in a general purpose single server.

This should also use scaled up databases (40 GB),

More backup solutions for ClickHouse There were a number of backup solutions for clickhouse that we chose to not explore beyond the theory. Limiting the number of technologies we had to explore was necassary, but future work could explore a wider range of backup solutions.

Successful performance test for clickhouse-backup Whilst initially within the scope of this thesis, we were unsuccessful in performing a performance test for clickhouse-backup. As such, we are not sure how this compares to the performance of Azure backup.

Multi-cloud backup using clickhouse-backup One of the advantages of clickhouse-backup was that the data could be stored wherever you choose. The potential advantages of using clickhouse-backup to store backup data in a different cloud environment should be explored in future. A multi-cloud backup solution in general is something this thesis did not explore, but something that would be interesting.

Remote Backup for PostgreSQL Look into the possibilities the pg_dump tool provides when it comes to having an air-gapped remote backup instance, in addition to the backup architecture used in our experiments.

Chapter 6

Conclusion

6.1 Summary

In our thesis we asked two research questions. What are some best practices for securing backups against ransomware and other malware, how can they be implemented in Azure, and are these security mechanisms effective against a modern ransomware attack?

We have described the threat that modern ransomware poses, and some ways in which the security of a backup architecture can be breached. Based on this, we showed how a backup architecture in the cloud was likely to be attacked with three different scenarios. These scenarios each had a number of experiments which examined the capabilities of the different backup solutions. On a larger scale these experiments provided insight into what some best practices for securing backups against ransomware could be in Azure.

The backup solutions we analyzed secured an unmanaged ClickHouse database running in an Azure VM, and a managed PostgreSQL database hosted in Azure. Our analysis was largely focused on a qualitative analysis of the security features available for these backup solutions, as well as requirements like cost, performance and ease-of-use. Our findings showed that Azure is a platform in continuous development with many services further in their development lifecycle than others. This was evident when comparing the backup solutions available for each of the databases where the effectiveness of the security mechanisms varied considerably.

One of our main takeaways was the importance of features such as multi-user authorization in conjunction with role-based access control to ensure elimination of single points of failure, or soft delete-functionality to hinder data loss due to unauthorized deletions. Where these features were missing, workarounds could be designed to achieve some degree of the same security. Our results show that Azure's security mechanisms are effective against a modern ransomware attack, given that they are implemented correctly.

6.2 Future developments

Security features are constantly in development, as shown in this thesis. The same is true for malware development. Malicious actors are always exploring new ways to exploit vulnerabilities and attempt to profit. The ransomware of today is the way it is because our architectures are designed with traditional server technology, and when that changes, so will the malware.

We can't predict the future, but as the world moves towards managed services hosted on cloud platforms, malware will follow. The same security principles will remain true, but the implementation differs. While it was hard for us to see how ransomware could attack a managed database, malicious actors are without a doubt exploring how to threaten those services as well.

Future work in this field must keep working to ensure security before malicious actors have a chance to profit.

6.3 Greater context

This thesis only looks at a small section of the security controls that an organisation need to implement in order to keep their data secure. We described backups as the last line of defense, and that means they should hopefully never be needed, and the guard should not be lowered even if a secure backup solution is implemented. Other security controls must ensure that.

These security controls include the culture in the organisation, to make sure that employees and personnel maintain a security-focused mindset at all times. As explored in this thesis, human error is a major vulnerability, and much of that can be lowered by working with the people that use the systems every day.

Another important aspect is the other security controls in place. In addition to secure backups, the data in production systems must remain secure as well. The computer systems of an organisation are still targets of attacks even if they neither have sensitive data nor access to backups.

Secure backup solutions are important, but they are not the whole picture. This thesis considers only one part of the puzzle, one link in the chain. Malicious actors will no doubt stretch the chain to look for the weakest link. The entire system must therefore be secure.

Bibliography

[1] N. A. Hassan, “Ransomware Overview,” en, in Ransomware Revealed: A Beginner’s Guide to Protecting and Recovering from Ransomware Attacks, N. A. Hassan, Ed., Berkeley, CA: Apress, 2019, pp. 3–28, ISBN: 978-1-4842-4255-1. DOI: 10.1007/978-1-4842-4255-1_1. [Online]. Available: https://doi.org/10.1007/978-1-4842-4255-1_1 (visited on 02/09/2022).

[2] K. Waddell, The Computer Virus That Haunted Early AIDS Researchers, en, Section: Technology, May 2016. [Online]. Available: https://www.theatlantic.com/technology/archive/2016/05/the-computer-virus-that-haunted-early-aids-researchers/481965/ (visited on 05/15/2022).

[3] Threat Intelligence Report: Ransomware Threat Actors & Victims. [Online]. Available: https://abnormalsecurity.com/resources/ransomware-victims-threat-actors (visited on 03/01/2022).

[4] T. Seals, 2021: The Evolution of Ransomware. [Online]. Available: https://media.threatpost.com/wp-content/uploads/sites/103/2021/04/19080601/0354039421fd7c82eb4e1b4a7c90f98e.pdf (visited on 05/02/2022).

[5] F. Richter, Amazon Leads $180-Billion Cloud Market, en, Feb. 2022. [Online]. Available: https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/ (visited on 05/12/2022).

[6] P. Hofmann and D. Woods, “Cloud Computing: The Limits of Public Clouds for Business Applications,” IEEE Internet Computing, vol. 14, no. 6, pp. 90–93, Nov. 2010, Conference Name: IEEE Internet Computing, ISSN: 1941-0131. DOI: 10.1109/MIC.2010.136.

[7] tfitzmac, Azure Resource Manager overview - Azure Resource Manager, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/overview (visited on 05/14/2022).

[8] S. Laan, IT infrastructure architecture: infrastructure building blocks and concepts, eng, Third edition. Morrisville, NC: Lulu Press Inc, 2017, ISBN: 978-1-326-91297-0.

[9] Singh, Understanding RPO and RTO, en-US, Oct. 2021. [Online]. Available: https://www.druva.com/blog/understanding-rpo-and-rto/ (visited on 05/20/2022).

[10] New Cybereason Ransomware Study Reveals True Cost to Business, en. [Online]. Available: https://www.cybereason.com/press/new-cybereason-ransomware-study-reveals-true-cost-to-business (visited on 05/14/2022).

[11] Å. H. Stemland, Cyberkriminelle følger pengene: Vekst i ulovlig kryptoutvinning, Aug. 2021. [Online]. Available: https://www.digi.no/artikler/cyberkriminelle-folger-pengene-vekst-i-ulovlig-kryptoutvinning/511561 (visited on 03/01/2022).

[12] The New Ransomware Threat: Triple Extortion, en-US, May 2021. [Online]. Available: https://blog.checkpoint.com/2021/05/12/the-new-ransomware-threat-triple-extortion/ (visited on 03/02/2022).

[13] C. Stupp, "Energy Tech Firm Hit in Ransomware Attack," en-US, Wall Street Journal, May 2021, ISSN: 0099-9660. [Online]. Available: https://www.wsj.com/articles/energy-tech-firm-hit-in-ransomware-attack-11620764034 (visited on 05/14/2022).

[14] I. Thomas, The State of Ransomware Attacks: Energy Industry | Blog, en, Aug. 2021. [Online]. Available: https://ironscales.com/blog/ransomware-in-energy-industry/ (visited on 05/14/2022).

[15] Stortinget utsatt for IT-angrep, no, Pressemelding, Mar. 2021. [Online]. Available: https://www.stortinget.no/no/Hva-skjer-pa-Stortinget/Nyhetsarkiv/Pressemeldingsarkiv/2020-2021/stortinget-utsatt-for-it-angrep/ (visited on 02/22/2022).

[16] Human-Operated Ransomware, en-US. [Online]. Available: https://www.checkpoint.com/cyber-hub/threat-prevention/ransomware/human-operated-ransomware/ (visited on 05/20/2022).

[17] R. Ploszek, P. Švec, and P. Debnár, "Analysis of encryption schemes in modern ransomware," Rad Hrvatske akademije znanosti i umjetnosti Matematičke znanosti, vol. 25(60), pp. 1–13, 2021, ISSN: 1845-4100. DOI: 10.21857/mnlqgc58gy. [Online]. Available: http://dizbi.hazu.hr/a/?pr=i&id=2342255 (visited on 05/12/2022).

[18] M. J. S. May 29 and 2020, Top Ransomware Attack Vectors: RDP, Drive-By, Phishing, en. [Online]. Available: https://www.bankinfosecurity.com/top-ransomware-attack-vectors-rdp-drive-by-phishing-a-14353 (visited on 02/28/2022).

[19] Remote Desktop Protocol (RDP) attack analysis, en-US. [Online]. Available: https://www.darktrace.com/en/blog/remote-desktop-protocol-rdp-attack-analysis (visited on 03/01/2022).

[20] K. Morris, Infrastructure as code: dynamic systems for the cloud age, eng, Second edition. Beijing Boston Farnham Sebastopol Tokyo: O'Reilly, 2020, ISBN: 978-1-09-811467-1.

[21] Types of Backup: Full, Differential & Incremental Backup, en-US, Apr. 2021. [Online]. Available: https://parablu.com/demystifying-data-backups-types-of-backups/ (visited on 05/20/2022).

[22] Types of Backup: Full, Differential, and Incremental, en-US, Mar. 2020. [Online]. Available: https://spanning.com/blog/types-of-backup-understanding-full-differential-incremental-backup/ (visited on 05/20/2022).

[23] Incremental backups on Microsoft Azure Backup: Save on long term storage | Azure-blogger og -oppdateringer | Microsoft Azure, nb. [Online]. Available: https://azure.microsoft.com/nb-no/blog/microsoft-azure-backup-save-on-long-term-storage/ (visited on 04/19/2022).

[24] v-amallick, What is Azure Backup? - Azure Backup, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/backup/backup-overview (visited on 04/17/2022).

[25] Generally available: Azure PostgreSQL backup with long term retention | Azure updates | Microsoft Azure, en. [Online]. Available: https://azure.microsoft.com/en-gb/updates/azure-postgresql-backup-with-long-term-retention-generally-available/ (visited on 04/17/2022).

[26] TerryLanfear, Azure backup and restore plan to protect against ransomware, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/security/fundamentals/backup-plan-to-protect-against-ransomware (visited on 02/22/2022).

[27] v-amallick, FAQ - Protect backups from Ransomware with Azure Backup - Azure Backup, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/backup/protect-backups-from-ransomware-faq (visited on 05/15/2022).

[28] v-amallick, Manage Backups with Azure role-based access control - Azure Backup, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/backup/backup-rbac-rs-vault (visited on 02/23/2022).

[29] v-amallick, Architecture Overview - Azure Backup, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/backup/backup-architecture (visited on 04/17/2022).

[30] v-amallick, Overview of security features - Azure Backup, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/backup/security-overview (visited on 02/22/2022).

[31] A. Somendra, Upgrade classic Backup and Site Recovery vaults to ARM Recovery Services vaults, en, May 2017. [Online]. Available: https://azure.microsoft.com/en-gb/blog/upgrade-classic-backup-and-siterecovery-vault-to-arm-recovery-services-vault/ (visited on 05/12/2022).

[32] Overview of Backup vaults - Azure Backup | Microsoft Docs. [Online]. Available: https://docs.microsoft.com/en-us/azure/backup/backup-vault-overview (visited on 05/12/2022).

[33] tamram, Data redundancy - Azure Storage, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/storage/common/storage-redundancy (visited on 04/02/2022).

[34] R. Modi, Azure for architects: implementing cloud design, DevOps, containers, IoT, and serverless solutions on your public cloud, eng, Second edition. Birmingham Mumbai: Packt, 2019, ISBN: 978-1-78961-450-3.

[35] V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, "Guide to Attribute Based Access Control (ABAC) Definition and Considerations," en, National Institute of Standards and Technology, Tech. Rep. NIST SP 800-162, Jan. 2014, NIST SP 800–162. DOI: 10.6028/NIST.SP.800-162. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-162.pdf (visited on 05/20/2022).

[36] Protect Critical Backup Operations with Multi-User Authorization for Azure Backup - CHARBEL NEMNOM - MVP | MCT | CCSP - Cloud & Cyber-Security, en-us, Section: Microsoft Azure, Oct. 2021. [Online]. Available: https://charbelnemnom.com/multi-user-authorization-for-azure-backup/ (visited on 03/21/2022).

[37] v-amallick, Soft delete for Azure Backup - Azure Backup, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/backup/backup-azure-security-feature-cloud (visited on 02/22/2022).

[38] v-amallick, Monitoring and reporting solutions for Azure Backup - Azure Backup, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/backup/monitoring-and-alerts-overview (visited on 05/12/2022).

[39] nishanil, Infrastructure as code, en-us. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/infrastructure-as-code (visited on 05/19/2022).

[40] tamram, Introduction to Blob (object) storage - Azure Storage, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction (visited on 05/19/2022).

[41] tamram, Soft delete for blobs - Azure Storage, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/storage/blobs/soft-delete-blob-overview (visited on 05/20/2022).

[42] What Is ClickHouse? | ClickHouse Docs, en. [Online]. Available: https://clickhouse.com/docs/en/intro/ (visited on 05/19/2022).

[43] Data Backup | ClickHouse Documentation, en. [Online]. Available: https://clickhouse.com/docs/en/operations/backup/ (visited on 03/22/2022).

[44] Azure Samples, en. [Online]. Available: https://github.com/Azure-Samples (visited on 02/10/2022).

[45] v-amallick, Overview of Recovery Services vaults - Azure Backup, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/backup/backup-azure-recovery-services-vault-overview (visited on 05/12/2022).

[46] Key Vault | Microsoft Azure, en. [Online]. Available: https://azure.microsoft.com/en-us/services/key-vault/ (visited on 05/19/2022).

[47] B. D. Ramel and 11/16/2021, Anatomy of a Ransomware Attack: Immutable Cloud Blob to the Rescue! -, en-US. [Online]. Available: https://virtualizationreview.com/articles/2021/11/16/ransomware-attack.aspx (visited on 03/09/2022).

[48] Cybersecurity and Infrastructure Security Agency and Cybersecurity Division, "CISA Zero Trust Maturity Model," en, p. 19, 2021.

[49] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero Trust Architecture," en, National Institute of Standards and Technology, Tech. Rep., Aug. 2020. DOI: 10.6028/NIST.SP.800-207. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf (visited on 03/09/2022).

[50] rishabv90, B-series burstable - Azure Virtual Machines, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/virtual-machines/sizes-b-series-burstable (visited on 05/19/2022).

[51] andysports8, Dv4 and Dsv4-series - Azure Virtual Machines, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/virtual-machines/dv4-dsv4-series (visited on 05/11/2022).

[52] Usage Recommendations | ClickHouse Docs, en. [Online]. Available: https://clickhouse.com/docs/en/operations/tips/ (visited on 04/27/2022).

[53] Requirements | ClickHouse Docs, en. [Online]. Available: https://clickhouse.com/docs/en/operations/requirements/ (visited on 05/11/2022).

[54] v-amallick, Azure Backup support matrix - Azure Backup, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/backup/backup-support-matrix (visited on 05/18/2022).

[55] A. Akulov, Clickhouse-backup, original-date: 2018-09-26T15:00:57Z, May 2022. [Online]. Available: https://github.com/AlexAkulov/clickhouse-backup/blob/0a19cd2b354b3b1fc94af4992d8c8499f5c1d9c9/Examples.md (visited on 05/15/2022).

[56] Pricing - Cloud Backup | Microsoft Azure, en. [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/backup/ (visited on 05/15/2022).

[57] Azure Storage Blobs Pricing | Microsoft Azure, en. [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/storage/blobs/ (visited on 05/15/2022).

[58] Installation | ClickHouse Docs, en. [Online]. Available: https://clickhouse.com/docs/en/getting-started/install/ (visited on 05/02/2022).

[59] Cell Towers | ClickHouse Docs, en. [Online]. Available: https://clickhouse.com/docs/en/getting-started/example-datasets/cell-towers/ (visited on 05/02/2022).

[60] A. Akulov, Clickhouse-backup, original-date: 2018-09-26T15:00:57Z, Apr. 2022. [Online]. Available: https://github.com/AlexAkulov/clickhouse-backup (visited on 05/02/2022).

[61] stevenmatthew, Quickstart: Upload, download, and list blobs - Azure CLI - Azure Storage, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/storage/blobs/storage-quickstart-blobs-cli (visited on 05/02/2022).

[62] v-amallick, Quickstart - Back up a VM with Azure CLI - Azure Backup, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/backup/quick-backup-vm-cli (visited on 05/02/2022).

[63] Parts | ClickHouse Docs, en. [Online]. Available: https://clickhouse.com/docs/en/operations/system-tables/parts/ (visited on 04/28/2022).

[64] v-amallick, Back up Azure Database for PostgreSQL - Azure Backup, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/backup/backup-azure-database-postgresql (visited on 01/20/2022).

[65] v-amallick, Script Sample - Delete a Recovery Services vault - Azure Backup, en-us. [Online]. Available: https://docs.microsoft.com/en-us/azure/backup/scripts/delete-recovery-services-vault (visited on 05/08/2022).

Appendix A

Experiment Data

This section contains details about how experiments related to the analysis were performed, as well as results.

A.1 Setup of test environment for ClickHouse experiments

This section contains documentation for how our ClickHouse test environment was set up in accordance with the specifications listed in the Method chapter (see 3.4.2). This test environment was used for tests that were not performance sensitive.

Most commands were run in the Azure Cloud Shell. Instead of using the bash environment for the Azure CLI, we used the PowerShell environment. Backticks (‘) are therefore used instead of backslashes (\) to escape newlines.

A.1.1 Declare variables

The following variables were declared to make scripts more reusable:

```
$AzCloudUser = "torstein"        # Name of Azure user used for CLI
 ↪  commands
$RGName = "testRG"               # Name of resource group
$CHName = "clickhouseVM"         # Name of VM running ClickHouse
$SSHKey = "mySSHKey"             # Name of SSH key used to connect
 ↪  to VM
$SSHPath = "/home/$AzCloudUser/.ssh/$SSHKey.pub"  # Path of SSH
 ↪  keys in Azure Cloud Shell storage
$SAName = "chbksa"               # Name of storage account used by
 ↪  clickhouse-backup
$ContainerName = "chbkcontainer" # Name of container storing
 ↪  clickhouse-backup data
$StagingSAName = "stagingchsa"   # Name of storage account used for
 ↪  Azure Backup staging
$SASExpDate = "2022-05-05"       # Expiry date of SAS token used by
 ↪  clickhouse-backup
$location = "eastus"             # Location of Azure resources
$RSVName = "myRSV"               # Name of Recovery Services Vault
$subscription = "4b48eb85-91f3-4902-b74b-e84641fb6785"  #
 ↪  Subscription ID
$PolicyName = "DefaultPolicy"    # Policy to be used by Azure Backup
```

These variables were loaded the other (non-performance) experiments as well.

A.1.2 Generate SSH keys

In order to connect to the VM via SSH, SSH keys needed to be generated.

```
az sshkey create --name $SSHKey --resource-group $RGName
```

Output:

```
No public key is provided. A key pair is being generated for you.
Private key is saved to "/home/torstein/.ssh/1650965634_7008111".
Public key is saved to "/home/torstein/.ssh/1650965634_7008111.pub".
```

```
{
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
   ↪  ceGroups/TESTRG/providers/Microsoft.Compute/sshPublicKeys/myS⌋
   ↪  SHKey",
  "location": "eastus",
  "name": "mySSHKey",
  "publicKey": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQC4N1FPh8lylHa⌋
   ↪  F1Y2BpTCNNYkO\r\nJ0aaMLSklwBzdA4MrLY0vqnIhJY1QzB7NC3tOxOAqEZV⌋
   ↪  QjqiOiFQRZu0ifV91odi\r\njwuEOWCWia8Wix0AKpxPlFIcUbWE4auf6KOSA⌋
   ↪  AuoTCuXp3H29H+tGtuy/l1ZjBJb\r\nazxFz32j9uwEjVLtkYiYtSPxtnhuTn⌋
   ↪  N6l5wkI36mxni6dQsTKQNTck4bmkU+BmSI\r\nQ4k5YXfhxV0UTqYL5WeVzVD⌋
   ↪  TJnBrDFu8fClio/kcbhCn+w6B2yWfNFuvLmOlpIA6\r\nVZm4MaWpD9mp25J9⌋
   ↪  YDcJ/Y5MewuNW7QqVNQDKfTrWXKz+blkKQ9DVzXvF63myNtQ\r\niMJdzLTvA⌋
   ↪  bgTL0zXPw111r5x8KKHW0nCehWtKvHk2xh2sOcdSVSDu4uZN6eacAPC\r\n4I⌋
   ↪  i3rJ3QgjL1IxRTITBJ/oZAFxxFZCtPbPYF9pD2iURzibAvLkp5BhLwk954So8⌋
   ↪  G\r\nMzRvVUnM7gtaVNL4UxCYHMbvibwkXaNtu2ZWjGU=
   ↪  generated-by-azure\r\n",
  "resourceGroup": "TESTRG",
  "tags": null,
  "type": null
}
```

The keys were then renamed:

```
cd ~/.ssh
mv ./1650965634_7008111 $SSHKey
mv ./1650965634_7008111.pub ($SSHKey + ".pub")
```

These files were then copied to a local machine in order to be able to access virtual machines using SSH.

A.1.3 Set up a resource group

```
az group create --name $RGName --location $location
```

Output:

```
{
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
   ↪  ceGroups/testRG",
  "location": "eastus",
  "managedBy": null,
  "name": "testRG",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
```

A.1.4 Set up a VM

Create an Ubuntu VM:

```
az vm create `
    --resource-group $RGName `
    --name $CHName `
    --image Canonical:UbuntuServer:16.04-LTS:16.04.202109280 `
    --admin-username azureuser `
    --size Standard_B1s `
    --ssh-key-values $SSHPath `
    --public-ip-sku Standard
```

Output:

```
{
  "fqdns": "",
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
   ↪  ceGroups/testRG/providers/Microsoft.Compute/virtualMachines/c⌋
   ↪  lickhouseVM",
  "location": "eastus",
  "macAddress": "00-0D-3A-4F-38-61",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.4",
  "publicIpAddress": "20.127.83.11",
  "resourceGroup": "testRG",
  "zones": ""
}
```

A.1.5 Install ClickHouse

Commands to install ClickHouse were copied from the installation guide in the ClickHouse documentation [58].

```
sudo apt-get install -y apt-transport-https ca-certificates dirmngr
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
 ↪  8919F6BD2B48D754
```

```
echo "deb https://packages.clickhouse.com/deb stable main" | sudo
↪  tee \
    /etc/apt/sources.list.d/clickhouse.list
sudo apt-get update

sudo apt-get install -y clickhouse-server clickhouse-client

sudo service clickhouse-server start
```

The commands were run as `azureuser` on the ClickHouse VM. The ClickHouse default user password was left empty.

Upon running `clickhouse-client`, which is used to interact with the database, the following warnings were printed:

```
 * Linux transparent hugepage are set to "always".
 * Linux threads max count is too low.
 * Available memory at server startup is too low (2GiB).
 * Maximum number of threads is lower than 30000. There could be
  ↪  problems with handling a lot of simultaneous queries.
```

This is caused by the VM having quite weak hardware. The dataset we used was small enough, and the queries we ran were simple enough, for this now to be an issue.

A.1.6 Load test data

A simple set of test data was retrieved from the ClickHouse website [59]. The data set contains cell tower data and is around 700MB. Since this data set is not intended for use in performance sensitive tests, we determined that it was sufficiently large.

The data was loaded by following the instructions in the documentation.

Load test data:

```
# Download dataset
wget https://datasets.clickhouse.com/cell_towers.csv.xz

# Decompress dataset
xz -d cell_towers.csv.xz

# Load data into clickhouse
clickhouse-client --query \
"CREATE TABLE cell_towers_two
(
    radio Enum8('' = 0, 'CDMA' = 1, 'GSM' = 2, 'LTE' = 3, 'NR' = 4,
↪  'UMTS' = 5),
    mcc UInt16,
    net UInt16,
    area UInt16,
    cell UInt64,
    unit Int16,
    lon Float64,
    lat Float64,
    range UInt32,
    samples UInt32,
    changeable UInt8,
    created DateTime,
    updated DateTime,
    averageSignal UInt8
)
ENGINE = MergeTree ORDER BY (radio, mcc, net, created);"

# Load test data into database
clickhouse-client --query "INSERT INTO cell_towers FORMAT
 ↪  CSVWithNames" < cell_towers.csv
```

A.1.7 Run test queries

The following test queries were run from `clickhouse-client` to verify that the data was loaded correctly. The results were compared with the results listed in the documentation [59].

The outputs of queries are shown in comments under the SQL statements. The `TabSeparated` format is used because the default output format contains Unicode characters that are difficult to display when using LaTeX.

Test query 1:

```
SELECT
    radio,
    count() AS c
FROM cell_towers
GROUP BY radio
ORDER BY c DESC
FORMAT TabSeparated

-- Query id: 067359b9-2a3b-4683-9e85-74a50cd93719
--
-- UMTS    20686487
-- LTE     12101148
-- GSM     9931312
-- CDMA    556344
-- NR      867
```

```
--
-- 10 rows in set. Elapsed: 0.205 sec. Processed 43.28 million
↪  rows, 86.55 MB (211.56 million rows/s., 423.12 MB/s.)
```

Test query 2:

```
SELECT
    mcc,
    count()
FROM cell_towers
GROUP BY mcc
ORDER BY count() DESC
LIMIT 10
FORMAT TabSeparated

-- Query id: 6f1c8ef2-f866-48d5-b521-abde9d34a65c
--
-- 310     5024650
-- 262     2622423
-- 250     1953176
-- 208     1891187
-- 724     1836150
-- 404     1729151
-- 234     1618924
-- 510     1353998
-- 440     1343355
-- 311     1332798
--
-- 10 rows in set. Elapsed: 0.397 sec. Processed 43.28 million
↪  rows, 86.55 MB (108.92 million rows/s., 217.84 MB/s.)
```

The results of both queries matched the results in the documentation.

A.1.8 Install `clickhouse-backup`

The most recent (as of 2022-05-02) `clickhouse-backup` binary was downloaded from `clickhouse-backup`'s GitHub page [60].

```
# Download archive containing binary
wget https://github.com/AlexAkulov/clickhouse-backup/releases/downl⌋
↪  oad/v1.3.2/clickhouse-backup-linux-amd64.tar.gz

# Decompress archive
tar -zxvf clickhouse-backup-linux-amd64.tar.gz

# Move binary to home directory
```

```
mv build/linux/amd64/clickhouse-backup ~

# Cleanup
rmdir -p build/linux/amd64
rm clickhouse-backup-linux-amd64.tar.gz
```

A.1.9 Set up Azure Blob storage for use with `clickhouse-backup`

`clickhouse-backup` can store remote backups in Azure Blob Storage. This requires an Azure Storage Account and a Storage Container, which we created by following the instructions in the Azure documentation [61]

Create a storage account

Create storage account:

```
az storage account create `
    --name $SAName `
    --resource-group $RGName `
    --location eastus `
    --sku Standard_LRS `
    --encryption-services blob
```

Output:

```
{
  "accessTier": "Hot",
  "allowBlobPublicAccess": true,
  "allowCrossTenantReplication": null,
  "allowSharedKeyAccess": null,
  "allowedCopyScope": null,
  "azureFilesIdentityBasedAuthentication": null,
  "blobRestoreStatus": null,
  "creationTime": "2022-04-30T10:14:26.700057+00:00",
  "customDomain": null,
  "defaultToOAuthAuthentication": null,
  "dnsEndpointType": null,
  "enableHttpsTrafficOnly": true,
  "enableNfsV3": null,
  "encryption": {
    "encryptionIdentity": null,
    "keySource": "Microsoft.Storage",
    "keyVaultProperties": null,
    "requireInfrastructureEncryption": null,
    "services": {
      "blob": {
        "enabled": true,
        "keyType": "Account",
        "lastEnabledTime": "2022-04-30T10:14:26.825032+00:00"
      },
      "file": {
        "enabled": true,
        "keyType": "Account",
        "lastEnabledTime": "2022-04-30T10:14:26.825032+00:00"
      },
      "queue": null,
      "table": null
    }
  },
  "extendedLocation": null,
  "failoverInProgress": null,
  "geoReplicationStats": null,
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
  ↪  ceGroups/testRG/providers/Microsoft.Storage/storageAccounts/c⌋
  ↪  hbksa",
  "identity": null,
  "immutableStorageWithVersioning": null,
  "isHnsEnabled": null,
  "isLocalUserEnabled": null,
  "isSftpEnabled": null,
  "keyCreationTime": {
    "key1": "2022-04-30T10:14:26.825032+00:00",
    "key2": "2022-04-30T10:14:26.825032+00:00"
  },
  "keyPolicy": null,
  "kind": "StorageV2",
  "largeFileSharesState": null,
  "lastGeoFailoverTime": null,
  "location": "eastus",
  "minimumTlsVersion": "TLS1_0",
  "name": "chbksa",
  "networkRuleSet": {
    "bypass": "AzureServices",
    "defaultAction": "Allow",
    "ipRules": [],
    "resourceAccessRules": null,
    "virtualNetworkRules": []
  },
  "primaryEndpoints": {
    "blob": "https://chbksa.blob.core.windows.net/",
    "dfs": "https://chbksa.dfs.core.windows.net/",
    "file": "https://chbksa.file.core.windows.net/",
    "internetEndpoints": null,
    "microsoftEndpoints": null,
    "queue": "https://chbksa.queue.core.windows.net/",
    "table": "https://chbksa.table.core.windows.net/",
    "web": "https://chbksa.z13.web.core.windows.net/"
  },
  "primaryLocation": "eastus",
  "privateEndpointConnections": [],
  "provisioningState": "Succeeded",
  "publicNetworkAccess": null,
  "resourceGroup": "testRG",
  "routingPreference": null,
  "sasPolicy": null,
  "secondaryEndpoints": null,
  "secondaryLocation": null,
  "sku": {
    "name": "Standard_LRS",
    "tier": "Standard"
  },
  "statusOfPrimary": "available",
  "statusOfSecondary": null,
  "storageAccountSkuConversionStatus": null,
  "tags": {},
  "type": "Microsoft.Storage/storageAccounts"
}
```

Create a storage container

Create storage container:

```
az storage container create `
    --account-name $SAName `
    --name $ContainerName `
    --auth-mode login
```

Output:

```
{
  "created": true
}
```

A.1.10 Enable soft delete for Blob container

Enable soft delete for Blob container:

```
az storage account blob-service-properties update `
    --enable-container-delete-retention true `
    --container-delete-retention-days 7 `
    --account-name $SAName `
    --resource-group $RGName
```

Output:

```
{
  "automaticSnapshotPolicyEnabled": null,
  "changeFeed": null,
  "containerDeleteRetentionPolicy": {
    "allowPermanentDelete": null,
    "days": 7,
    "enabled": true
  },
  "cors": {
    "corsRules": []
  },
  "defaultServiceVersion": null,
  "deleteRetentionPolicy": {
    "allowPermanentDelete": false,
    "days": null,
    "enabled": false
  },
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌟
  ↪  ceGroups/testRG/providers/Microsoft.Storage/storageAccounts/c⌟
  ↪  hbksa/blobServices/default",
  "isVersioningEnabled": null,
  "lastAccessTimeTrackingPolicy": null,
  "name": "default",
  "resourceGroup": "testRG",
  "restorePolicy": null,
  "sku": null,
  "type": "Microsoft.Storage/storageAccounts/blobServices"
}
```

A.1.11 Configure `clickhouse-backup` to use Blob storage

Generate `clickkhouse-backup` configuration file

A configuration file for `clickhouse-backup` was generated with `sudo ./clickhouse-backup default-config > config.yaml`

```
general:
  remote_storage: none
  max_file_size: 0
  disable_progress_bar: true
  backups_to_keep_local: 0
  backups_to_keep_remote: 0
  log_level: info
  allow_empty_backups: false
  download_concurrency: 1
  upload_concurrency: 1
  restore_schema_on_cluster: ""
  upload_by_part: true
  download_by_part: true
clickhouse:
  username: default
  password: ""
  host: localhost
  port: 9000
  disk_mapping: {}
  skip_tables:
  - system.*
  - INFORMATION_SCHEMA.*
  - information_schema.*
  timeout: 5m
  freeze_by_part: false
  secure: false
  skip_verify: false
  sync_replicated_tables: false
  log_sql_queries: true
  config_dir: /etc/clickhouse-server/
  restart_command: systemctl restart clickhouse-server
  ignore_not_exists_error_during_freeze: true
  tls_key: ""
  tls_cert: ""
  tls_ca: ""
  debug: false
s3:
  access_key: ""
  secret_key: ""
  bucket: ""
  endpoint: ""
  region: us-east-1
  acl: private
  assume_role_arn: ""
  force_path_style: false
  path: ""
  disable_ssl: false
  compression_level: 1
  compression_format: tar
  sse: ""
  disable_cert_verification: false
  storage_class: STANDARD
  concurrency: 1
  part_size: 0
  max_parts_count: 10000
  debug: false
gcs:
  credentials_file: ""
  credentials_json: ""
  bucket: ""
  path: ""
  compression_level: 1
  compression_format: tar
  debug: false
  endpoint: ""
cos:
  url: ""
  timeout: 2m
  secret_id: ""
  secret_key: ""
  path: ""
  compression_format: tar
  compression_level: 1
  debug: false
api:
  listen: localhost:7171
  enable_metrics: true
  enable_pprof: false
  username: ""
  password: ""
  secure: false
  certificate_file: ""
  private_key_file: ""
  create_integration_tables: false
  allow_parallel: false
ftp:
  address: ""
  timeout: 2m
  username: ""
  password: ""
  tls: false
  path: ""
  compression_format: tar
  compression_level: 1
  concurrency: 1
  debug: false
sftp:
  address: ""
  port: 22
  username: ""
  password: ""
  key: ""
  path: ""
  compression_format: tar
  compression_level: 1
  concurrency: 1
  debug: false
azblob:
  endpoint_suffix: core.windows.net
  account_name: ""
  account_key: ""
  sas: ""
  use_managed_identity: false
  container: ""
  path: ""
  compression_level: 1
  compression_format: tar
  sse_key: ""
  buffer_size: 0
  buffer_count: 3
  max_parts_count: 0
```

Get necessary config details

In order to configure `clickhouse-backup` for use with Azure Storage Blobs, we need an access key and a SAS token.

1. Get access key for storage account

   ```
   az storage account keys list `
     --resource-group $RGName `
     --account-name $SAName
   ```

   Output:

   ```
   [
     {
   ```

```
      "creationTime": "2022-04-30T10:14:26.825032+00:00",
      "keyName": "key1",
      "permissions": "FULL",
      "value": "NdbW07WlHBf5zcpMXundwkP88Ie2SO1Ad+84VD8moaUg1ihI
       ↪  eRR7cEdy4FXIgHRvIQwPIMc7eD2q+ASt6EqxWg=="
    },
    {
      "creationTime": "2022-04-30T10:14:26.825032+00:00",
      "keyName": "key2",
      "permissions": "FULL",
      "value": "vs33au0M52gdtKdhqeOiC0vBGRRqO1qUtvQk0Eg2c4TVIFHE
       ↪  TmWR7taS8w2ZU5mEAPzYS4ySWBXY+AStcBusiQ=="
    }
  ]
```

2. Get SAS token for container

```
az storage container generate-sas `
    --account-name $SAName `
    --name $ContainerName `
    --permissions acdlrw `
    --expiry $SASExpDate `
    --auth-mode login `
    --as-user
```

Output:

```
"se=2022-05-05&sp=racwdl&sv=2021-04-10&sr=c&skoid=d404139d-e15
 ↪  6-421c-9450-19e9734a8141&sktid=09a10672-822f-4467-a5ba-5bb
 ↪  375967c05&skt=2022-04-30T10%3A32%3A05Z&ske=2022-05-05T00%3
 ↪  A00%3A00Z&sks=b&skv=2021-04-10&sig=CoeOsK63ZSnuqUqn%2Bv2Zg
 ↪  E9/sELwp/Hinxj6abW9qfo%3D"
```

Modify configuration file to make it work with Blob storage

Some `general` settings were modified:

- `remote_storage` was set to `azblob` (from none).
- `max_parts_count` was set to 1 (from 0).
- `disable_progress_bar` was set to `false` (from `true`).

The necessary `azblob` details were also filled in.
Modified version of `config.yaml`:

```
general:
  remote_storage: azblob
  max_file_size: 0
  disable_progress_bar: false
  backups_to_keep_local: 0
  backups_to_keep_remote: 0
  log_level: info
  allow_empty_backups: false
  download_concurrency: 1
  upload_concurrency: 1
  restore_schema_on_cluster: ""
  upload_by_part: true
  download_by_part: true
clickhouse:
  username: default
  password: ""
  host: localhost
  port: 9000
  disk_mapping: {}
  skip_tables:
  - system.*
  - INFORMATION_SCHEMA.*
  - information_schema.*
  timeout: 5m
  freeze_by_part: false
  secure: false
  skip_verify: false
  sync_replicated_tables: false
  log_sql_queries: true
  config_dir: /etc/clickhouse-server/
  restart_command: systemctl restart clickhouse-server
  ignore_not_exists_error_during_freeze: true
  tls_key: ""
  tls_cert: ""
  tls_ca: ""
  debug: false
s3:
  access_key: ""
  secret_key: ""
  bucket: ""
  endpoint: ""
  region: us-east-1
  acl: private
  assume_role_arn: ""
  force_path_style: false
  path: ""
  disable_ssl: false
  compression_level: 1
  compression_format: tar
  sse: ""
  disable_cert_verification: false
  storage_class: STANDARD
  concurrency: 1
  part_size: 0
  max_parts_count: 10000
  debug: false
gcs:
  credentials_file: ""
  credentials_json: ""
  bucket: ""
  path: ""
  compression_level: 1
  compression_format: tar
  debug: false
  endpoint: ""
cos:
  url: ""
  timeout: 2m
  secret_id: ""
  secret_key: ""
  path: ""
  compression_format: tar
  compression_level: 1
  debug: false
api:
  listen: localhost:7171
  enable_metrics: true
  enable_pprof: false
  username: ""
  password: ""
  secure: false
  certificate_file: ""
  private_key_file: ""
  create_integration_tables: false
  allow_parallel: false
ftp:
  address: ""
  timeout: 2m
  username: ""
  password: ""
  tls: false
  path: ""
  compression_format: tar
  compression_level: 1
  concurrency: 1
  debug: false
sftp:
  address: ""
  port: 22
  username: ""
  password: ""
  key: ""
  path: ""
  compression_format: tar
  compression_level: 1
  concurrency: 1
  debug: false
azblob:
  endpoint_suffix: core.windows.net
  account_name: "chbksa"
  account_key: "NdbW07WlHBf5zcpMXundwkP88Ie2SO1Ad+84VD8moaUg1ihIeRR⌋
   ↪  7cEdy4FXIgHRvIQwPIMc7eD2q+ASt6EqxWg=="
  sas: "?sv=2020-08-04&ss=bfqt&srt=c&sp=rwdlacupitfx&se=2022-05-02T⌋
   ↪  14:36:42Z&st=2022-05-02T06:36:42Z&spr=https&sig=ONxr9dN3ayvlq⌋
   ↪  1fwk4b516u%2F9X1ZjVNamV88yeeYksU%3D"
  use_managed_identity: false
  container: "chbkcontainer"
  path: "https://chbksa.blob.core.windows.net/chbkcontainer"
  compression_level: 1
  compression_format: tar
  sse_key: ""
  buffer_size: 0
  buffer_count: 3
  max_parts_count: 1
```

Perform local backup

Create a local backup:

```
sudo ./clickhouse-backup create

# 2022/05/02 09:34:00.837899  info SELECT name, engine FROM
 ↪  system.databases WHERE name NOT IN ('system',
 ↪  'INFORMATION_SCHEMA', 'information_schema')
# 2022/05/02 09:34:00.844477  info SHOW CREATE DATABASE `default`
# 2022/05/02 09:34:00.849781  info SELECT count() FROM
 ↪  system.settings WHERE name =
 ↪  'show_table_uuid_in_table_create_query_if_not_nil'
```

```
# 2022/05/02 09:34:00.861793  info SELECT name FROM
 ↪  system.databases WHERE engine IN ('MySQL','PostgreSQL')
# 2022/05/02 09:34:00.867246  info
#                 SELECT
#                         countIf(name='data_path')
 ↪  is_data_path_present,
#                         countIf(name='data_paths')
 ↪  is_data_paths_present,
#                         countIf(name='uuid') is_uuid_present,
#                         countIf(name='create_table_query')
 ↪  is_create_table_query_present,
#                         countIf(name='total_bytes')
 ↪  is_total_bytes_present
#                 FROM system.columns WHERE database='system' AND
 ↪  table='tables'
#
# 2022/05/02 09:34:00.881984  info SELECT database, name, engine ,
 ↪  data_paths , uuid , create_table_query , coalesce(total_bytes,
 ↪  0) AS total_bytes   FROM system.tables WHERE is_temporary = 0
 ↪  SETTINGS show_table_uuid_in_table_create_query_if_not_nil=1
# 2022/05/02 09:34:00.911828  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/02 09:34:00.920436  info SELECT * FROM system.disks;
# 2022/05/02 09:34:00.929018  info ALTER TABLE
 ↪  `default`.`cell_towers` FREEZE WITH NAME
 ↪  '671596e8b89e4fc4bc4b69c4992011d4';
# 2022/05/02 09:34:01.002079  info done
 ↪  backup=2022-05-02T09-34-00 operation=create
 ↪  table=default.cell_towers
# 2022/05/02 09:34:01.002456  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_DESCRIBE'
# 2022/05/02 09:34:01.008365  info done
 ↪  backup=2022-05-02T09-34-00 duration=173ms operation=create
```

List local backups:

```
sudo ./clickhouse-backup list

# 2022/05/02 09:36:01.867327  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/02 09:36:01.871571  info SELECT * FROM system.disks;
# 2022-05-02T09-34-00   1.07GiB   02/05/2022 09:34:01   local
```

Perform remote backup

Create a remote backup:

```
sudo ./clickhouse-backup create_remote --config config.yaml

# 2022/05/02 09:48:18.613252  info SELECT name, engine FROM
 ↪  system.databases WHERE name NOT IN ('system',
 ↪  'INFORMATION_SCHEMA', 'information_schema')
# 2022/05/02 09:48:18.616953  info SHOW CREATE DATABASE `default`
# 2022/05/02 09:48:18.619861  info SELECT count() FROM
 ↪  system.settings WHERE name =
 ↪  'show_table_uuid_in_table_create_query_if_not_nil'
# 2022/05/02 09:48:18.622844  info SELECT name FROM
 ↪  system.databases WHERE engine IN ('MySQL','PostgreSQL')
# 2022/05/02 09:48:18.625686  info
#                 SELECT
#                         countIf(name='data_path')
 ↪  is_data_path_present,
#                         countIf(name='data_paths')
 ↪  is_data_paths_present,
#                         countIf(name='uuid') is_uuid_present,
#                         countIf(name='create_table_query')
 ↪  is_create_table_query_present,
#                         countIf(name='total_bytes')
 ↪  is_total_bytes_present
#                 FROM system.columns WHERE database='system' AND
 ↪  table='tables'
#
# 2022/05/02 09:48:18.633230  info SELECT database, name, engine ,
 ↪  data_paths , uuid , create_table_query , coalesce(total_bytes,
 ↪  0) AS total_bytes   FROM system.tables WHERE is_temporary = 0
 ↪  SETTINGS show_table_uuid_in_table_create_query_if_not_nil=1
# 2022/05/02 09:48:18.642818  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/02 09:48:18.648335  info SELECT * FROM system.disks;
# 2022/05/02 09:48:18.652902  info ALTER TABLE
 ↪  `default`.`cell_towers` FREEZE WITH NAME
 ↪  '005688e38c3e41dfa2da8d275e7d3c2b';
# 2022/05/02 09:48:18.736020  info done
 ↪  backup=2022-05-02T09-48-18 operation=create
 ↪  table=default.cell_towers
# 2022/05/02 09:48:18.736333  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_DESCRIBE'
# 2022/05/02 09:48:18.741265  info done
 ↪  backup=2022-05-02T09-48-18 duration=132ms operation=create
# 2022/05/02 09:48:18.747950  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
```

```
# 2022/05/02 09:48:18.752206  info SELECT * FROM system.disks;
# 2022/05/02 09:48:18.758250  info SELECT max(toInt64(bytes_on_disk
↪  * 1.02)) AS max_file_size FROM system.parts
# 2022/05/02 09:48:39.787095  info done
↪  backup=2022-05-02T09-48-18 duration=20.971s operation=upload
↪  size=1.07GiB table=default.cell_towers
# 2022/05/02 09:48:39.804194  info done
↪  backup=2022-05-02T09-48-18 duration=21.063s operation=upload
↪  size=1.07GiB
```

List remote backups:

```
sudo ./clickhouse-backup list remote --config config.yaml

# 2022/05/02 09:53:50.813272  info SELECT max(toInt64(bytes_on_disk
↪  * 1.02)) AS max_file_size FROM system.parts
# 2022-05-02T09-48-18   1.07GiB   02/05/2022 09:48:39   remote
↪  tar
```

The backup was also visible in the Azure Portal.

A.1.12 Set up Azure Backup

Azure Backup was set up to back up the ClickHouse VM. Commands are mostly based on instructions from the Azure Documentation [62]

Create a Recovery Services vault

```
az backup vault create --location $location --name $RSVName
↪  --resource-group $RGName
```

Output:

```
{
  "etag": "W/\"datetime'2022-05-02T12%3A34%3A13.452294Z'\"",
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
  ↪  ceGroups/testRG/providers/Microsoft.RecoveryServices/vaults/m⌋
  ↪  yRSV",
  "identity": null,
  "location": "eastus",
  "name": "myRSV",
  "properties": {
    "encryption": null,
    "privateEndpointConnections": null,
    "privateEndpointStateForBackup": "None",
    "privateEndpointStateForSiteRecovery": "None",
    "provisioningState": "Succeeded",
    "upgradeDetails": null
  },
  "resourceGroup": "testRG",
  "sku": {
    "name": "Standard",
    "tier": null
  },
  "systemData": null,
  "tags": null,
  "type": "Microsoft.RecoveryServices/vaults"
}
```

Disable geo-redundancy

To save on costs, we disabled geo-redundant storage.

```
az backup vault backup-properties set --backup-storage-redundancy
↪  LocallyRedundant --name $RSVName --resource-group $RGName
↪  --subscription $subscription
```

Output:

```
{
  "eTag": null,
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
  ↪  ceGroups/testRG/providers/Microsoft.RecoveryServices/vaults/m⌋
  ↪  yRSV/backupstorageconfig/vaultstorageconfig",
  "location": null,
  "name": "vaultstorageconfig",
  "properties": {
    "crossRegionRestoreFlag": false,
    "dedupState": "Disabled",
    "storageModelType": "LocallyRedundant",
    "storageType": "LocallyRedundant",
    "storageTypeState": "Unlocked",
    "xcoolState": "Disabled"
  },
  "resourceGroup": "testRG",
  "tags": null,
  "type": "Microsoft.RecoveryServices/vaults/backupstorageconfig"
}
```

Decide which backup policy to use

We determined that the default policy is sufficient for our experiments. The policy is listed below.

```
az backup policy show -g $RGName -v $RSVName -n DefaultPolicy
```

Output:

```
{
  "eTag": null,
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
   ↪  ceGroups/testRG/providers/Microsoft.RecoveryServices/vaults/m⌋
   ↪  yRSV/backupPolicies/DefaultPolicy",
  "location": null,
  "name": "DefaultPolicy",
  "properties": {
    "backupManagementType": "AzureIaasVM",
    "instantRpDetails": {
      "azureBackupRgNamePrefix": null,
      "azureBackupRgNameSuffix": null
    },
    "instantRpRetentionRangeInDays": 2,
    "policyType": null,
    "protectedItemsCount": 0,
    "resourceGuardOperationRequests": null,
    "retentionPolicy": {
      "dailySchedule": {
        "retentionDuration": {
          "count": 30,
          "durationType": "Days"
        },
        "retentionTimes": [
          "2022-05-02T20:00:00+00:00"
        ]
      },
      "monthlySchedule": null,
      "retentionPolicyType": "LongTermRetentionPolicy",
      "weeklySchedule": null,
      "yearlySchedule": null
    },
    "schedulePolicy": {
      "hourlySchedule": null,
      "schedulePolicyType": "SimpleSchedulePolicy",
      "scheduleRunDays": null,
      "scheduleRunFrequency": "Daily",
      "scheduleRunTimes": [
        "2022-05-02T20:00:00+00:00"
      ],
      "scheduleWeeklyFrequency": 0
    },
    "timeZone": "UTC"
  },
  "resourceGroup": "testRG",
  "tags": null,
  "type": "Microsoft.RecoveryServices/vaults/backupPolicies"
}
```

Enable backup for the VM

```
az backup protection enable-for-vm `
 --resource-group $RGName `
 --vault-name $RSVName `
 --vm $CHName `
 --policy-name $PolicyName
```

Output:

```
{
  "eTag": null,
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
   ↪  ceGroups/testRG/providers/Microsoft.RecoveryServices/vaults/m⌋
   ↪  yRSV/backupJobs/a09a2633-ec57-43c5-b121-b48d19cd936a",
  "location": null,
  "name": "a09a2633-ec57-43c5-b121-b48d19cd936a",
  "properties": {
    "actionsInfo": null,
    "activityId": "62452fde-ca14-11ec-a1e3-0a580af43d64",
    "backupManagementType": "AzureIaasVM",
    "containerName": "iaasvmcontainerv2;testrg;clickhousevm",
    "duration": "0:00:30.954373",
    "endTime": "2022-05-02T12:36:20.005999+00:00",
    "entityFriendlyName": "clickhousevm",
    "errorDetails": null,
    "extendedInfo": {
      "dynamicErrorMessage": null,
      "estimatedRemainingDuration": null,
      "internalPropertyBag": null,
      "progressPercentage": null,
      "propertyBag": {
        "Policy Name": "DefaultPolicy",
        "VM Name": "clickhousevm"
      },
      "tasksList": []
    },
    "isUserTriggered": null,
    "jobType": "AzureIaaSVMJob",
    "operation": "ConfigureBackup",
    "startTime": "2022-05-02T12:35:49.051626+00:00",
    "status": "Completed",
    "virtualMachineVersion": "Compute"
  },
  "resourceGroup": "testRG",
  "tags": null,
  "type": "Microsoft.RecoveryServices/vaults/backupJobs"
}
```

Make a backup of the ClickHouse VM

In order to trigger a backup job instantly, we ran the following:

```
az backup protection backup-now `
    --resource-group $RGName `
    --vault-name $RSVName `
    --container-name $CHName `
    --item-name $CHName `
    --backup-management-type AzureIaaSVM `
    --retain-until 06-05-2022
```

Output:

```
{
  "eTag": null,
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour
  ↪  ceGroups/testRG/providers/Microsoft.RecoveryServices/vaults/m
  ↪  yRSV/backupJobs/d52437c1-3ad2-4312-98a7-a75b4eea6e7e",
  "location": null,
  "name": "d52437c1-3ad2-4312-98a7-a75b4eea6e7e",
  "properties": {
    "actionsInfo": [
      "1"
    ],
    "activityId": "950770e4-ca14-11ec-a745-0a580af43d64",
    "backupManagementType": "AzureIaasVM",
    "containerName": "iaasvmcontainerv2;testrg;clickhousevm",
    "duration": "0:00:01.836772",
    "endTime": null,
    "entityFriendlyName": "clickhousevm",
    "errorDetails": null,
    "extendedInfo": {
      "dynamicErrorMessage": null,
      "estimatedRemainingDuration": null,
      "internalPropertyBag": {
        "IsInstantRpJob": "True"
      },
      "progressPercentage": null,
      "propertyBag": {
        "Recovery Point Expiry Time in UTC": "5/6/2022 12:00:00 AM",
        "VM Name": "clickhousevm"
      },
      "tasksList": [
        {
          "duration": "0:00:00",
          "endTime": null,
          "instanceId": null,
          "progressPercentage": null,
          "startTime": null,
          "status": "InProgress",
          "taskExecutionDetails": null,
          "taskId": "Take Snapshot"
        },
        {
          "duration": "0:00:00",
          "endTime": null,
          "instanceId": null,
          "progressPercentage": null,
          "startTime": null,
          "status": "NotStarted",
          "taskExecutionDetails": null,
          "taskId": "Transfer data to vault"
        }
      ]
    },
    "isUserTriggered": null,
    "jobType": "AzureIaaSVMJob",
    "operation": "Backup",
    "startTime": "2022-05-02T12:37:10.978974+00:00",
    "status": "InProgress",
    "virtualMachineVersion": "Compute"
  },
  "resourceGroup": "testRG",
  "tags": null,
  "type": "Microsoft.RecoveryServices/vaults/backupJobs"
}
```

A.2 Test environment for ClickHouse performance tests

Azure CLI commands were run in the Azure Cloud Shell (PowerShell).

Bash commands (the commands run on the VM) were run as `azureuser` (the default user) in `/home/azureuser`, unless specified otherwise.

SQL queries were run in `clickhouse-client`.

The VM used had the following specifications:

- 16 GB of RAM
- 4 vCPUs
- 2048GB SSD
- Ubuntu 16.04

The procedure for setting the environment up is generally the same as for the normal test environment (see [cref]).

A.2.1 Declare variables

The following variables were declared to make scripts more reusable. They are mostly the same as the normal test environment, except having “perf” (performance) prefixed.

```
$AzCloudUser = "torstein"        # Name of Azure user used for CLI
↪  commands
$RGName = "perfRG"               # Name of resource group
$CHName = "perfClickhouseVM"         # Name of VM running ClickHouse
$SSHKey = "mySSHKey"             # Name of SSH key used to connect
↪  to VM
$SSHPath = "/home/$AzCloudUser/.ssh/$SSHKey.pub"  # Path of SSH
↪  keys in Azure Cloud Shell storage
$SAName = "perfchbksa"               # Name of storage account used
↪  by clickhouse-backup
$ContainerName = "chbkperfcontainer" # Name of container storing
↪  clickhouse-backup data
$StagingSAName = "perfstagingchsa"   # Name of storage account used
↪  for Azure Backup staging
$SASExpDate = "2022-05-18"       # Expiry date of SAS token used by
↪  clickhouse-backup
$location = "eastus"             # Location of Azure resources
$RSVName = "perfRSV"               # Name of Recovery Services Vault
$subscription = "4b48eb85-91f3-4902-b74b-e84641fb6785"  #
↪  Subscription ID
$PolicyName = "DefaultPolicy"    # Policy to be used by Azure Backup
```

A.2.2 Set up a resource group

Create resource group:

```
az group create --name $RGName --location $location
```

Output:

```
{
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
  ↪  ceGroups/perfRG",
  "location": "eastus",
  "managedBy": null,
  "name": "perfRG",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
```

A.2.3 Set up a VM

Create an Ubuntu VM:

```
az vm create `
    --resource-group $RGName `
    --name $CHName `
    --image Canonical:UbuntuServer:16.04-LTS:16.04.202109280 `
    --admin-username azureuser `
    --size Standard_D4s_v4 `
    --os-disk-size-gb 2048 `
    --ssh-key-values $SSHPath `
    --public-ip-sku Standard
```

Output:

```
{
  "fqdns": "",
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
  ↪  ceGroups/perfRG/providers/Microsoft.Compute/virtualMachines/p⌋
  ↪  erfClickhouseVM",
  "location": "eastus",
  "macAddress": "00-22-48-25-AA-BE",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.4",
  "publicIpAddress": "20.237.81.139",
  "resourceGroup": "perfRG",
  "zones": ""
}
```

A.2.4 Install ClickHouse

Commands to install ClickHouse were copied from the installation guide in the ClickHouse documentation [InstallationClickHouseDocs]. The ClickHouse default user password was left empty.

```
sudo apt-get install -y apt-transport-https ca-certificates dirmngr
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
 ↪  8919F6BD2B48D754

echo "deb https://packages.clickhouse.com/deb stable main" | sudo
 ↪  tee \
    /etc/apt/sources.list.d/clickhouse.list
sudo apt-get update

sudo apt-get install -y clickhouse-server clickhouse-client

sudo service clickhouse-server start
```

A.2.5 Load test data

Followed instructions at [https://ghe.clickhouse.tech/]

The compressed size was 200GB when loaded in the database. The dataset was therefore loaded 5 times into different tables in ClickHouse, in order to fill the database with 1TB of compressed data.

Commands were run on the VM.

Install tool for decompression:

```
sudo apt install pixz
```

Download compressed dataset:

```
wget https://datasets.clickhouse.com/github_events_v2.native.xz
```

Create five tables for the data in clickhouse-client:

```
CREATE TABLE github_events1
(
    file_time DateTime,
    event_type Enum('CommitCommentEvent' = 1, 'CreateEvent' = 2,
     ↪  'DeleteEvent' = 3, 'ForkEvent' = 4,
                    'GollumEvent' = 5, 'IssueCommentEvent' = 6,
                     ↪  'IssuesEvent' = 7, 'MemberEvent' = 8,
                    'PublicEvent' = 9, 'PullRequestEvent' = 10,
                     ↪  'PullRequestReviewCommentEvent' = 11,
                    'PushEvent' = 12, 'ReleaseEvent' = 13,
                     ↪  'SponsorshipEvent' = 14, 'WatchEvent' = 15,
                    'GistEvent' = 16, 'FollowEvent' = 17,
                        ↪  'DownloadEvent' = 18,
                        ↪  'PullRequestReviewEvent' = 19,
                    'ForkApplyEvent' = 20, 'Event' = 21,
                     ↪  'TeamAddEvent' = 22),
    actor_login LowCardinality(String),
    repo_name LowCardinality(String),
    created_at DateTime,
    updated_at DateTime,
    action Enum('none' = 0, 'created' = 1, 'added' = 2, 'edited' =
        ↪  3, 'deleted' = 4, 'opened' = 5, 'closed' = 6, 'reopened' =
        ↪  7, 'assigned' = 8, 'unassigned' = 9,
                'labeled' = 10, 'unlabeled' = 11,
                    ↪  'review_requested' = 12,
                    ↪  'review_request_removed' = 13, 'synchronize' =
                    ↪  14, 'started' = 15, 'published' = 16, 'update'
                    ↪  = 17, 'create' = 18, 'fork' = 19, 'merged' =
                    ↪  20),
    comment_id UInt64,
    body String,
    path String,
    position Int32,
    line Int32,
    ref LowCardinality(String),
    ref_type Enum('none' = 0, 'branch' = 1, 'tag' = 2, 'repository'
     ↪  = 3, 'unknown' = 4),
    creator_user_login LowCardinality(String),
    number UInt32,
    title String,
    labels Array(LowCardinality(String)),
    state Enum('none' = 0, 'open' = 1, 'closed' = 2),
    locked UInt8,
    assignee LowCardinality(String),
    assignees Array(LowCardinality(String)),
    comments UInt32,
    author_association Enum('NONE' = 0, 'CONTRIBUTOR' = 1, 'OWNER'
     ↪  = 2, 'COLLABORATOR' = 3, 'MEMBER' = 4, 'MANNEQUIN' = 5),
    closed_at DateTime,
    merged_at DateTime,
    merge_commit_sha String,
    requested_reviewers Array(LowCardinality(String)),
    requested_teams Array(LowCardinality(String)),
    head_ref LowCardinality(String),
    head_sha String,
    base_ref LowCardinality(String),
    base_sha String,
    merged UInt8,
    mergeable UInt8,
    rebaseable UInt8,
    mergeable_state Enum('unknown' = 0, 'dirty' = 1, 'clean' = 2,
     ↪  'unstable' = 3, 'draft' = 4),
    merged_by LowCardinality(String),
    review_comments UInt32,
    maintainer_can_modify UInt8,
    commits UInt32,
    additions UInt32,
    deletions UInt32,
    changed_files UInt32,
    diff_hunk String,
    original_position UInt32,
    commit_id String,
    original_commit_id String,
    push_size UInt32,
    push_distinct_size UInt32,
    member_login LowCardinality(String),
    release_tag_name String,
    release_name String,
    review_state Enum('none' = 0, 'approved' = 1,
     ↪  'changes_requested' = 2, 'commented' = 3, 'dismissed' = 4,
     ↪  'pending' = 5)
)
ENGINE = MergeTree
ORDER BY (event_type, repo_name, created_at)

CREATE TABLE github_events2
(
    file_time DateTime,
    event_type Enum('CommitCommentEvent' = 1, 'CreateEvent' = 2,
     ↪  'DeleteEvent' = 3, 'ForkEvent' = 4,
                    'GollumEvent' = 5, 'IssueCommentEvent' = 6,
                     ↪  'IssuesEvent' = 7, 'MemberEvent' = 8,
                    'PublicEvent' = 9, 'PullRequestEvent' = 10,
                     ↪  'PullRequestReviewCommentEvent' = 11,
                    'PushEvent' = 12, 'ReleaseEvent' = 13,
                     ↪  'SponsorshipEvent' = 14, 'WatchEvent' = 15,
                    'GistEvent' = 16, 'FollowEvent' = 17,
                     ↪  'DownloadEvent' = 18,
                     ↪  'PullRequestReviewEvent' = 19,
                'ForkApplyEvent' = 20, 'Event' = 21,
                 ↪  'TeamAddEvent' = 22),
    actor_login LowCardinality(String),
    repo_name LowCardinality(String),
    created_at DateTime,
    updated_at DateTime,
    action Enum('none' = 0, 'created' = 1, 'added' = 2, 'edited' =
     ↪  3, 'deleted' = 4, 'opened' = 5, 'closed' = 6, 'reopened' =
     ↪  7, 'assigned' = 8, 'unassigned' = 9,
                'labeled' = 10, 'unlabeled' = 11,
                 ↪  'review_requested' = 12,
                 ↪  'review_request_removed' = 13, 'synchronize' =
                 ↪  14, 'started' = 15, 'published' = 16, 'update'
                 ↪  = 17, 'create' = 18, 'fork' = 19, 'merged' =
                 ↪  20),
    comment_id UInt64,
    body String,
    path String,
    position Int32,
    line Int32,
    ref LowCardinality(String),
    ref_type Enum('none' = 0, 'branch' = 1, 'tag' = 2, 'repository'
     ↪  = 3, 'unknown' = 4),
    creator_user_login LowCardinality(String),
    number UInt32,
    title String,
    labels Array(LowCardinality(String)),
    state Enum('none' = 0, 'open' = 1, 'closed' = 2),
    locked UInt8,
    assignee LowCardinality(String),
    assignees Array(LowCardinality(String)),
    comments UInt32,
    author_association Enum('NONE' = 0, 'CONTRIBUTOR' = 1, 'OWNER'
     ↪  = 2, 'COLLABORATOR' = 3, 'MEMBER' = 4, 'MANNEQUIN' = 5),
    closed_at DateTime,
    merged_at DateTime,
    merge_commit_sha String,
    requested_reviewers Array(LowCardinality(String)),
    requested_teams Array(LowCardinality(String)),
    head_ref LowCardinality(String),
    head_sha String,
    base_ref LowCardinality(String),
    base_sha String,
    merged UInt8,
    mergeable UInt8,
    rebaseable UInt8,
    mergeable_state Enum('unknown' = 0, 'dirty' = 1, 'clean' = 2,
     ↪  'unstable' = 3, 'draft' = 4),
    merged_by LowCardinality(String),
    review_comments UInt32,
    maintainer_can_modify UInt8,
    commits UInt32,
    additions UInt32,
    deletions UInt32,
    changed_files UInt32,
    diff_hunk String,
    original_position UInt32,
    commit_id String,
    original_commit_id String,
    push_size UInt32,
    push_distinct_size UInt32,
    member_login LowCardinality(String),
    release_tag_name String,
    release_name String,
    review_state Enum('none' = 0, 'approved' = 1,
     ↪  'changes_requested' = 2, 'commented' = 3, 'dismissed' = 4,
     ↪  'pending' = 5)
)
ENGINE = MergeTree
ORDER BY (event_type, repo_name, created_at)

CREATE TABLE github_events3
(
    file_time DateTime,
    event_type Enum('CommitCommentEvent' = 1, 'CreateEvent' = 2,
     ↪  'DeleteEvent' = 3, 'ForkEvent' = 4,
                    'GollumEvent' = 5, 'IssueCommentEvent' = 6,
                     ↪  'IssuesEvent' = 7, 'MemberEvent' = 8,
                    'PublicEvent' = 9, 'PullRequestEvent' = 10,
                     ↪  'PullRequestReviewCommentEvent' = 11,
                    'PushEvent' = 12, 'ReleaseEvent' = 13,
                     ↪  'SponsorshipEvent' = 14, 'WatchEvent' = 15,
                    'GistEvent' = 16, 'FollowEvent' = 17,
                     ↪  'DownloadEvent' = 18,
                     ↪  'PullRequestReviewEvent' = 19,
                    'ForkApplyEvent' = 20, 'Event' = 21,
                     ↪  'TeamAddEvent' = 22),
    actor_login LowCardinality(String),
    repo_name LowCardinality(String),
    created_at DateTime,
    updated_at DateTime,
    action Enum('none' = 0, 'created' = 1, 'added' = 2, 'edited' =
     ↪  3, 'deleted' = 4, 'opened' = 5, 'closed' = 6, 'reopened' =
     ↪  7, 'assigned' = 8, 'unassigned' = 9,
                'labeled' = 10, 'unlabeled' = 11,
                 ↪  'review_requested' = 12,
                 ↪  'review_request_removed' = 13, 'synchronize' =
                 ↪  14, 'started' = 15, 'published' = 16, 'update'
                 ↪  = 17, 'create' = 18, 'fork' = 19, 'merged' =
                 ↪  20),
    comment_id UInt64,
    body String,
    path String,
    position Int32,
    line Int32,
    ref LowCardinality(String),
    ref_type Enum('none' = 0, 'branch' = 1, 'tag' = 2, 'repository'
     ↪  = 3, 'unknown' = 4),
    creator_user_login LowCardinality(String),
    number UInt32,
    title String,
    labels Array(LowCardinality(String)),
    state Enum('none' = 0, 'open' = 1, 'closed' = 2),
    locked UInt8,
    assignee LowCardinality(String),
    assignees Array(LowCardinality(String)),
    comments UInt32,
    author_association Enum('NONE' = 0, 'CONTRIBUTOR' = 1, 'OWNER'
     ↪  = 2, 'COLLABORATOR' = 3, 'MEMBER' = 4, 'MANNEQUIN' = 5),
    closed_at DateTime,
    merged_at DateTime,
    merge_commit_sha String,
    requested_reviewers Array(LowCardinality(String)),
    requested_teams Array(LowCardinality(String)),
    head_ref LowCardinality(String),
    head_sha String,
    base_ref LowCardinality(String),
    base_sha String,
    merged UInt8,
    mergeable UInt8,
    rebaseable UInt8,
    mergeable_state Enum('unknown' = 0, 'dirty' = 1, 'clean' = 2,
     ↪  'unstable' = 3, 'draft' = 4),
    merged_by LowCardinality(String),
    review_comments UInt32,
    maintainer_can_modify UInt8,
    commits UInt32,
    additions UInt32,
    deletions UInt32,
    changed_files UInt32,
    diff_hunk String,
    original_position UInt32,
    commit_id String,
    original_commit_id String,
    push_size UInt32,
    push_distinct_size UInt32,
    member_login LowCardinality(String),
    release_tag_name String,
    release_name String,
    review_state Enum('none' = 0, 'approved' = 1,
     ↪  'changes_requested' = 2, 'commented' = 3, 'dismissed' = 4,
     ↪  'pending' = 5)
)
ENGINE = MergeTree
ORDER BY (event_type, repo_name, created_at)

CREATE TABLE github_events4
(
    file_time DateTime,
    event_type Enum('CommitCommentEvent' = 1, 'CreateEvent' = 2,
     ↪  'DeleteEvent' = 3, 'ForkEvent' = 4,
                    'GollumEvent' = 5, 'IssueCommentEvent' = 6,
                     ↪  'IssuesEvent' = 7, 'MemberEvent' = 8,
                    'PublicEvent' = 9, 'PullRequestEvent' = 10,
                     ↪  'PullRequestReviewCommentEvent' = 11,
                    'PushEvent' = 12, 'ReleaseEvent' = 13,
                     ↪  'SponsorshipEvent' = 14, 'WatchEvent' = 15,
                    'GistEvent' = 16, 'FollowEvent' = 17,
                     ↪  'DownloadEvent' = 18,
                     ↪  'PullRequestReviewEvent' = 19,
                    'ForkApplyEvent' = 20, 'Event' = 21,
                     ↪  'TeamAddEvent' = 22),
    actor_login LowCardinality(String),
    repo_name LowCardinality(String),
    created_at DateTime,
updated_at DateTime,
action Enum('none' = 0, 'created' = 1, 'added' = 2, 'edited' =
 ↪  3, 'deleted' = 4, 'opened' = 5, 'closed' = 6, 'reopened' =
 ↪  7, 'assigned' = 8, 'unassigned' = 9,
            'labeled' = 10, 'unlabeled' = 11,
             ↪  'review_requested' = 12,
             ↪  'review_request_removed' = 13, 'synchronize' =
             ↪  14, 'started' = 15, 'published' = 16, 'update'
             ↪  = 17, 'create' = 18, 'fork' = 19, 'merged' =
             ↪  20),
comment_id UInt64,
body String,
path String,
position Int32,
line Int32,
ref LowCardinality(String),
ref_type Enum('none' = 0, 'branch' = 1, 'tag' = 2, 'repository'
 ↪  = 3, 'unknown' = 4),
creator_user_login LowCardinality(String),
number UInt32,
title String,
labels Array(LowCardinality(String)),
state Enum('none' = 0, 'open' = 1, 'closed' = 2),
locked UInt8,
assignee LowCardinality(String),
assignees Array(LowCardinality(String)),
comments UInt32,
author_association Enum('NONE' = 0, 'CONTRIBUTOR' = 1, 'OWNER'
 ↪  = 2, 'COLLABORATOR' = 3, 'MEMBER' = 4, 'MANNEQUIN' = 5),
closed_at DateTime,
merged_at DateTime,
merge_commit_sha String,
requested_reviewers Array(LowCardinality(String)),
requested_teams Array(LowCardinality(String)),
head_ref LowCardinality(String),
head_sha String,
base_ref LowCardinality(String),
base_sha String,
merged UInt8,
mergeable UInt8,
rebaseable UInt8,
mergeable_state Enum('unknown' = 0, 'dirty' = 1, 'clean' = 2,
 ↪  'unstable' = 3, 'draft' = 4),
merged_by LowCardinality(String),
    review_comments UInt32,
    maintainer_can_modify UInt8,
    commits UInt32,
    additions UInt32,
    deletions UInt32,
    changed_files UInt32,
    diff_hunk String,
    original_position UInt32,
    commit_id String,
    original_commit_id String,
    push_size UInt32,
    push_distinct_size UInt32,
    member_login LowCardinality(String),
    release_tag_name String,
    release_name String,
    review_state Enum('none' = 0, 'approved' = 1,
     ↪  'changes_requested' = 2, 'commented' = 3, 'dismissed' = 4,
     ↪  'pending' = 5)
)
ENGINE = MergeTree
ORDER BY (event_type, repo_name, created_at)

CREATE TABLE github_events5
(
    file_time DateTime,
    event_type Enum('CommitCommentEvent' = 1, 'CreateEvent' = 2,
     ↪  'DeleteEvent' = 3, 'ForkEvent' = 4,
                    'GollumEvent' = 5, 'IssueCommentEvent' = 6,
                     ↪  'IssuesEvent' = 7, 'MemberEvent' = 8,
                    'PublicEvent' = 9, 'PullRequestEvent' = 10,
                     ↪  'PullRequestReviewCommentEvent' = 11,
                    'PushEvent' = 12, 'ReleaseEvent' = 13,
                     ↪  'SponsorshipEvent' = 14, 'WatchEvent' = 15,
                    'GistEvent' = 16, 'FollowEvent' = 17,
                     ↪  'DownloadEvent' = 18,
                     ↪  'PullRequestReviewEvent' = 19,
                    'ForkApplyEvent' = 20, 'Event' = 21,
                     ↪  'TeamAddEvent' = 22),
    actor_login LowCardinality(String),
    repo_name LowCardinality(String),
    created_at DateTime,
    updated_at DateTime,
action Enum('none' = 0, 'created' = 1, 'added' = 2, 'edited' =
 ↪  3, 'deleted' = 4, 'opened' = 5, 'closed' = 6, 'reopened' =
 ↪  7, 'assigned' = 8, 'unassigned' = 9,
            'labeled' = 10, 'unlabeled' = 11,
             ↪  'review_requested' = 12,
             ↪  'review_request_removed' = 13, 'synchronize' =
             ↪  14, 'started' = 15, 'published' = 16, 'update'
             ↪  = 17, 'create' = 18, 'fork' = 19, 'merged' =
             ↪  20),
comment_id UInt64,
body String,
path String,
position Int32,
line Int32,
ref LowCardinality(String),
ref_type Enum('none' = 0, 'branch' = 1, 'tag' = 2, 'repository'
 ↪  = 3, 'unknown' = 4),
creator_user_login LowCardinality(String),
number UInt32,
title String,
labels Array(LowCardinality(String)),
state Enum('none' = 0, 'open' = 1, 'closed' = 2),
locked UInt8,
assignee LowCardinality(String),
assignees Array(LowCardinality(String)),
comments UInt32,
author_association Enum('NONE' = 0, 'CONTRIBUTOR' = 1, 'OWNER'
 ↪  = 2, 'COLLABORATOR' = 3, 'MEMBER' = 4, 'MANNEQUIN' = 5),
closed_at DateTime,
merged_at DateTime,
merge_commit_sha String,
requested_reviewers Array(LowCardinality(String)),
requested_teams Array(LowCardinality(String)),
head_ref LowCardinality(String),
head_sha String,
base_ref LowCardinality(String),
base_sha String,
merged UInt8,
mergeable UInt8,
rebaseable UInt8,
mergeable_state Enum('unknown' = 0, 'dirty' = 1, 'clean' = 2,
 ↪  'unstable' = 3, 'draft' = 4),
merged_by LowCardinality(String),
review_comments UInt32,
    maintainer_can_modify UInt8,
    commits UInt32,
    additions UInt32,
    deletions UInt32,
    changed_files UInt32,
    diff_hunk String,
    original_position UInt32,
    commit_id String,
    original_commit_id String,
    push_size UInt32,
    push_distinct_size UInt32,
    member_login LowCardinality(String),
    release_tag_name String,
    release_name String,
    review_state Enum('none' = 0, 'approved' = 1,
     ↪  'changes_requested' = 2, 'commented' = 3, 'dismissed' = 4,
     ↪  'pending' = 5)
)
ENGINE = MergeTree
ORDER BY (event_type, repo_name, created_at)
```

Show the tables:

```
SHOW TABLES
FORMAT TabSeparated

-- Query id: 4b7e02e7-4afa-4440-be42-c10481fa0732
--
-- github_events1
-- github_events2
-- github_events3
-- github_events4
-- github_events5
--
-- 5 rows in set. Elapsed: 0.002 sec.
```

Insert data into tables

```
pixz -d < github_events_v2.native.xz | clickhouse-client --query
 ↪  "INSERT INTO github_events1 FORMAT Native"
pixz -d < github_events_v2.native.xz | clickhouse-client --query
 ↪  "INSERT INTO github_events2 FORMAT Native"
pixz -d < github_events_v2.native.xz | clickhouse-client --query
 ↪  "INSERT INTO github_events3 FORMAT Native"
pixz -d < github_events_v2.native.xz | clickhouse-client --query
 ↪  "INSERT INTO github_events4 FORMAT Native"
```

```
pixz -d < github_events_v2.native.xz | clickhouse-client --query
 ↪  "INSERT INTO github_events5 FORMAT Native"
```

Each command took around 3 hours. They were left to run overnight in a `screen` session. Everything appears to have worked fine.

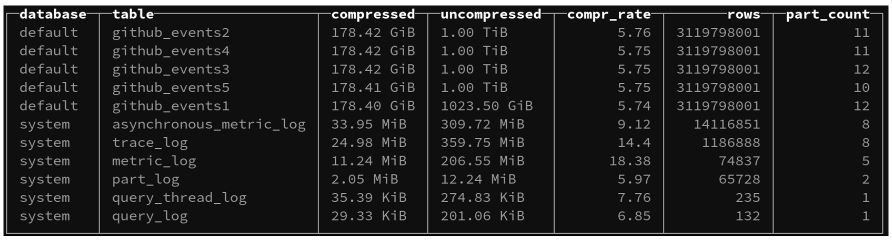
A.2.6 Verify data **ATTACH**

To verify that the data, we ran a query created by the Altinity Knowledge Base to view the size of tables in ClickHouse [DatabaseSizeTable]. See figure [@fig:table$_{sizes}$] for the results. There is a slight difference in the exact size of each table. Despite this, the number of rows in each table is exactly the same. We believe this has to do with minor "decisions" ClickHouse made when loading the data. The part count is for example not the same for all tables. The total amount of (compressed) data is 955GB (890GiB), which should be sufficient for our performance tests. Uncompressed, the data would be around 5.5TB (5TiB).

Get size of tables in ClickHouse:

```
SELECT
    database,
    table,
    formatReadableSize(sum(data_compressed_bytes) AS size) AS
     ↪  compressed,
    formatReadableSize(sum(data_uncompressed_bytes) AS usize) AS
     ↪  uncompressed,
    round(usize / size, 2) AS compr_rate,
    sum(rows) AS rows,
    count() AS part_count
FROM system.parts
WHERE (active = 1) AND (database LIKE '%') AND (table LIKE '%')
GROUP BY
    database,
    table
ORDER BY size DESC
```

Size of database tables after loading test data:



```
┌─database─┬─table───────────────────┬─compressed─┬─uncompressed─┬─compr_rate─┬───────rows─┬─part_count─┐
│ default  │ github_events2          │ 178.42 GiB │ 1.00 TiB     │       5.76 │ 3119798001 │         11 │
│ default  │ github_events4          │ 178.42 GiB │ 1.00 TiB     │       5.75 │ 3119798001 │         11 │
│ default  │ github_events3          │ 178.42 GiB │ 1.00 TiB     │       5.75 │ 3119798001 │         12 │
│ default  │ github_events5          │ 178.41 GiB │ 1.00 TiB     │       5.75 │ 3119798001 │         10 │
│ default  │ github_events1          │ 178.40 GiB │ 1023.50 GiB  │       5.74 │ 3119798001 │         12 │
│ system   │ asynchronous_metric_log │ 33.95 MiB  │ 309.72 MiB   │       9.12 │   14116851 │          8 │
│ system   │ trace_log               │ 24.98 MiB  │ 359.75 MiB   │       14.4 │    1186888 │          8 │
│ system   │ metric_log              │ 11.24 MiB  │ 206.55 MiB   │      18.38 │      74837 │          5 │
│ system   │ part_log                │ 2.05 MiB   │ 12.24 MiB    │       5.97 │      65728 │          2 │
│ system   │ query_thread_log        │ 35.39 KiB  │ 274.83 KiB   │       7.76 │        235 │          1 │
│ system   │ query_log               │ 29.33 KiB  │ 201.06 KiB   │       6.85 │        132 │          1 │
└──────────┴─────────────────────────┴────────────┴──────────────┴────────────┴────────────┴────────────┘
```

A.2.7 Install `clickhouse-backup`

The most recent (as of 2022-05-09) `clickhouse-backup` binary was downloaded from `clickhouse-backup`'s GitHub page [60].

```
# Download archive containing binary
wget https://github.com/AlexAkulov/clickhouse-backup/releases/downl⌋
↪  oad/v1.3.2/clickhouse-backup-linux-amd64.tar.gz

# Decompress archive
tar -zxvf clickhouse-backup-linux-amd64.tar.gz

# Move binary to home directory
mv build/linux/amd64/clickhouse-backup ~

# Cleanup
rmdir -p build/linux/amd64
rm clickhouse-backup-linux-amd64.tar.gz
```

A.2.8 Set up Azure Blob storage for use with `clickhouse-backup`

Commands run from the Azure Cloud Shell

Create a storage account

```
az storage account create `
    --name $SAName `
    --resource-group $RGName `
    --location eastus `
    --sku Standard_LRS `
    --encryption-services blob
```

Output:

```
{
  "accessTier": "Hot",
  "allowBlobPublicAccess": true,
  "allowCrossTenantReplication": null,
  "allowSharedKeyAccess": null,
  "allowedCopyScope": null,
  "azureFilesIdentityBasedAuthentication": null,
  "blobRestoreStatus": null,
  "creationTime": "2022-05-10T09:55:00.481299+00:00",
  "customDomain": null,
  "defaultToOAuthAuthentication": null,
  "dnsEndpointType": null,
  "enableHttpsTrafficOnly": true,
  "enableNfsV3": null,
  "encryption": {
    "encryptionIdentity": null,
    "keySource": "Microsoft.Storage",
    "keyVaultProperties": null,
    "requireInfrastructureEncryption": null,
    "services": {
      "blob": {
        "enabled": true,
        "keyType": "Account",
        "lastEnabledTime": "2022-05-10T09:55:00.606291+00:00"
      },
      "file": {
        "enabled": true,
        "keyType": "Account",
        "lastEnabledTime": "2022-05-10T09:55:00.606291+00:00"
      },
      "queue": null,
      "table": null
    }
  },
  "extendedLocation": null,
  "failoverInProgress": null,
  "geoReplicationStats": null,
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
 ↪  ceGroups/perfRG/providers/Microsoft.Storage/storageAccounts/p⌋
 ↪  erfchbksa",
  "identity": null,
  "immutableStorageWithVersioning": null,
  "isHnsEnabled": null,
  "isLocalUserEnabled": null,
  "isSftpEnabled": null,
  "keyCreationTime": {
    "key1": "2022-05-10T09:55:00.606291+00:00",
    "key2": "2022-05-10T09:55:00.606291+00:00"
  },
  "keyPolicy": null,
  "kind": "StorageV2",
  "largeFileSharesState": null,
  "lastGeoFailoverTime": null,
  "location": "eastus",
  "minimumTlsVersion": "TLS1_0",
  "name": "perfchbksa",
  "networkRuleSet": {
    "bypass": "AzureServices",
    "defaultAction": "Allow",
    "ipRules": [],
    "resourceAccessRules": null,
    "virtualNetworkRules": []
  },
  "primaryEndpoints": {
    "blob": "https://perfchbksa.blob.core.windows.net/",
    "dfs": "https://perfchbksa.dfs.core.windows.net/",
    "file": "https://perfchbksa.file.core.windows.net/",
    "internetEndpoints": null,
    "microsoftEndpoints": null,
    "queue": "https://perfchbksa.queue.core.windows.net/",
    "table": "https://perfchbksa.table.core.windows.net/",
    "web": "https://perfchbksa.z13.web.core.windows.net/"
  },
  "primaryLocation": "eastus",
  "privateEndpointConnections": [],
  "provisioningState": "Succeeded",
  "publicNetworkAccess": null,
  "resourceGroup": "perfRG",
  "routingPreference": null,
  "sasPolicy": null,
  "secondaryEndpoints": null,
  "secondaryLocation": null,
  "sku": {
    "name": "Standard_LRS",
    "tier": "Standard"
  },
  "statusOfPrimary": "available",
  "statusOfSecondary": null,
  "storageAccountSkuConversionStatus": null,
  "tags": {},
  "type": "Microsoft.Storage/storageAccounts"
}
```

Create a storage container

```
az storage container create `
    --account-name $SAName `
    --name $ContainerName `
    --auth-mode login
```

Output:

```
{
  "created": true
}
```

Enable soft delete for Blobs and Blob container

Enable soft delete for Blob container:

```
az storage account blob-service-properties update `
    --enable-container-delete-retention true `
    --container-delete-retention-days 7 `
    --account-name $SAName `
    --resource-group $RGName
```

Output:

```
{
  "automaticSnapshotPolicyEnabled": null,
  "changeFeed": null,
  "containerDeleteRetentionPolicy": {
    "allowPermanentDelete": null,
    "days": 7,
    "enabled": true
  },
  "cors": {
    "corsRules": []
  },
  "defaultServiceVersion": null,
  "deleteRetentionPolicy": {
    "allowPermanentDelete": false,
    "days": null,
    "enabled": false
  },
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
   ↪  ceGroups/perfRG/providers/Microsoft.Storage/storageAccounts/p⌋
   ↪  erfchbksa/blobServices/default",
  "isVersioningEnabled": null,
  "lastAccessTimeTrackingPolicy": null,
  "name": "default",
  "resourceGroup": "perfRG",
  "restorePolicy": null,
  "sku": null,
  "type": "Microsoft.Storage/storageAccounts/blobServices"
}
```

Enable soft delete for all Blobs:

```
az storage account blob-service-properties update --account-name
 ↪  $SAName `
    --resource-group $RGName `
    --enable-delete-retention true `
    --delete-retention-days 7
```

Output:

```
{
  "automaticSnapshotPolicyEnabled": null,
  "changeFeed": null,
  "containerDeleteRetentionPolicy": {
    "allowPermanentDelete": null,
    "days": 7,
    "enabled": true
  },
  "cors": {
    "corsRules": []
  },
  "defaultServiceVersion": null,
  "deleteRetentionPolicy": {
    "allowPermanentDelete": null,
    "days": 7,
    "enabled": true
  },
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
   ↪  ceGroups/perfRG/providers/Microsoft.Storage/storageAccounts/p⌋
   ↪  erfchbksa/blobServices/default",
  "isVersioningEnabled": null,
  "lastAccessTimeTrackingPolicy": null,
  "name": "default",
  "resourceGroup": "perfRG",
  "restorePolicy": null,
  "sku": null,
  "type": "Microsoft.Storage/storageAccounts/blobServices"
}
```

A.2.9 Configure `clickhouse-backup` to use Blob storage

Get necessary config details

The configuration details needed to configure `clickhouse-backup` were retrieved and copied into the configuration file.

Commands were run from the Azure Cloud Shell.

1. Get access key for storage account Get storage account keys:

```
az storage account keys list `
  --resource-group $RGName `
  --account-name $SAName
```

Output:

```
[
  {
    "creationTime": "2022-05-10T09:55:00.606291+00:00",
    "keyName": "key1",
    "permissions": "FULL",
    "value": "EIhee0y3zertAVbAa7cRAhYQ/2Oui25vdr6DoL05qkA+CfZZ⌋
     ↪  pFMlabzJFPtJG8Xdc735PAbA8w8r+AStl89ieA=="
  },
  {
    "creationTime": "2022-05-10T09:55:00.606291+00:00",
    "keyName": "key2",
    "permissions": "FULL",
    "value": "4Rm6943iYttVqjLBE/csYCa7rnS02xQKjAPW7C/sScZ189My⌋
     ↪  20+/Opl0HmOql1CLnL53x5NYzB/5+AStAqv2kw=="
  }
]
```

key1 was copied.

2. Get SAS token for container Generate SAS token

```
az storage container generate-sas `
    --account-name $SAName `
    --name $ContainerName `
    --permissions acdlrw `
    --expiry $SASExpDate `
    --auth-mode login `
    --as-user
```

Output:

```
"se=2022-05-18&sp=racwdl&sv=2021-04-10&sr=c&skoid=d404139d-e15⌋
 ↪  6-421c-9450-19e9734a8141&sktid=09a10672-822f-4467-a5ba-5bb⌋
 ↪  375967c05&skt=2022-05-11T07%3A01%3A18Z&ske=2022-05-18T00%3⌋
 ↪  A00%3A00Z&sks=b&skv=2021-04-10&sig=GgFsZifUWr0r71gCZaD9Pbx⌋
 ↪  RXrybhWS09Hpb2scfsKg%3D"
```

Configure `clickhouse-backup` to use Blob storage

The configuration file from the non-performance test environment was copied and modified to use the new account key and SAS token. `account_name`, `container` and `path` was also modified. It was saved as `/home/azureuser/config.yaml` on the ClickHouse VM.

```yaml
general:
  remote_storage: azblob
  max_file_size: 0
  disable_progress_bar: false
  backups_to_keep_local: 0
  backups_to_keep_remote: 0
  log_level: info
  allow_empty_backups: false
  download_concurrency: 1
  upload_concurrency: 1
  restore_schema_on_cluster: ""
  upload_by_part: true
  download_by_part: true
clickhouse:
  username: default
  password: ""
  host: localhost
  port: 9000
  disk_mapping: {}
  skip_tables:
  - system.*
  - INFORMATION_SCHEMA.*
  - information_schema.*
  timeout: 5m
  freeze_by_part: false
  secure: false
  skip_verify: false
  sync_replicated_tables: false
  log_sql_queries: true
  config_dir: /etc/clickhouse-server/
  restart_command: systemctl restart clickhouse-server
  ignore_not_exists_error_during_freeze: true
  tls_key: ""
  tls_cert: ""
  tls_ca: ""
  debug: false
s3:
  access_key: ""
  secret_key: ""
  bucket: ""
  endpoint: ""
  region: us-east-1
  acl: private
  assume_role_arn: ""
  force_path_style: false
  path: ""
  disable_ssl: false
  compression_level: 1
  compression_format: tar
  sse: ""
  disable_cert_verification: false
  storage_class: STANDARD
  concurrency: 1
  part_size: 0
  max_parts_count: 10000
  debug: false
gcs:
  credentials_file: ""
  credentials_json: ""
  bucket: ""
  path: ""
  compression_level: 1
  compression_format: tar
  debug: false
  endpoint: ""
cos:
  url: ""
  timeout: 2m
  secret_id: ""
  secret_key: ""
  path: ""
  compression_format: tar
  compression_level: 1
  debug: false
api:
  listen: localhost:7171
  enable_metrics: true
  enable_pprof: false
  username: ""
  password: ""
  secure: false
  certificate_file: ""
  private_key_file: ""
  create_integration_tables: false
  allow_parallel: false
ftp:
  address: ""
  timeout: 2m
  username: ""
  password: ""
  tls: false
  path: ""
  compression_format: tar
  compression_level: 1
  concurrency: 1
  debug: false
sftp:
  address: ""
  port: 22
  username: ""
  password: ""
  key: ""
  path: ""
  compression_format: tar
  compression_level: 1
  concurrency: 1
  debug: false
azblob:
  endpoint_suffix: core.windows.net
  account_name: "perfchbksa"
  account_key: "EIhee0y3zertAVbAa7cRAhYQ/2Oui25vdr6DoL05qkA+CfZZpFM⌟
   ↪  labzJFPtJG8Xdc735PAbA8w8r+AStl89ieA=="
  sas: "se=2022-05-18&sp=racwdl&sv=2021-04-10&sr=c&skoid=d404139d-e⌟
   ↪  156-421c-9450-19e9734a8141&sktid=09a10672-822f-4467-a5ba-5bb3⌟
   ↪  75967c05&skt=2022-05-11T07%3A01%3A18Z&ske=2022-05-18T00%3A00%⌟
   ↪  3A00Z&sks=b&skv=2021-04-10&sig=GgFsZifUWr0r71gCZaD9PbxRXrybhW⌟
   ↪  S09Hpb2scfsKg%3D"
  use_managed_identity: false
  container: "chbkperfcontainer"
  path: "https://perfchbksa.blob.core.windows.net/chbkperfcontainer"
  compression_level: 1
  compression_format: tar
  sse_key: ""
  buffer_size: 0
  buffer_count: 3
  max_parts_count: 1
```

Perform remote backup

Commands were run in `bash` on the ClickHouse VM.

While the time to make backups was not one of the criteria we decided to focus on, we measured the time anyway. The time was measured using the builtin bash tool time. In order to avoid timing sudo the commands were run as root.

Create a remote backup:

```
time ./clickhouse-backup create_remote --config config.yaml
```

Time to make a full backup with clickhouse-backup:

```
real    198m48.369s
user    14m17.555s
sys     12m38.432s
```

List remote backups:

```
./clickhouse-backup list remote --config config.yaml
# 2022/05/11 10:59:53.978855  info SELECT max(toInt64(bytes_on_disk
↪  * 1.02)) AS max_file_size FROM system.parts
# 2022-05-11T07-25-16   895.29GiB   11/05/2022 10:44:05   remote
↪    tar
```

895GiB is roughly equal to 961GB, which seems correct based on the table sizes.

A.2.10 Set up Azure Backup

Create a Recovery Services vault

```
az backup vault create --location $location --name $RSVName
↪  --resource-group $RGName
```

Output:

```
{
  "etag": "W/\"datetime'2022-05-10T09%3A58%3A14.2637541Z'\"",
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour
  ↪  ceGroups/perfRG/providers/Microsoft.RecoveryServices/vaults/p
  ↪  erfRSV",
  "identity": null,
  "location": "eastus",
  "name": "perfRSV",
  "properties": {
    "encryption": null,
    "privateEndpointConnections": null,
    "privateEndpointStateForBackup": "None",
    "privateEndpointStateForSiteRecovery": "None",
    "provisioningState": "Succeeded",
    "upgradeDetails": null
  },
  "resourceGroup": "perfRG",
  "sku": {
    "name": "Standard",
    "tier": null
  },
  "systemData": null,
  "tags": null,
  "type": "Microsoft.RecoveryServices/vaults"
}
```

Disable geo-redundancy

```
az backup vault backup-properties set --backup-storage-redundancy
 ↪  LocallyRedundant --name $RSVName --resource-group $RGName
 ↪  --subscription $subscription
```

Output:

```
{
  "eTag": null,
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
   ↪  ceGroups/perfRG/providers/Microsoft.RecoveryServices/vaults/p⌋
   ↪  erfRSV/backupstorageconfig/vaultstorageconfig",
  "location": null,
  "name": "vaultstorageconfig",
  "properties": {
    "crossRegionRestoreFlag": false,
    "dedupState": "Disabled",
    "storageModelType": "LocallyRedundant",
    "storageType": "LocallyRedundant",
    "storageTypeState": "Unlocked",
    "xcoolState": "Disabled"
  },
  "resourceGroup": "perfRG",
  "tags": null,
  "type": "Microsoft.RecoveryServices/vaults/backupstorageconfig"
}
```

Enable backup for the VM

```
az backup protection enable-for-vm `
 --resource-group $RGName `
 --vault-name $RSVName `
 --vm $CHName `
 --policy-name $PolicyName
```

Output:

```
{
  "eTag": null,
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
  ↪  ceGroups/perfRG/providers/Microsoft.RecoveryServices/vaults/p⌋
  ↪  erfRSV/backupJobs/e17cce76-2b1e-47e4-a47d-07672d5400cc",
  "location": null,
  "name": "e17cce76-2b1e-47e4-a47d-07672d5400cc",
  "properties": {
    "actionsInfo": null,
    "activityId": "529e46da-d11a-11ec-aebb-0a580af40666",
    "backupManagementType": "AzureIaasVM",
    "containerName": "iaasvmcontainerv2;perfrg;perfclickhousevm",
    "duration": "0:00:30.947098",
    "endTime": "2022-05-11T11:06:30.280573+00:00",
    "entityFriendlyName": "perfclickhousevm",
    "errorDetails": null,
    "extendedInfo": {
      "dynamicErrorMessage": null,
      "estimatedRemainingDuration": null,
      "internalPropertyBag": null,
      "progressPercentage": null,
      "propertyBag": {
        "Policy Name": "DefaultPolicy",
        "VM Name": "perfclickhousevm"
      },
      "tasksList": []
    },
    "isUserTriggered": null,
    "jobType": "AzureIaaSVMJob",
    "operation": "ConfigureBackup",
    "startTime": "2022-05-11T11:05:59.333474+00:00",
    "status": "Completed",
    "virtualMachineVersion": "Compute"
  },
  "resourceGroup": "perfRG",
  "tags": null,
  "type": "Microsoft.RecoveryServices/vaults/backupJobs"
}
```

Make a backup of the ClickHouse VM

Instantly trigger a backup job:

```
az backup protection backup-now `
    --resource-group $RGName `
    --vault-name $RSVName `
    --container-name $CHName `
    --item-name $CHName `
    --backup-management-type AzureIaaSVM `
    --retain-until 18-05-2022
```

Output:

```
{
  "eTag": null,
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
   ↪  ceGroups/perfRG/providers/Microsoft.RecoveryServices/vaults/p⌋
   ↪  erfRSV/backupJobs/07a1a270-885a-4210-b7bf-c51f2dcacdad",
  "location": null,
  "name": "07a1a270-885a-4210-b7bf-c51f2dcacdad",
  "properties": {
    "actionsInfo": [
      "1"
    ],
    "activityId": "bb5f9db8-d11a-11ec-8215-0a580af40666",
    "backupManagementType": "AzureIaasVM",
    "containerName": "iaasvmcontainerv2;perfrg;perfclickhousevm",
    "duration": "0:00:01.278993",
    "endTime": null,
    "entityFriendlyName": "perfclickhousevm",
    "errorDetails": null,
    "extendedInfo": {
      "dynamicErrorMessage": null,
      "estimatedRemainingDuration": null,
      "internalPropertyBag": {
        "IsInstantRpJob": "True"
      },
      "progressPercentage": null,
      "propertyBag": {
        "Recovery Point Expiry Time in UTC": "5/18/2022 12:00:00
         ↪  AM",
        "VM Name": "perfclickhousevm"
      },
      "tasksList": [
        {
          "duration": "0:00:00",
          "endTime": null,
          "instanceId": null,
          "progressPercentage": null,
          "startTime": null,
          "status": "InProgress",
          "taskExecutionDetails": null,
          "taskId": "Take Snapshot"
        },
        {
          "duration": "0:00:00",
          "endTime": null,
          "instanceId": null,
          "progressPercentage": null,
          "startTime": null,
          "status": "NotStarted",
          "taskExecutionDetails": null,
          "taskId": "Transfer data to vault"
        }
      ]
    },
    "isUserTriggered": null,
    "jobType": "AzureIaaSVMJob",
    "operation": "Backup",
    "startTime": "2022-05-11T11:08:52.010975+00:00",
    "status": "InProgress",
    "virtualMachineVersion": "Compute"
  },
  "resourceGroup": "perfRG",
  "tags": null,
  "type": "Microsoft.RecoveryServices/vaults/backupJobs"
}
```

Create a storage account for staging

A storage account is needed to stage recoveries from Azure Backup.

Create storage account:

```
az storage account create `
    --name $StagingSAName `
    --resource-group $RGName `
    --location eastus `
    --sku Standard_LRS
```

```
{
  "accessTier": "Hot",
  "allowBlobPublicAccess": true,
  "allowCrossTenantReplication": null,
  "allowSharedKeyAccess": null,
  "allowedCopyScope": null,
"azureFilesIdentityBasedAuthentication": null,
"blobRestoreStatus": null,
"creationTime": "2022-05-12T06:35:41.131490+00:00",
"customDomain": null,
"defaultToOAuthAuthentication": null,
"dnsEndpointType": null,
"enableHttpsTrafficOnly": true,
"enableNfsV3": null,
"encryption": {
  "encryptionIdentity": null,
  "keySource": "Microsoft.Storage",
  "keyVaultProperties": null,
  "requireInfrastructureEncryption": null,
  "services": {
    "blob": {
      "enabled": true,
      "keyType": "Account",
      "lastEnabledTime": "2022-05-12T06:35:41.272095+00:00"
    },
    "file": {
      "enabled": true,
      "keyType": "Account",
      "lastEnabledTime": "2022-05-12T06:35:41.272095+00:00"
    },
    "queue": null,
    "table": null
  }
},
"extendedLocation": null,
"failoverInProgress": null,
"geoReplicationStats": null,
"id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
 ↪  ceGroups/perfRG/providers/Microsoft.Storage/storageAccounts/p⌋
 ↪  erfstagingchsa",
"identity": null,
"immutableStorageWithVersioning": null,
"isHnsEnabled": null,
"isLocalUserEnabled": null,
"isSftpEnabled": null,
"keyCreationTime": {
  "key1": "2022-05-12T06:35:41.256500+00:00",
  "key2": "2022-05-12T06:35:41.256500+00:00"
},
"keyPolicy": null,
  "kind": "StorageV2",
  "largeFileSharesState": null,
  "lastGeoFailoverTime": null,
  "location": "eastus",
  "minimumTlsVersion": "TLS1_0",
  "name": "perfstagingchsa",
  "networkRuleSet": {
    "bypass": "AzureServices",
    "defaultAction": "Allow",
    "ipRules": [],
    "resourceAccessRules": null,
    "virtualNetworkRules": []
  },
  "primaryEndpoints": {
    "blob": "https://perfstagingchsa.blob.core.windows.net/",
    "dfs": "https://perfstagingchsa.dfs.core.windows.net/",
    "file": "https://perfstagingchsa.file.core.windows.net/",
    "internetEndpoints": null,
    "microsoftEndpoints": null,
    "queue": "https://perfstagingchsa.queue.core.windows.net/",
    "table": "https://perfstagingchsa.table.core.windows.net/",
    "web": "https://perfstagingchsa.z13.web.core.windows.net/"
  },
  "primaryLocation": "eastus",
  "privateEndpointConnections": [],
  "provisioningState": "Succeeded",
  "publicNetworkAccess": null,
  "resourceGroup": "perfRG",
  "routingPreference": null,
  "sasPolicy": null,
  "secondaryEndpoints": null,
  "secondaryLocation": null,
  "sku": {
    "name": "Standard_LRS",
    "tier": "Standard"
  },
  "statusOfPrimary": "available",
  "statusOfSecondary": null,
  "storageAccountSkuConversionStatus": null,
  "tags": {},
  "type": "Microsoft.Storage/storageAccounts"
}
```

A.3 Azure Backup performance tests

Four attempts were made at recovering the VM with Azure Backup. The first two failed, presumably because something was wrong with the restore point used.

The two last attempts used a different restore point. The time spent was measured using a stopwatch. Measurements were started the moment the restore job was started and stopped when it was possible to SSH to the VM and query the database.

Azure measures the time taken by the restore job itself. This is noted separately. The total time is influenced by how fast we were able to paste commands into the CLI. The restore job time is probably the most interesting time.

Recovery using Azure Portal:

	Time (hours:minutes:seconds)
Restore job:	00:02:10
Total:	00:03:30

Recovery using Azure CLI:

	Time (hours:minutes:seconds)
Restore job:	00:01:06
Total:	00:06:33

A.3.1 Recovery with Azure Backup via CLI (failed first attempt)

Delete VM

There are not enough vCPUs available in the quota, which means the VM has to be deleted before recovery can happen.

```
az vm delete --name $CHName --resource-group $RGName --yes
```

Preparation

Prepare environment and retrieve restore points

```
# Get RSV and set context
$RSV = Get-AzRecoveryServicesVault -Name $RSVName
 ↪  -ResourceGroupName $RGName
Set-AzRecoveryServicesVaultContext -Vault $RSV

# Select VM
$namedContainer = Get-AzRecoveryServicesBackupContainer
 ↪  -ContainerType "AzureVM" -Status "Registered" -FriendlyName
 ↪  $CHName -VaultId $RSV.ID
$backupitem = Get-AzRecoveryServicesBackupItem -Container
 ↪  $namedContainer  -WorkloadType "AzureVM" -VaultId $RSV.ID
```

```
# Get start and end date
$startDate = (Get-Date).AddDays(-7)
$endDate = Get-Date

# Store recovery points in variable
$rp = Get-AzRecoveryServicesBackupRecoveryPoint -Item $backupitem
 ↪  -StartDate $startdate.ToUniversalTime() -EndDate
 ↪  $enddate.ToUniversalTime() -VaultId $RSV.ID
```

List contents of `$rp`:

```
$rp
# RecoveryPointId    RecoveryPointType  RecoveryPointTime
 ↪  ContainerName                        ContainerType
# ---------------    -----------------  -----------------
 ↪  -------------                        -------------
# 15649757922643     FileSystemConsist... 5/11/2022 7:44:53 PM
 ↪  iaasvmcontainerv2;perfrg;perfclickh... AzureVM
# 12290901249728     FileSystemConsist... 5/11/2022 11:09:00 AM
 ↪  iaasvmcontainerv2;perfrg;perfclickh... AzureVM
```

12290901249728 is the point that was triggered manually when Azure Backup was set up (see A.2). The point is stored in `$rp[1]`.

Create restore job

Start restore job:

```
# Select recovery point
$RecPoint = $rp[1]

# Create a restore job for the backup item
$restorejob = Restore-AzRecoveryServicesBackupItem -RecoveryPoint
 ↪  $RecPoint -StorageAccountName $StagingSAName
 ↪  -StorageAccountResourceGroupName $RGName
 ↪  -TargetResourceGroupName $RGName -VaultId $RSV.ID

# Wait for the restore job to complete
Wait-AzRecoveryServicesBackupJob -Job $restorejob -Timeout 43200
# WorkloadName     Operation            Status
 ↪  StartTime                 EndTime                   JobID
# ------------     ---------            ------
 ↪  ---------                 -------                   -----
# perfclickhousevm Restore              Completed
 ↪  5/12/2022 7:13:45 AM      5/12/2022 7:14:51 AM
 ↪  30d32412-e718-4dde-a52e-c48444364cf3
```

```
# Get details of the restore job
$restorejob = Get-AzRecoveryServicesBackupJob -Job $restorejob
 ↪  -VaultId $RSV.ID
$details = Get-AzRecoveryServicesBackupJobDetail -Job $restorejob
 ↪  -VaultId $RSV.ID
```

Contents of `$details.Properties`:

```
$details.Properties

# Key                          Value
# ---                          -----
# Job Type                     Recover disks
# Target VM Name               vmName
# Target Storage Account Name perfstagingchsa
# Recovery point time          5/11/2022 11:09:00 AM
# Config Blob Name             config-perfclickhousevm-30d32412-e718-
 ↪  4dde-a52e-c48444364cf3.json
# Config Blob Container Name
 ↪  perfclickhousevm-8726c9fbe67e4ca4a5ff7e06238eac29
# Config Blob Uri              https://perfstagingchsa.blob.core.win⌋
 ↪  dows.net/perfclickhousevm-8726c9fbe67e4ca4a5ff7e06238eac29/conf⌋
 ↪  ig-perfclickhousevm-30d32412-e718-4dde-a52e-c48444364cf3.json
# Target resource group        perfRG
# Template Blob Uri            https://perfstagingchsa.blob.core.win⌋
 ↪  dows.net/perfclickhousevm-8726c9fbe67e4ca4a5ff7e06238eac29/azur⌋
 ↪  edeploy30d32412-e718-4dde-a52e-c48444364cf3.json
```

Save details:

```
$properties = $details.properties
$storageAccountName = $properties["Target Storage Account Name"]
$containerName = $properties["Config Blob Container Name"]
$templateBlobURI = $properties["Template Blob Uri"]
```

Deploy VM

Deploy VM:

```
# Template name was copied from the last part of "Template Blob Uri"
$templateName =
 ↪  "azuredeploy30d32412-e718-4dde-a52e-c48444364cf3.json"

# Set the storage account
Set-AzCurrentStorageAccount -Name $storageAccountName
 ↪  -ResourceGroupName $RGName
```

```
# Generate SAS token
$templateBlobFullURI = New-AzStorageBlobSASToken -Container
 ↪  $containerName -Blob $templateName -Permission r -FullUri

# Deploy VM (VirtualMachineName had to be specified interactively)
New-AzResourceGroupDeployment -Name $CHName -ResourceGroupName
 ↪  $RGName -TemplateUri $templateBlobFullURI
#DeploymentName          : perfClickhouseVM
#ResourceGroupName       : perfRG
#ProvisioningState       : Succeeded
#Timestamp               : 5/12/2022 7:20:25 AM
#Mode                    : Incremental
#TemplateLink            :
#         Uri            : https://perfstagingchsa.blob.core.window⌋
 ↪  s.net/perfclickhousevm-8726c9fbe67e4ca4a5ff7e06238eac29/azurede⌋
 ↪  ploy30d32412-e718-4dde-a52e-c48444364cf3.json?sv=2021-04-10&se=⌋
 ↪  2022-05-12T08%3A19%3A11Z&sr=b&sp=r
#         &sig=5Uzo66b%2B1I9IvrpVokKaqHtJiBCx0ZEl6r8SIyM0XGc%3D
#         ContentVersion : 1.0.0.0
#
#Parameters              :
# Name                             Type                       Value
# =============================    =========================
 ↪  ==========
# virtualMachineName               String
 ↪  "perfClickhouseVM"
# virtualNetwork                   String
 ↪  "perfClickhouseVMVNET"
# virtualNetworkResourceGroup      String                     "perfRG"
# subnet                           String
 ↪  "perfClickhouseVMSubnet"
# osDiskName                       String
 ↪  "perfClickhouseVMOSDisk"
# networkInterfacePrefixName       String
 ↪  "perfClickhouseVMRestoredNIC"
# publicIpAddressName              String
 ↪  "perfClickhouseVMRestoredip"
#
#Outputs                 :
#DeploymentDebugLogLevel :
```

Trouble connecting to the VM

`clickhouse-client` would not start at first. The VM was therefore restarted. After this, SSH would time out repeatedly. The VM was restarted again, but SSH still did not work. Azure's SSH "Connection troubleshoot" in the Azure Portal was attempted, but it would also hang for a long time.

Another attempt at recovering the data was attempted with the Azure Portal. The recovery appears to have succeeded. The same problems with `clickhouse-client` not starting, and SSH timing out still appeared, though. Maybe the restore point was broken in some way.

We decided to abort the attempt at recovering with Azure Backup for now. Instead, we decided to try to recover with `clickhouse-backup` instead, and then make a new restore point in Azure Backup to recover from.

A.3.2 Recovery with Azure Backup (failed second attempt)

We tried to recover from Azure Backup once again. Output was omitted for brevity, except for when waiting for the restore job, as that output contains information about the time the restore job took.

Delete VM

```
az vm delete --name $CHName --resource-group $RGName --yes
```

Get restore point

```
# Get RSV and set context
$RSV = Get-AzRecoveryServicesVault -Name $RSVName
 ↪  -ResourceGroupName $RGName
Set-AzRecoveryServicesVaultContext -Vault $RSV

# Select VM
$namedContainer = Get-AzRecoveryServicesBackupContainer
 ↪  -ContainerType "AzureVM" -Status "Registered" -FriendlyName
 ↪  $CHName -VaultId $RSV.ID
$backupitem = Get-AzRecoveryServicesBackupItem -Container
 ↪  $namedContainer  -WorkloadType "AzureVM" -VaultId $RSV.ID

# Get start and end date
$startDate = (Get-Date).AddDays(-7)
$endDate = Get-Date

# Store recovery points in variable
```

```
$rp = Get-AzRecoveryServicesBackupRecoveryPoint -Item $backupitem
 ↪  -StartDate $startdate.ToUniversalTime() -EndDate
 ↪  $enddate.ToUniversalTime() -VaultId $RSV.ID
```

Restore

Start restore job:

```
# Select recovery point
$RecPoint = $rp[1]

# Create a restore job for the backup item
$restorejob = Restore-AzRecoveryServicesBackupItem -RecoveryPoint
 ↪  $RecPoint -StorageAccountName $StagingSAName
 ↪  -StorageAccountResourceGroupName $RGName
 ↪  -TargetResourceGroupName $RGName -VaultId $RSV.ID

# Wait for the restore job to complete
Wait-AzRecoveryServicesBackupJob -Job $restorejob -Timeout 43200
# WorkloadName     Operation            Status
 ↪  StartTime                EndTime                  JobID
# ------------     ---------            ------
 ↪  ---------                -------                  -----
# perfclickhousevm Restore              Completed
 ↪  5/12/2022 11:30:24 AM     5/12/2022 11:31:34 AM
 ↪  2940f1d3-b57f-4428-9b1b-066e116a1389

# Get details of the restore job
$restorejob = Get-AzRecoveryServicesBackupJob -Job $restorejob
 ↪  -VaultId $RSV.ID
$details = Get-AzRecoveryServicesBackupJobDetail -Job $restorejob
 ↪  -VaultId $RSV.ID
```

Save details:

```
$properties = $details.properties
$storageAccountName = $properties["Target Storage Account Name"]
$containerName = $properties["Config Blob Container Name"]
$templateBlobURI = $properties["Template Blob Uri"]
```

Deploy VM (failed)

Deploy VM:

```
# Template name was copied from the last part of "Template Blob Uri"
$templateName =
 ↪  "azuredeploy2940f1d3-b57f-4428-9b1b-066e116a1389.json"
```

```
# Set the storage account
Set-AzCurrentStorageAccount -Name $storageAccountName
 ↪  -ResourceGroupName $RGName

# Generate SAS token
$templateBlobFullURI = New-AzStorageBlobSASToken -Container
 ↪  $containerName -Blob $templateName -Permission r -FullUri

# Deploy VM (VirtualMachineName had to be specified interactively)
New-AzResourceGroupDeployment -Name $CHName -ResourceGroupName
 ↪  $RGName -TemplateUri $templateBlobFullURI
# New-AzResourceGroupDeployment: 11:34:35 AM - The deployment
 ↪  'perfClickhouseVM' failed with error(s). Showing 1 out of 1
 ↪  error(s).
# Status Message: Resource /subscriptions/4b48eb85-91f3-4902-b74b-e⌟
 ↪  84641fb6785/resourceGroups/perfRG/providers/Microsoft.Network/n⌟
 ↪  etworkInterfaces/perfClickhouseVMRestoredNIC5324890b7fe44a77ae9⌟
 ↪  def7a461b6e81/ipConfigurations/c0dbf1d5514749ec849957897d9405d1
 ↪  is referencing public IP address /subscriptions/4b48eb85-91f3-4⌟
 ↪  902-b74b-e84641fb6785/resourceGroups/perfRG/providers/Microsoft⌟
 ↪  .Network/publicIPAddresses/perfClickhouseVMRestoredip that is
 ↪  already allocated to resource
 ↪  /subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resourceGro⌟
 ↪  ups/perfRG/providers/Microsoft.Network/networkInterfaces/perfCl⌟
 ↪  ickhouseVMRestoredNICb9bd95260138425bab106f324a42acbc/ipConfigu⌟
 ↪  rations/7ba12bb4efcf444d9f08dd1aff9a1cc6. (Code:
 ↪  PublicIPAddressInUse)
# CorrelationId: 2677f822-ff57-4405-8600-c96b2eb802b4
#
# DeploymentName          : perfClickhouseVM
# ResourceGroupName       : perfRG
# ProvisioningState       : Failed
# Timestamp               : 5/12/2022 11:34:30 AM
# Mode                    : Incremental
# TemplateLink            :
#                           Uri            :
 ↪  https://perfstagingchsa.blob.core.windows.net/perfclickhousevm-⌟
 ↪  b52ea1d968e04fc68ea92dea4d441451/azuredeploy2940f1d3-b57f-4428-⌟
 ↪  9b1b-066e116a1389.json?sv=2
#                           021-04-10&se=2022-05-12T12%3A33%3A08Z&s⌟
 ↪  r=b&sp=r&sig=ELTwTiFCjOc3kLiTipWd0cElPgJcGFidk8EXxJxaYfU%3D
#                           ContentVersion : 1.0.0.0
#
```

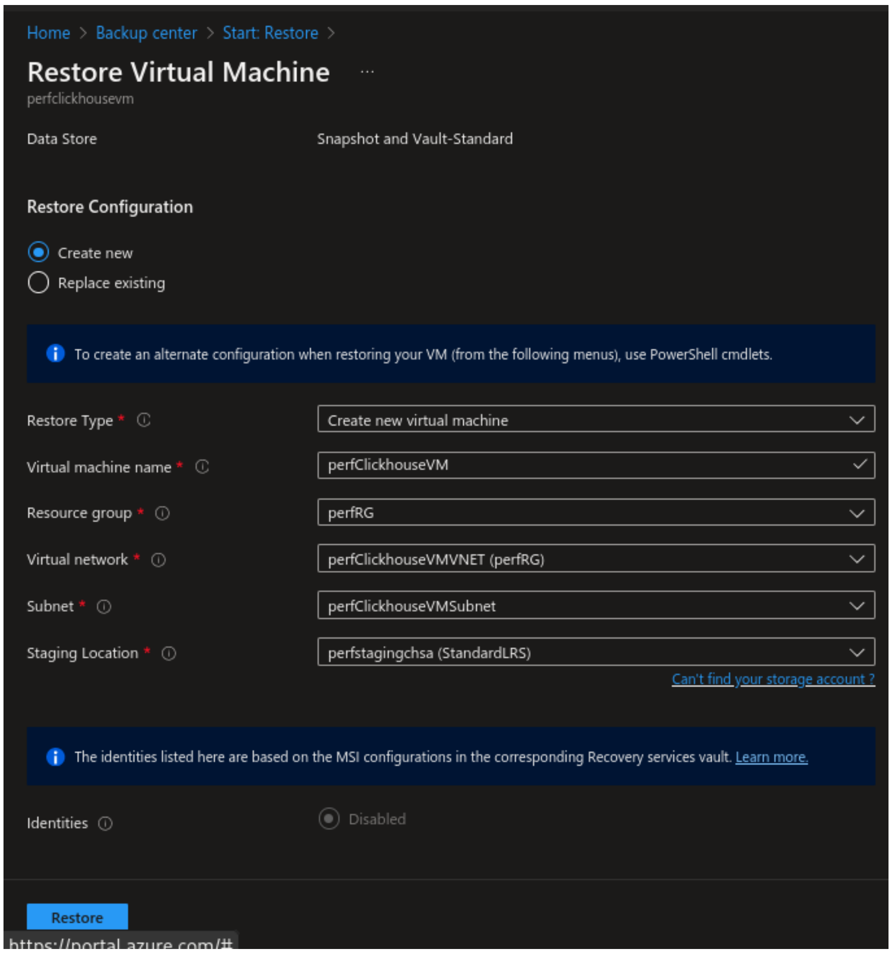
```
# Parameters                :
#                            Name                          Type
 ↪                     Value
#                            =============================
 ↪   =========================  ==========
#                            virtualMachineName            String
 ↪                     "perfClickhouseVM"
#                            virtualNetwork                String
 ↪                     "perfClickhouseVMVNET"
#                            virtualNetworkResourceGroup   String
 ↪                     "perfRG"
#                            subnet                        String
 ↪                     "perfClickhouseVMSubnet"
#                            osDiskName                    String
 ↪                     "perfClickhouseVMOSDisk"
#                            networkInterfacePrefixName    String
 ↪                     "perfClickhouseVMRestoredNIC"
#                            publicIpAddressName           String
 ↪                     "perfClickhouseVMRestoredip"
#
# Outputs                   :
```

The deployment failed because the IP address was consider to already be in use. The Azure Portal seems to be able to deal with these kinds of conflicts better, so we tried to deploy using the Portal.

Deploy VM using the Azure Portal

We decided to try to restore the VM using the Azure Portal, instead of the CLI.

Restoring the VM using the Azure Portal:



Restore Virtual Machine ...
perfclickhousevm

Data Store	Snapshot and Vault-Standard

Restore Configuration

(●) Create new
( ) Replace existing

To create an alternate configuration when restoring your VM (from the following menus), use PowerShell cmdlets.

Restore Type *	Create new virtual machine
Virtual machine name *	perfClickhouseVM
Resource group *	perfRG
Virtual network *	perfClickhouseVMVNET (perfRG)
Subnet *	perfClickhouseVMSubnet
Staging Location *	perfstagingchsa (StandardLRS)

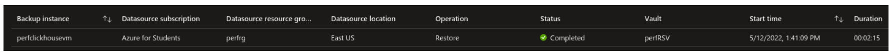
Can't find your storage account ?

The identities listed here are based on the MSI configurations in the corresponding Recovery services vault. Learn more.

Identities	(●) Disabled

According to the Azure Portal, the duration of the restore was 2 minutes and 15 seconds.

Details for the restore job:



Backup instance ↑↓	Datasource subscription	Datasource resource gro...	Datasource location	Operation	Status	Vault	Start time ↑↓	Duration
perfclickhousevm	Azure for Students	perfrg	East US	Restore	✓ Completed	perfRSV	5/12/2022, 1:41:09 PM	00:02:15

Connect to VM and verify recovery

At first `clickhouse-client` would not start. Several attempt were made at starting or restarting the ClickHouse server with `sudo systemctl [start/restart] clickhouse-server.service`.

We attempted to reboot the VM. After that, we experienced the same problem as earlier, where SSH would not connect.

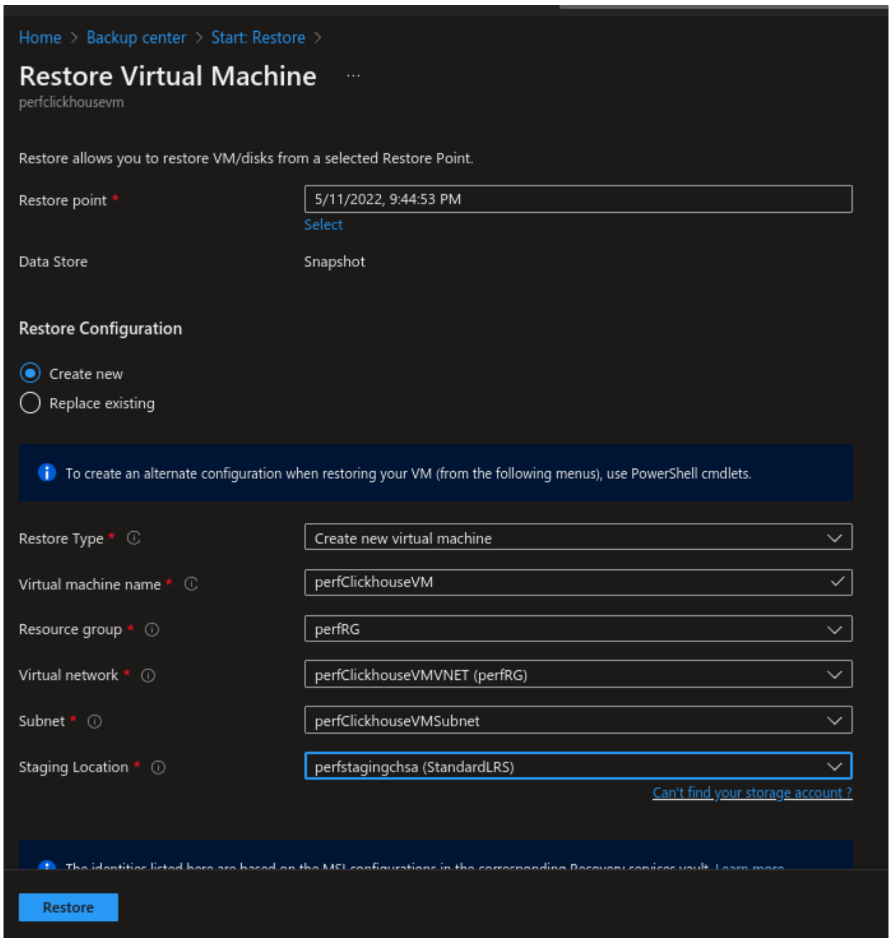
After a lot of troubleshooting, we gave up.

A.3.3 Recovery with Azure Backup via Portal (successful third attempt)

The VM was first deleted using the Azure Portal.

The VM was then restored from a different restore point (one that was automatically created on the same day, but later).

Restoring the VM:



Home > Backup center > Start: Restore >

Restore Virtual Machine

perfclickhousevm

Restore allows you to restore VM/disks from a selected Restore Point.

Restore point *: 5/11/2022, 9:44:53 PM

Data Store: Snapshot

Restore Configuration

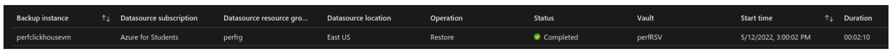
[✓ Create new] Replace existing

To create an alternate configuration when restoring your VM (from the following menus), use PowerShell cmdlets.

Restore Type *	Create new virtual machine
Virtual machine name *	perfClickhouseVM
Resource group *	perfRG
Virtual network *	perfClickhouseVMVNET (perfRG)
Subnet *	perfClickhouseVMSubnet
Staging Location *	perfstagingchsa (StandardLRS)

The identities listed here are based on the MSI configurations in the corresponding Recovery services vault. Learn more

The duration was 2 minutes, 10 seconds:



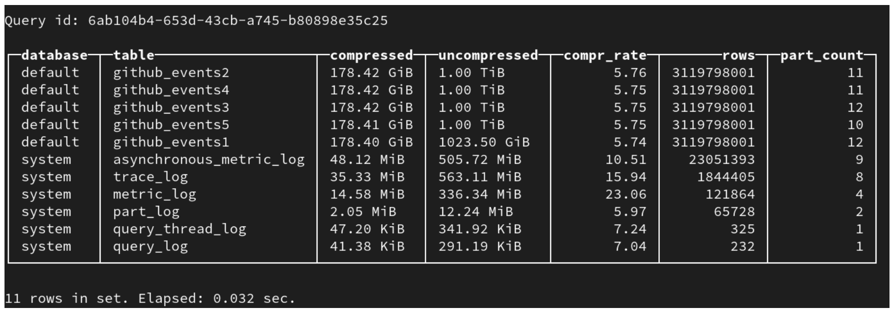
Backup instance	Datasource subscription	Datasource resource gro...	Datasource location	Operation	Status	Vault	Start time	Duration
perfclickhousevm	Azure for Students	perfrg	East US	Restore	✓ Completed	perfRSV	5/12/2022, 3:00:02 PM	00:02:10

We were able to connect to the VM immediately. `clickhouse-client` would not start on the first boot. The server was rebooted. After this, `clickhouse-client` started like normal. It took 1 minutes and 20 seconds to get `clickhouse-client` up and running after the restore job. The total time for the entire restoration was 3 minutes and 30 seconds.

Listing the table sizes:

```
SELECT
    database,
    table,
    formatReadableSize(sum(data_compressed_bytes) AS size) AS
     ↪  compressed,
    formatReadableSize(sum(data_uncompressed_bytes) AS usize) AS
     ↪  uncompressed,
    round(usize / size, 2) AS compr_rate,
    sum(rows) AS rows,
    count() AS part_count
FROM system.parts
WHERE (active = 1) AND (database LIKE '%') AND (table LIKE '%')
GROUP BY
    database,
    table
ORDER BY size DESC
```

Result:



```
Query id: 6ab104b4-653d-43cb-a745-b80898e35c25

┌─database─┬─table──────────────────┬─compressed─┬─uncompressed─┬─compr_rate─┬───────rows─┬─part_count─┐
│ default  │ github_events2         │ 178.42 GiB │ 1.00 TiB     │       5.76 │ 3119798001 │         11 │
│ default  │ github_events4         │ 178.42 GiB │ 1.00 TiB     │       5.75 │ 3119798001 │         11 │
│ default  │ github_events3         │ 178.42 GiB │ 1.00 TiB     │       5.75 │ 3119798001 │         12 │
│ default  │ github_events5         │ 178.41 GiB │ 1.00 TiB     │       5.75 │ 3119798001 │         10 │
│ default  │ github_events1         │ 178.40 GiB │ 1023.50 GiB  │       5.74 │ 3119798001 │         12 │
│ system   │ asynchronous_metric_log│ 48.12 MiB  │ 505.72 MiB   │      10.51 │   23051393 │          9 │
│ system   │ trace_log              │ 35.33 MiB  │ 563.11 MiB   │      15.94 │    1844405 │          8 │
│ system   │ metric_log             │ 14.58 MiB  │ 336.34 MiB   │      23.06 │     121864 │          4 │
│ system   │ part_log               │ 2.05 MiB   │ 12.24 MiB    │       5.97 │      65728 │          2 │
│ system   │ query_thread_log       │ 47.20 KiB  │ 341.92 KiB   │       7.24 │        325 │          1 │
│ system   │ query_log              │ 41.38 KiB  │ 291.19 KiB   │       7.04 │        232 │          1 │
└──────────┴────────────────────────┴────────────┴──────────────┴────────────┴────────────┴────────────┘

11 rows in set. Elapsed: 0.032 sec.
```

Everything appears to be in order.

A.3.4 Recovery with Azure Backup via CLI (successful fourth attempt)

We decided to try to recover the VM via the CLI one last time, this time using the same restore point as we did in the successful attempt where we used the Azure Portal.

The restore job itself took 1 minute and 6 seconds. The total time was 6 minutes and 33 seconds.

Preparation

Delete VM:

```
az vm delete --name $CHName --resource-group $RGName --yes
```

Prepare environment and retrieve restore points

```
# Get RSV and set context
$RSV = Get-AzRecoveryServicesVault -Name $RSVName
 ↪  -ResourceGroupName $RGName
Set-AzRecoveryServicesVaultContext -Vault $RSV

# Select VM
$namedContainer = Get-AzRecoveryServicesBackupContainer
 ↪  -ContainerType "AzureVM" -Status "Registered" -FriendlyName
 ↪  $CHName -VaultId $RSV.ID
$backupitem = Get-AzRecoveryServicesBackupItem -Container
 ↪  $namedContainer  -WorkloadType "AzureVM" -VaultId $RSV.ID

# Get start and end date
$startDate = (Get-Date).AddDays(-7)
$endDate = Get-Date

# Store recovery points in variable
$rp = Get-AzRecoveryServicesBackupRecoveryPoint -Item $backupitem
 ↪  -StartDate $startdate.ToUniversalTime() -EndDate
 ↪  $enddate.ToUniversalTime() -VaultId $RSV.ID
```

List contents of `$rp`:

```
$rp
# RecoveryPointId    RecoveryPointType  RecoveryPointTime
 ↪  ContainerName                        ContainerType
# ---------------    -----------------  -----------------
 ↪  -------------                        -------------
# 15649757922643     FileSystemConsist... 5/11/2022 7:44:53 PM
 ↪  iaasvmcontainerv2;perfrg;perfclickh... AzureVM
# 12290901249728     FileSystemConsist... 5/11/2022 11:09:00 AM
 ↪  iaasvmcontainerv2;perfrg;perfclickh... AzureVM
```

Since we were able to restore the VM using `$rp[0]` (15649757922643), when using the Azure Portal, we decided to try using it with the CLI as well.

Create restore job

A stopwatch was started the moment the restore job was initiated.

```
# Select recovery point
$RecPoint = $rp[0]

# Create a restore job for the backup item
```

```
$restorejob = Restore-AzRecoveryServicesBackupItem -RecoveryPoint
 ↪  $RecPoint -StorageAccountName $StagingSAName
 ↪  -StorageAccountResourceGroupName $RGName
 ↪  -TargetResourceGroupName $RGName -VaultId $RSV.ID

# Wait for the restore job to complete
Wait-AzRecoveryServicesBackupJob -Job $restorejob -Timeout 43200
# WorkloadName     Operation           Status
 ↪  StartTime                 EndTime                   JobID
# ------------     ---------           ------
 ↪  ---------                 -------                   -----
# perfclickhousevm Restore             Completed
 ↪  5/12/2022 1:29:59 PM      5/12/2022 1:31:05 PM
 ↪  17882329-4b0f-416f-8080-bbfd7a32f81b

# Get details of the restore job
$restorejob = Get-AzRecoveryServicesBackupJob -Job $restorejob
 ↪  -VaultId $RSV.ID
$details = Get-AzRecoveryServicesBackupJobDetail -Job $restorejob
 ↪  -VaultId $RSV.ID
```

The restore job lasted from 1:29:59 PM to 1:31:05 PM, which means the duration was 1 minute and 6 seconds.

Save details:

```
$properties = $details.properties
$storageAccountName = $properties["Target Storage Account Name"]
$containerName = $properties["Config Blob Container Name"]
$templateBlobURI = $properties["Template Blob Uri"]
```

Deploy VM

Generate SAS:

```
# Template name was copied from the last part of "Template Blob Uri"
$templateName =
 ↪  "azuredeploy17882329-4b0f-416f-8080-bbfd7a32f81b.json"

# Set the storage account
Set-AzCurrentStorageAccount -Name $storageAccountName
 ↪  -ResourceGroupName $RGName

# Generate SAS token
$templateBlobFullURI = New-AzStorageBlobSASToken -Container
 ↪  $containerName -Blob $templateName -Permission r -FullUri
```

Deploy VM:

```
# Deploy VM (VirtualMachineName had to be specified interactively)
New-AzResourceGroupDeployment -Name $CHName -ResourceGroupName
 ↪  $RGName -TemplateUri $templateBlobFullURI
#DeploymentName          : perfClickhouseVM
#ResourceGroupName       : perfRG
#ProvisioningState       : Succeeded
#Timestamp               : 5/12/2022 1:38:51 PM
#Mode                    : Incremental
#TemplateLink            :
#                           Uri            :
 ↪  https://perfstagingchsa.blob.core.windows.net/perfclickhousevm-⌋
 ↪  0411d36fae58488d809caa803c51a60a/azuredeploy17882329-4b0f-416f-⌋
 ↪  8080-bbfd7a32f81b.json?sv=2
#                           021-04-10&se=2022-05-12T14%3A36%3A06Z&sr⌋
 ↪  =b&sp=r&sig=xIXTM%2FIAZW40zO5nv%2BRFcB4MJrNeAhaLbYWo4oMHKz0%3D
#                           ContentVersion : 1.0.0.0
#
#Parameters              :
#                           Name                           Type
 ↪                    Value
#                           =============================
 ↪  =========================  ==========
#                           virtualMachineName             String
 ↪                    "perfClickhouseVM"
#                           virtualNetwork                 String
 ↪                    "perfClickhouseVMVNET"
#                           virtualNetworkResourceGroup    String
 ↪                    "perfRG"
#                           subnet                         String
 ↪                    "perfClickhouseVMSubnet"
#                           osDiskName                     String
 ↪                    "perfClickhouseVMOSDisk"
#                           networkInterfacePrefixName     String
 ↪                    "perfClickhouseVMRestoredNIC"
#                           publicIpAddressName            String
 ↪                    "perfClickhouseVMRestoredip"
#
#Outputs                 :
#DeploymentDebugLogLevel :
```

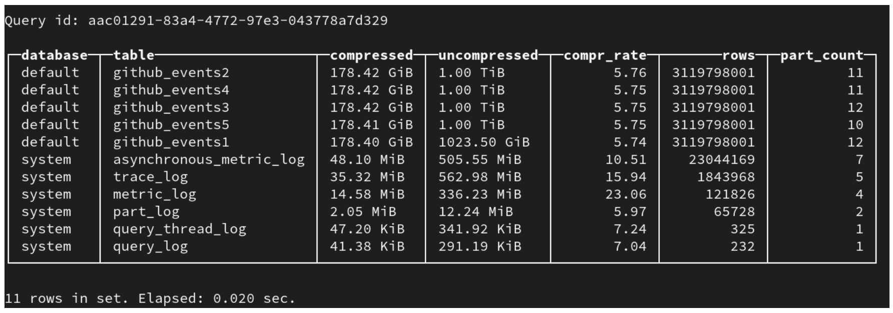
Verify that the recovery was successful

We were able to connect to the VM with SSH after 4 minutes and 49 seconds. `clickhouse-client` would not start on the first boot. The VM was restarted. After this, `clickhouse-client` started properly and we were able to query the database. The total time was 6 minutes and 33 seconds.

Listing the table sizes:

```
SELECT
    database,
    table,
    formatReadableSize(sum(data_compressed_bytes) AS size) AS
     ↪  compressed,
    formatReadableSize(sum(data_uncompressed_bytes) AS usize) AS
     ↪  uncompressed,
    round(usize / size, 2) AS compr_rate,
    sum(rows) AS rows,
    count() AS part_count
FROM system.parts
WHERE (active = 1) AND (database LIKE '%') AND (table LIKE '%')
GROUP BY
    database,
    table
ORDER BY size DESC
```

Result:



```
Query id: aac01291-83a4-4772-97e3-043778a7d329

┌─database─┬─table─────────────────────┬─compressed─┬─uncompressed─┬─compr_rate─┬───────rows─┬─part_count─┐
│ default  │ github_events2            │ 178.42 GiB │ 1.00 TiB     │       5.76 │ 3119798001 │         11 │
│ default  │ github_events4            │ 178.42 GiB │ 1.00 TiB     │       5.75 │ 3119798001 │         11 │
│ default  │ github_events3            │ 178.42 GiB │ 1.00 TiB     │       5.75 │ 3119798001 │         12 │
│ default  │ github_events5            │ 178.41 GiB │ 1.00 TiB     │       5.75 │ 3119798001 │         10 │
│ default  │ github_events1            │ 178.40 GiB │ 1023.50 GiB  │       5.74 │ 3119798001 │         12 │
│ system   │ asynchronous_metric_log   │ 48.10 MiB  │ 505.55 MiB   │      10.51 │   23044169 │          7 │
│ system   │ trace_log                 │ 35.32 MiB  │ 562.98 MiB   │      15.94 │    1843968 │          5 │
│ system   │ metric_log                │ 14.58 MiB  │ 336.23 MiB   │      23.06 │     121826 │          4 │
│ system   │ part_log                  │ 2.05 MiB   │ 12.24 MiB    │       5.97 │      65728 │          2 │
│ system   │ query_thread_log          │ 47.20 KiB  │ 341.92 KiB   │       7.24 │        325 │          1 │
│ system   │ query_log                 │ 41.38 KiB  │ 291.19 KiB   │       7.04 │        232 │          1 │
└──────────┴───────────────────────────┴────────────┴──────────────┴────────────┴────────────┴────────────┘

11 rows in set. Elapsed: 0.020 sec.
```

It appears that the recovery was successful!

A.4 `clickhouse-backup` performance test

While backing up and uploading a backup to remote storage seemed to work fine (see A.2), we were unable to actually recover from the backup. Several attempts were made at restoring from or downloading the remote backups. All were unsuccessful. Errors occured during the download phase of the recovery.

An attempt was made to back up up a single table, upload it to remote storage and then restore from it. This attempt was successful, but only used a tiny amount of data. This, and the fact that we were able to list remote backups, indicates that the configuration/connection to the remote storage itself was probably fine.

Unfortunately, we were not able to perform a proper performance test of `clickhouse-backup` with 1TB of data.

A.4.1 Recovery with `clickhouse-backup` (first set of attempts)

Rebuild VM

Delete VM:

```
# Delete the VM
az vm delete --name $CHName --resource-group $RGName --yes

# Get all resources in resource group
$resources = az resource list --resource-group $RGName |
 ↪  ConvertFrom-Json -AsHashtable

# Fetch only the ids of resources with names containing
 ↪  "Clickhouse".
$filtered = foreach($r in $resources) {
    Write-Output $r["id"] | grep Clickhouse
}

# Delete the resources
$filtered | % {Remove-AzResource -ResourceId $_ -Force}
$filtered | % {Remove-AzResource -ResourceId $_ -Force}
```

Create VM:

```
az vm create `
    --resource-group $RGName `
    --name $CHName `
    --image Canonical:UbuntuServer:16.04-LTS:16.04.202109280 `
    --admin-username azureuser `
    --size Standard_D4s_v4 `
    --os-disk-size-gb 2048 `
    --ssh-key-values $SSHPath `
    --public-ip-sku Standard
```

Install ClickHouse:

```
sudo apt-get install -y apt-transport-https ca-certificates dirmngr
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
 ↪  8919F6BD2B48D754
```

```
echo "deb https://packages.clickhouse.com/deb stable main" | sudo
 ↪  tee \
    /etc/apt/sources.list.d/clickhouse.list
sudo apt-get update

sudo apt-get install -y clickhouse-server clickhouse-client

sudo clickhouse start
sudo service clickhouse-server start
```

Install `clickhouse-backup`:

```
# Download archive containing binary
wget https://github.com/AlexAkulov/clickhouse-backup/releases/downl⌋
 ↪  oad/v1.3.2/clickhouse-backup-linux-amd64.tar.gz

# Decompress archive
tar -zxvf clickhouse-backup-linux-amd64.tar.gz

# Move binary to home directory
mv build/linux/amd64/clickhouse-backup ~

# Cleanup
rmdir -p build/linux/amd64
rm clickhouse-backup-linux-amd64.tar.gz
```

`config.yaml` was copied from the test environment (see A.2).
`config.yaml`:

```
general:
  remote_storage: azblob
  max_file_size: 0
  disable_progress_bar: false
  backups_to_keep_local: 0
  backups_to_keep_remote: 0
  log_level: info
  allow_empty_backups: false
  download_concurrency: 1
  upload_concurrency: 1
  restore_schema_on_cluster: ""
  upload_by_part: true
  download_by_part: true
clickhouse:
  username: default
  password: ""
  host: localhost
  port: 9000
  disk_mapping: {}
  skip_tables:
  - system.*
  - INFORMATION_SCHEMA.*
  - information_schema.*
  timeout: 5m
  freeze_by_part: false
  secure: false
  skip_verify: false
  sync_replicated_tables: false
  log_sql_queries: true
  config_dir: /etc/clickhouse-server/
  restart_command: systemctl restart clickhouse-server
  ignore_not_exists_error_during_freeze: true
  tls_key: ""
  tls_cert: ""
  tls_ca: ""
  debug: false
s3:
  access_key: ""
  secret_key: ""
  bucket: ""
  endpoint: ""
  region: us-east-1
  acl: private
  assume_role_arn: ""
  force_path_style: false
  path: ""
  disable_ssl: false
  compression_level: 1
  compression_format: tar
  sse: ""
  disable_cert_verification: false
  storage_class: STANDARD
  concurrency: 1
  part_size: 0
  max_parts_count: 10000
  debug: false
gcs:
  credentials_file: ""
  credentials_json: ""
  bucket: ""
  path: ""
```

```yaml
  compression_level: 1
  compression_format: tar
  debug: false
  endpoint: ""
cos:
  url: ""
  timeout: 2m
  secret_id: ""
  secret_key: ""
  path: ""
  compression_format: tar
  compression_level: 1
  debug: false
api:
  listen: localhost:7171
  enable_metrics: true
  enable_pprof: false
  username: ""
  password: ""
  secure: false
  certificate_file: ""
  private_key_file: ""
  create_integration_tables: false
  allow_parallel: false
ftp:
  address: ""
  timeout: 2m
  username: ""
  password: ""
  tls: false
  path: ""
  compression_format: tar
  compression_level: 1
  concurrency: 1
  debug: false
sftp:
  address: ""
  port: 22
  username: ""
  password: ""
  key: ""
  path: ""
  compression_format: tar
  compression_level: 1
  concurrency: 1
  debug: false
azblob:
  endpoint_suffix: core.windows.net
  account_name: "perfchbksa"
  account_key: "EIhee0y3zertAVbAa7cRAhYQ/2Oui25vdr6DoL05qkA+CfZZpFM⌟
   ↪  labzJFPtJG8Xdc735PAbA8w8r+AStl89ieA=="
  sas: "se=2022-05-18&sp=racwdl&sv=2021-04-10&sr=c&skoid=d404139d-e⌟
   ↪  156-421c-9450-19e9734a8141&sktid=09a10672-822f-4467-a5ba-5bb3⌟
   ↪  75967c05&skt=2022-05-11T07%3A01%3A18Z&ske=2022-05-18T00%3A00%⌟
   ↪  3A00Z&sks=b&skv=2021-04-10&sig=GgFsZifUWr0r71gCZaD9PbxRXrybhW⌟
   ↪  S09Hpb2scfsKg%3D"
  use_managed_identity: false
  container: "chbkperfcontainer"
  path: "https://perfchbksa.blob.core.windows.net/chbkperfcontainer"
  compression_level: 1
  compression_format: tar
  sse_key: ""
  buffer_size: 0
  buffer_count: 3
  max_parts_count: 1
```

Try to restore from remote backup

Bash commands were performed as root in /home/azureuser. SQL statements were performed in clickhouse-client.

Showing the tables in the database (no tables):

```
SHOW TABLES

-- Query id: e41781d1-477e-4d48-be0c-73e3b5f26eaa
--
-- Ok.
--
-- 0 rows in set. Elapsed: 0.002 sec.
```

Listing remote backups:

```
./clickhouse-backup list remote --config ./config.yaml
# 2022/05/12 09:39:48.379024  info SELECT max(toInt64(bytes_on_disk
 ↪  * 1.02)) AS max_file_size FROM system.parts
# 2022-05-11T07-25-16   895.29GiB   11/05/2022 10:44:05   remote
 ↪    tar
```

Restoring from the remote backup:

```
time ./clickhouse-backup restore_remote 2022-05-11T07-25-16
 ↪  --config ./config.yaml

# 2022/05/12 09:48:21.324699  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/12 09:48:21.326557  info SELECT * FROM system.disks;
# 2022/05/12 09:48:21.332100  info SELECT max(toInt64(bytes_on_disk
 ↪  * 1.02)) AS max_file_size FROM system.parts
# 2022/05/12 09:48:21.459701  info done
 ↪  backup=2022-05-11T07-25-16 duration=54ms operation=download
 ↪  size=3.36KiB table_metadata=default.github_events1
# 2022/05/12 09:48:21.507383  info done
 ↪  backup=2022-05-11T07-25-16 duration=48ms operation=download
 ↪  size=3.32KiB table_metadata=default.github_events2
# 2022/05/12 09:48:21.512022  info done
 ↪  backup=2022-05-11T07-25-16 duration=5ms operation=download
 ↪  size=3.36KiB table_metadata=default.github_events3
# 2022/05/12 09:48:21.516128  info done
 ↪  backup=2022-05-11T07-25-16 duration=4ms operation=download
 ↪  size=3.32KiB table_metadata=default.github_events4
# 2022/05/12 09:48:21.520984  info done
 ↪  backup=2022-05-11T07-25-16 duration=5ms operation=download
 ↪  size=3.28KiB table_metadata=default.github_events5
# 2022/05/12 09:49:21.524887 error can't acquire semaphore during
 ↪  downloadTableData: context canceled---------]  57.67% 59s
# 2022/05/12 09:49:21.642779 error can't acquire semaphore during
 ↪  Download: context canceled backup=2022-05-11T07-25-16
 ↪  operation=download
# 2022/05/12 09:50:21.672318 error can't acquire semaphore during
 ↪  downloadTableData: context canceled---------]   8.04% 59s
# 2022/05/12 09:50:21.770237 error one of Download go-routine
 ↪  return error: one of downloadTableData go-routine return error:
 ↪  handling file: /all_3441_4218_4/file_time.bin: context deadline
 ↪  exceeded
#
# real    2m0.461s
# user    0m10.068s
# sys     0m14.108s
```

The restoration failed.

Try to download remote backup

Since `clickhouse-backup restore_remote` failed, we tried to instead download the backup first, and then restore afterwards. This also failed with the same error.

```
time ./clickhouse-backup download 2022-05-11T07-25-16 --config
 ↪  ./config.yaml

# 2022/05/12 10:00:42.788222  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/12 10:00:42.790756  info SELECT * FROM system.disks;
# 2022/05/12 10:00:42.797256  info SELECT max(toInt64(bytes_on_disk
 ↪  * 1.02)) AS max_file_size FROM system.parts
# 2022/05/12 10:00:42.846572  info done
 ↪  backup=2022-05-11T07-25-16 duration=5ms operation=download
 ↪  size=3.36KiB table_metadata=default.github_events1
# 2022/05/12 10:00:42.853218  info done
 ↪  backup=2022-05-11T07-25-16 duration=7ms operation=download
 ↪  size=3.32KiB table_metadata=default.github_events2
# 2022/05/12 10:00:42.857772  info done
 ↪  backup=2022-05-11T07-25-16 duration=4ms operation=download
 ↪  size=3.36KiB table_metadata=default.github_events3
# 2022/05/12 10:00:42.864379  info done
 ↪  backup=2022-05-11T07-25-16 duration=7ms operation=download
 ↪  size=3.32KiB table_metadata=default.github_events4
# 2022/05/12 10:00:42.869284  info done
 ↪  backup=2022-05-11T07-25-16 duration=5ms operation=download
 ↪  size=3.28KiB table_metadata=default.github_events5
# 2022/05/12 10:01:42.873871 error can't acquire semaphore during
 ↪  downloadTableData: context canceled---------]  63.84% 59s
# 2022/05/12 10:01:42.966195 error can't acquire semaphore during
 ↪  Download: context canceled backup=2022-05-11T07-25-16
 ↪  operation=download
# 2022/05/12 10:02:43.002110 error can't acquire semaphore during
 ↪  downloadTableData: context canceled---------]   8.08% 59s
# 2022/05/12 10:02:43.087969 error one of Download go-routine
 ↪  return error: one of downloadTableData go-routine return error:
 ↪  handling file: /all_3441_4218_4/file_time.bin: context deadline
 ↪  exceeded
#
# real    2m0.314s
# user    0m10.550s
# sys     0m12.509s
```

Rebuild VM again

The VM was once again rebuilt, in the same way as in Rebuild VM.

Download remote backup again (third attempt)

Tried to download the remote backup again, like in Try to download remote backup, but it failed once again.

A.4.2 Recovery with `clickhouse-backup` (second set of attempts)

We had another go at restoring from the remote `clickhouse-backup` backups. Downloading/restoring from remote backups failed once again, and we were not sure what to do about it.

We made an attempt at uploading a very small backup to remote storage and then recovering from it. This worked fine.

Finally, we made an attempt at recovering from the local version of the remote backup. This worked well. The restoration took only 2.687 seconds.

Restore VM

The VM was restored via Azure Backup. This process is the same as in [cref].

Drop tables

Make it possible to drop tables (had to be run between drop table statements):

```
sudo touch '/var/lib/clickhouse/flags/force_drop_table' && sudo
↪  chmod 666 '/var/lib/clickhouse/flags/force_drop_table'
```

```
DROP TABLE github_events1
DROP TABLE github_events2
DROP TABLE github_events3
DROP TABLE github_events4
DROP TABLE github_events5
```

Download backup

Commands were performed as root.

List remote backups:

```
./clickhouse-backup list remote --config config.yaml
# 2022/05/14 10:00:29.509778  info SELECT max(toInt64(bytes_on_disk
↪  * 1.02)) AS max_file_size FROM system.parts
# 2022-05-11T07-25-16   895.29GiB   11/05/2022 10:44:05   remote
↪    tar
```

Drop

```
./clickhouse-backup delete local 2022-05-11T07-25-16
```

Restore from remote backup:

```
time ./clickhouse-backup restore_remote 2022-05-11T07-25-16
 ↪  --config config.yaml
# 2022/05/14 10:04:26.068449  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/14 10:04:26.071514  info SELECT * FROM system.disks;
# 2022/05/14 10:04:26.078712  info SELECT max(toInt64(bytes_on_disk
 ↪  * 1.02)) AS max_file_size FROM system.parts
# 2022/05/14 10:04:26.202816  info done
 ↪  backup=2022-05-11T07-25-16 duration=54ms operation=download
 ↪  size=3.36KiB table_metadata=default.github_events1
# 2022/05/14 10:04:26.207742  info done
 ↪  backup=2022-05-11T07-25-16 duration=5ms operation=download
 ↪  size=3.32KiB table_metadata=default.github_events2
# 2022/05/14 10:04:26.220053  info done
 ↪  backup=2022-05-11T07-25-16 duration=12ms operation=download
 ↪  size=3.36KiB table_metadata=default.github_events3
# 2022/05/14 10:04:26.225487  info done
 ↪  backup=2022-05-11T07-25-16 duration=5ms operation=download
 ↪  size=3.32KiB table_metadata=default.github_events4
# 2022/05/14 10:04:26.230170  info done
 ↪  backup=2022-05-11T07-25-16 duration=5ms operation=download
 ↪  size=3.28KiB table_metadata=default.github_events5
# 2022/05/14 10:05:26.234385 error can't acquire semaphore during
 ↪  downloadTableData: context canceled---------]  66.83% 59s
# 2022/05/14 10:05:26.314936 error can't acquire semaphore during
 ↪  Download: context canceled backup=2022-05-11T07-25-16
 ↪  operation=download
# 2022/05/14 10:06:26.354581 error can't acquire semaphore during
 ↪  downloadTableData: context canceled---------]   9.17% 59s
# 2022/05/14 10:06:26.452971 error one of Download go-routine
 ↪  return error: one of downloadTableData go-routine return error:
 ↪  handling file: /all_3441_4218_4/merge_commit_sha.bin: context
 ↪  deadline exceeded
#
# real    2m0.400s
# user    0m11.283s
# sys     0m14.051s
```

Test with a smaller amount of data (not measuring performance)

We did a test where we backup up and recovered a small amount of data, to prove that the configuration was valid.

Create empty table:

```
CREATE TABLE github_events1
```

```
(
    file_time DateTime,
    event_type Enum('CommitCommentEvent' = 1, 'CreateEvent' = 2,
     ↪  'DeleteEvent' = 3, 'ForkEvent' = 4,
                    'GollumEvent' = 5, 'IssueCommentEvent' = 6,
                     ↪  'IssuesEvent' = 7, 'MemberEvent' = 8,
                    'PublicEvent' = 9, 'PullRequestEvent' = 10,
                     ↪  'PullRequestReviewCommentEvent' = 11,
                    'PushEvent' = 12, 'ReleaseEvent' = 13,
                     ↪  'SponsorshipEvent' = 14, 'WatchEvent' = 15,
                    'GistEvent' = 16, 'FollowEvent' = 17,
                     ↪  'DownloadEvent' = 18,
                     ↪  'PullRequestReviewEvent' = 19,
                    'ForkApplyEvent' = 20, 'Event' = 21,
                     ↪  'TeamAddEvent' = 22),
    actor_login LowCardinality(String),
    repo_name LowCardinality(String),
    created_at DateTime,
    updated_at DateTime,
    action Enum('none' = 0, 'created' = 1, 'added' = 2, 'edited' =
     ↪  3, 'deleted' = 4, 'opened' = 5, 'closed' = 6, 'reopened' =
     ↪  7, 'assigned' = 8, 'unassigned' = 9,
                'labeled' = 10, 'unlabeled' = 11,
                 ↪  'review_requested' = 12,
                 ↪  'review_request_removed' = 13, 'synchronize' =
                 ↪  14, 'started' = 15, 'published' = 16, 'update'
                 ↪  = 17, 'create' = 18, 'fork' = 19, 'merged' =
                 ↪  20),
    comment_id UInt64,
    body String,
    path String,
    position Int32,
    line Int32,
    ref LowCardinality(String),
    ref_type Enum('none' = 0, 'branch' = 1, 'tag' = 2, 'repository'
     ↪  = 3, 'unknown' = 4),
    creator_user_login LowCardinality(String),
    number UInt32,
    title String,
    labels Array(LowCardinality(String)),
    state Enum('none' = 0, 'open' = 1, 'closed' = 2),
    locked UInt8,
    assignee LowCardinality(String),
    assignees Array(LowCardinality(String)),
    comments UInt32,
    author_association Enum('NONE' = 0, 'CONTRIBUTOR' = 1, 'OWNER'
     ↪  = 2, 'COLLABORATOR' = 3, 'MEMBER' = 4, 'MANNEQUIN' = 5),
    closed_at DateTime,
    merged_at DateTime,
    merge_commit_sha String,
    requested_reviewers Array(LowCardinality(String)),
    requested_teams Array(LowCardinality(String)),
    head_ref LowCardinality(String),
    head_sha String,
    base_ref LowCardinality(String),
    base_sha String,
    merged UInt8,
    mergeable UInt8,
    rebaseable UInt8,
    mergeable_state Enum('unknown' = 0, 'dirty' = 1, 'clean' = 2,
     ↪  'unstable' = 3, 'draft' = 4),
    merged_by LowCardinality(String),
    review_comments UInt32,
    maintainer_can_modify UInt8,
    commits UInt32,
    additions UInt32,
    deletions UInt32,
    changed_files UInt32,
    diff_hunk String,
    original_position UInt32,
    commit_id String,
    original_commit_id String,
    push_size UInt32,
    push_distinct_size UInt32,
    member_login LowCardinality(String),
    release_tag_name String,
    release_name String,
    review_state Enum('none' = 0, 'approved' = 1,
     ↪  'changes_requested' = 2, 'commented' = 3, 'dismissed' = 4,
     ↪  'pending' = 5)
)
ENGINE = MergeTree
ORDER BY (event_type, repo_name, created_at)
```

Create local backup:

```
./clickhouse-backup create
# 2022/05/14 10:12:59.474330  info SELECT name, engine FROM
 ↪  system.databases WHERE name NOT IN ('system', 'INFORMATION_SCHE
# MA', 'information_schema')
# 2022/05/14 10:12:59.477846  info SHOW CREATE DATABASE `default`
# 2022/05/14 10:12:59.481048  info SELECT count() FROM
 ↪  system.settings WHERE name =
 ↪  'show_table_uuid_in_table_create_query_
# if_not_nil'
# 2022/05/14 10:12:59.483473  info SELECT name FROM
 ↪  system.databases WHERE engine IN ('MySQL','PostgreSQL')
# 2022/05/14 10:12:59.485660  info
#                 SELECT
#                         countIf(name='data_path')
 ↪  is_data_path_present,
#                         countIf(name='data_paths')
 ↪  is_data_paths_present,
#                         countIf(name='uuid') is_uuid_present,
#                         countIf(name='create_table_query')
 ↪  is_create_table_query_present,
#                         countIf(name='total_bytes')
 ↪  is_total_bytes_present
#                 FROM system.columns WHERE database='system' AND
 ↪  table='tables'
#
# 2022/05/14 10:12:59.488383  info SELECT database, name, engine ,
 ↪  data_paths , uuid , create_table_query , coalesce(total_
# bytes, 0) AS total_bytes   FROM system.tables WHERE is_temporary
 ↪  = 0 SETTINGS show_table_uuid_in_table_create_query_if_no
# t_nil=1
# 2022/05/14 10:12:59.496461  info SELECT sum(bytes_on_disk) as
 ↪  size FROM system.parts WHERE database='default' AND table='
# github_events1' GROUP BY database, table
# 2022/05/14 10:12:59.502336  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/14 10:12:59.505121  info SELECT * FROM system.disks;
# 2022/05/14 10:12:59.508045  info ALTER TABLE
 ↪  `default`.`github_events1` FREEZE WITH NAME
 ↪  'b380781a01ec426e8493f83f940e7f5
# 8';
# 2022/05/14 10:12:59.624218  info done
 ↪  backup=2022-05-14T10-12-59 operation=create table=default.gith
# ub_events1
```

List backup:

```
./clickhouse-backup list
```

```
# 2022/05/14 10:13:19.784137  info SELECT value FROM
↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/14 10:13:19.787608  info SELECT * FROM system.disks;
# 2022-05-14T10-12-59   2.81KiB   14/05/2022 10:12:59   local
```

Upload backup to remote storage:

```
./clickhouse-backup upload 2022-05-14T10-12-59 -c config.yaml
```

Delete local backup:

```
./clickhouse-backup delete local 2022-05-14T10-12-59
# 2022/05/14 10:18:24.470651  info SELECT value FROM
↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/14 10:18:24.472547  info SELECT * FROM system.disks;
# 2022/05/14 10:18:24.476090  info done
↪  backup=2022-05-14T10-12-59 duration=8ms location=local
↪  operation=delete
```

Drop the table:

```
DROP TABLE github_events1
```

Restore from remote backup:

```
./clickhouse-backup restore_remote 2022-05-14T10-12-59 -c
↪  config.yaml
# 2022/05/14 10:20:05.358861  info SELECT value FROM
↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/14 10:20:05.360724  info SELECT * FROM system.disks;
# 2022/05/14 10:20:05.368823  info SELECT max(toInt64(bytes_on_disk
↪  * 1.02)) AS max_file_size FROM system.parts
# 2022/05/14 10:20:05.425455  info done
↪  backup=2022-05-14T10-12-59 duration=5ms operation=download
↪  size=2.81KiB table_metadata=default.github_events1
# 2022/05/14 10:20:05.425540  info done
↪  diff_parts=0 duration=0s operation=downloadDiffParts
# 2022/05/14 10:20:05.425578  info done
↪  backup=2022-05-14T10-12-59 duration=0s operation=download_data
↪  size=0B table=default.github_events1
# 2022/05/14 10:20:05.433103  info done
↪  backup=2022-05-14T10-12-59 duration=70ms operation=download
↪  size=2.81KiB
# 2022/05/14 10:20:05.436457  info SELECT value FROM
↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/14 10:20:05.439819  info SELECT * FROM system.disks;
# 2022/05/14 10:20:05.442308  info CREATE DATABASE IF NOT EXISTS
↪  default
```

```
# ENGINE = Atomic
# 2022/05/14 10:20:05.443762  info SELECT engine FROM
 ↪  system.databases WHERE name = 'default'
# 2022/05/14 10:20:05.447623  info DROP TABLE IF EXISTS
 ↪  `default`.`github_events1` NO DELAY
# 2022/05/14 10:20:05.451154  info CREATE DATABASE IF NOT EXISTS
 ↪  `default`
```

```
# 2022/05/14 10:20:05.452970  info CREATE TABLE
↪  default.github_events1 UUID
↪  '8742d240-9dc9-4a43-89d0-f06e9eac0b61' (`file_time` DateTime,
↪  `event_type` Enum8('CommitCommentEvent' = 1, 'CreateEvent' = 2,
↪  'DeleteEvent' = 3, 'ForkEvent' = 4, 'GollumEvent' = 5,
↪  'IssueCommentEvent' = 6, 'IssuesEvent' = 7, 'MemberEvent' = 8,
↪  'PublicEvent' = 9, 'PullRequestEvent' = 10,
↪  'PullRequestReviewCommentEvent' = 11, 'PushEvent' = 12,
↪  'ReleaseEvent' = 13, 'SponsorshipEvent' = 14, 'WatchEvent' =
↪  15, 'GistEvent' = 16, 'FollowEvent' = 17, 'DownloadEvent' = 18,
↪  'PullRequestReviewEvent' = 19, 'ForkApplyEvent' = 20, 'Event' =
↪  21, 'TeamAddEvent' = 22), `actor_login` LowCardinality(String),
↪  `repo_name` LowCardinality(String), `created_at` DateTime,
↪  `updated_at` DateTime, `action` Enum8('none' = 0, 'created' =
↪  1, 'added' = 2, 'edited' = 3, 'deleted' = 4, 'opened' = 5,
↪  'closed' = 6, 'reopened' = 7, 'assigned' = 8, 'unassigned' = 9,
↪  'labeled' = 10, 'unlabeled' = 11, 'review_requested' = 12,
↪  'review_request_removed' = 13, 'synchronize' = 14, 'started' =
↪  15, 'published' = 16, 'update' = 17, 'create' = 18, 'fork' =
↪  19, 'merged' = 20), `comment_id` UInt64, `body` String, `path`
↪  String, `position` Int32, `line` Int32, `ref`
↪  LowCardinality(String), `ref_type` Enum8('none' = 0, 'branch' =
↪  1, 'tag' = 2, 'repository' = 3, 'unknown' = 4),
↪  `creator_user_login` LowCardinality(String), `number` UInt32,
↪  `title` String, `labels` Array(LowCardinality(String)), `state`
↪  Enum8('none' = 0, 'open' = 1, 'closed' = 2), `locked` UInt8,
↪  `assignee` LowCardinality(String), `assignees`
↪  Array(LowCardinality(String)), `comments` UInt32,
↪  `author_association` Enum8('NONE' = 0, 'CONTRIBUTOR' = 1,
↪  'OWNER' = 2, 'COLLABORATOR' = 3, 'MEMBER' = 4, 'MANNEQUIN' =
↪  5), `closed_at` DateTime, `merged_at` DateTime,
↪  `merge_commit_sha` String, `requested_reviewers`
↪  Array(LowCardinality(String)), `requested_teams`
↪  Array(LowCardinality(String)), `head_ref`
↪  LowCardinality(String), `head_sha` String, `base_ref`
↪  LowCardinality(String), `base_sha` String, `merged` UInt8,
↪  `mergeable` UInt8, `rebaseable` UInt8, `mergeable_state`
↪  Enum8('unknown' = 0, 'dirty' = 1, 'clean' = 2, 'unstable' = 3,
↪  'draft' = 4), `merged_by` LowCardinality(String),
↪  `review_comments` UInt32, `maintainer_can_modify` UInt8,
↪  `commits` UInt32, `additions` UInt32, `deletions` UInt32,
↪  `changed_files` UInt32, `diff_hunk` String, `original_position`
↪  UInt32, `commit_id` String, `original_commit_id` String,
↪  `push_size` UInt32, `push_distinct_size` UInt32, `member_login`
↪  LowCardinality(String), `release_tag_name` String,
↪  `release_name` String, `review_state` Enum8('none' = 0,
↪  'approved' = 1, 'changes_requested' = 2, 'commented' = 3,
↪  'dismissed' = 4, 'pending' = 5)) ENGINE = MergeTree ORDER BY
↪  (event_type, repo_name, created_at) SETTINGS index_granularity
↪  = 8192
```

```
# 2022/05/14 10:20:05.527062  info SELECT count() FROM
↪  system.settings WHERE name =
↪  'show_table_uuid_in_table_create_query_if_not_nil'
# 2022/05/14 10:20:05.544058  info SELECT name FROM
↪  system.databases WHERE engine IN ('MySQL','PostgreSQL')
# 2022/05/14 10:20:05.546963  info
#                 SELECT
#                         countIf(name='data_path')
↪  is_data_path_present,
#                         countIf(name='data_paths')
↪  is_data_paths_present,
#                         countIf(name='uuid') is_uuid_present,
#                         countIf(name='create_table_query')
↪  is_create_table_query_present,
#                         countIf(name='total_bytes')
↪  is_total_bytes_present
#                 FROM system.columns WHERE database='system' AND
↪  table='tables'
#
# 2022/05/14 10:20:05.568889  info SELECT database, name, engine ,
↪  data_paths , uuid , create_table_query , coalesce(total_bytes,
↪  0) AS total_bytes   FROM system.tables WHERE is_temporary = 0
↪  SETTINGS show_table_uuid_in_table_create_query_if_not_nil=1
# 2022/05/14 10:20:05.578972  info SELECT sum(bytes_on_disk) as
↪  size FROM system.parts WHERE database='default' AND
↪  table='github_events1' GROUP BY database, table
# 2022/05/14 10:20:05.582682  info done
↪  backup=2022-05-14T10-12-59 operation=restore
↪  table=default.github_events1
# 2022/05/14 10:20:05.582734  info done
↪  backup=2022-05-14T10-12-59 duration=59ms operation=restore
# 2022/05/14 10:20:05.582763  info done
↪  backup=2022-05-14T10-12-59 operation=restore
```

Attempt to restore from local backups

The VM was restored with Azure Backup, in order to recover the local backup.
The restore was timed with `time`. It took 2.687 seconds.

```
./clickhouse-backup list
# 2022/05/14 10:47:31.108857  info SELECT value FROM
↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/14 10:47:31.112370  info SELECT * FROM system.disks;
# 2022-05-11T07-25-16   895.28GiB   11/05/2022 07:25:17   local
```

In `clickhouse-client`, all tables were dropped:

```
DROP TABLE github_events1
DROP TABLE github_events2
DROP TABLE github_events3
DROP TABLE github_events4
DROP TABLE github_events5
```

Before each drop table statement, the following command had to be run in bash:

```
sudo touch '/var/lib/clickhouse/flags/force_drop_table' && sudo
↪  chmod 666 '/var/lib/clickhouse/flags/force_drop_table'
```

Show tables:

```
SHOW TABLES

-- Query id: d104549f-c266-498a-a1c2-07dc85f9ea0e
--
-- Ok.
--
-- 0 rows in set. Elapsed: 0.002 sec.
```

ClickHouse was restarted with `clickhouse restart` before restoring from backup.

Measure time to restore from local backup:

```
time ./clickhouse-backup restore 2022-05-11T07-25-16
# 2022/05/14 11:03:48.480915  info SELECT value FROM
↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/14 11:03:48.484637  info SELECT * FROM system.disks;
# 2022/05/14 11:03:48.490860  info CREATE DATABASE IF NOT EXISTS
↪  default
# ENGINE = Atomic
# 2022/05/14 11:03:48.492717  info SELECT engine FROM
↪  system.databases WHERE name = 'default'
# 2022/05/14 11:03:48.495213  info DROP TABLE IF EXISTS
↪  `default`.`github_events1` NO DELAY
# 2022/05/14 11:03:48.496391  info SELECT engine FROM
↪  system.databases WHERE name = 'default'
# 2022/05/14 11:03:48.499202  info DROP TABLE IF EXISTS
↪  `default`.`github_events2` NO DELAY
# 2022/05/14 11:03:48.504034  info SELECT engine FROM
↪  system.databases WHERE name = 'default'
# 2022/05/14 11:03:48.509525  info DROP TABLE IF EXISTS
↪  `default`.`github_events3` NO DELAY
```

```
# 2022/05/14 11:03:48.510781  info SELECT engine FROM
 ↪  system.databases WHERE name = 'default'
# 2022/05/14 11:03:48.515357  info DROP TABLE IF EXISTS
 ↪  `default`.`github_events4` NO DELAY
# 2022/05/14 11:03:48.516841  info SELECT engine FROM
 ↪  system.databases WHERE name = 'default'
# 2022/05/14 11:03:48.519688  info DROP TABLE IF EXISTS
 ↪  `default`.`github_events5` NO DELAY
# 2022/05/14 11:03:48.522352  info CREATE DATABASE IF NOT EXISTS
 ↪  `default`
```

```
# 2022/05/14 11:03:48.523692  info CREATE TABLE
↪  default.github_events1 UUID
↪  '1711c114-1286-4a0b-8f9a-11fbc6d35b89' (`file_time` DateTime,
↪  `event_type` Enum8('CommitCommentEvent' = 1, 'CreateEvent' = 2,
↪  'DeleteEvent' = 3, 'ForkEvent' = 4, 'GollumEvent' = 5,
↪  'IssueCommentEvent' = 6, 'IssuesEvent' = 7, 'MemberEvent' = 8,
↪  'PublicEvent' = 9, 'PullRequestEvent' = 10,
↪  'PullRequestReviewCommentEvent' = 11, 'PushEvent' = 12,
↪  'ReleaseEvent' = 13, 'SponsorshipEvent' = 14, 'WatchEvent' =
↪  15, 'GistEvent' = 16, 'FollowEvent' = 17, 'DownloadEvent' = 18,
↪  'PullRequestReviewEvent' = 19, 'ForkApplyEvent' = 20, 'Event' =
↪  21, 'TeamAddEvent' = 22), `actor_login` LowCardinality(String),
↪  `repo_name` LowCardinality(String), `created_at` DateTime,
↪  `updated_at` DateTime, `action` Enum8('none' = 0, 'created' =
↪  1, 'added' = 2, 'edited' = 3, 'deleted' = 4, 'opened' = 5,
↪  'closed' = 6, 'reopened' = 7, 'assigned' = 8, 'unassigned' = 9,
↪  'labeled' = 10, 'unlabeled' = 11, 'review_requested' = 12,
↪  'review_request_removed' = 13, 'synchronize' = 14, 'started' =
↪  15, 'published' = 16, 'update' = 17, 'create' = 18, 'fork' =
↪  19, 'merged' = 20), `comment_id` UInt64, `body` String, `path`
↪  String, `position` Int32, `line` Int32, `ref`
↪  LowCardinality(String), `ref_type` Enum8('none' = 0, 'branch' =
↪  1, 'tag' = 2, 'repository' = 3, 'unknown' = 4),
↪  `creator_user_login` LowCardinality(String), `number` UInt32,
↪  `title` String, `labels` Array(LowCardinality(String)), `state`
↪  Enum8('none' = 0, 'open' = 1, 'closed' = 2), `locked` UInt8,
↪  `assignee` LowCardinality(String), `assignees`
↪  Array(LowCardinality(String)), `comments` UInt32,
↪  `author_association` Enum8('NONE' = 0, 'CONTRIBUTOR' = 1,
↪  'OWNER' = 2, 'COLLABORATOR' = 3, 'MEMBER' = 4, 'MANNEQUIN' =
↪  5), `closed_at` DateTime, `merged_at` DateTime,
↪  `merge_commit_sha` String, `requested_reviewers`
↪  Array(LowCardinality(String)), `requested_teams`
↪  Array(LowCardinality(String)), `head_ref`
↪  LowCardinality(String), `head_sha` String, `base_ref`
↪  LowCardinality(String), `base_sha` String, `merged` UInt8,
↪  `mergeable` UInt8, `rebaseable` UInt8, `mergeable_state`
↪  Enum8('unknown' = 0, 'dirty' = 1, 'clean' = 2, 'unstable' = 3,
↪  'draft' = 4), `merged_by` LowCardinality(String),
↪  `review_comments` UInt32, `maintainer_can_modify` UInt8,
↪  `commits` UInt32, `additions` UInt32, `deletions` UInt32,
↪  `changed_files` UInt32, `diff_hunk` String, `original_position`
↪  UInt32, `commit_id` String, `original_commit_id` String,
↪  `push_size` UInt32, `push_distinct_size` UInt32, `member_login`
↪  LowCardinality(String), `release_tag_name` String,
↪  `release_name` String, `review_state` Enum8('none' = 0,
↪  'approved' = 1, 'changes_requested' = 2, 'commented' = 3,
↪  'dismissed' = 4, 'pending' = 5)) ENGINE = MergeTree ORDER BY
↪  (event_type, repo_name, created_at) SETTINGS index_granularity
↪  = 8192
```

```
# 2022/05/14 11:03:48.529620  warn can't create table
 ↪  'default.github_events1': code: 57, message: Directory for
 ↪  table data store/171/1711c114-1286-4a0b-8f9a-11fbc6d35b89/
 ↪  already exists, will try again backup=2022-05-11T07-25-16
 ↪  operation=restore
# 2022/05/14 11:03:48.529647  info CREATE DATABASE IF NOT EXISTS
 ↪  `default`
```

```
# 2022/05/14 11:03:48.531904  info CREATE TABLE
↪  default.github_events2 UUID
↪  '1991bd38-adaf-4691-aa49-c04b27f61485' (`file_time` DateTime,
↪  `event_type` Enum8('CommitCommentEvent' = 1, 'CreateEvent' = 2,
↪  'DeleteEvent' = 3, 'ForkEvent' = 4, 'GollumEvent' = 5,
↪  'IssueCommentEvent' = 6, 'IssuesEvent' = 7, 'MemberEvent' = 8,
↪  'PublicEvent' = 9, 'PullRequestEvent' = 10,
↪  'PullRequestReviewCommentEvent' = 11, 'PushEvent' = 12,
↪  'ReleaseEvent' = 13, 'SponsorshipEvent' = 14, 'WatchEvent' =
↪  15, 'GistEvent' = 16, 'FollowEvent' = 17, 'DownloadEvent' = 18,
↪  'PullRequestReviewEvent' = 19, 'ForkApplyEvent' = 20, 'Event' =
↪  21, 'TeamAddEvent' = 22), `actor_login` LowCardinality(String),
↪  `repo_name` LowCardinality(String), `created_at` DateTime,
↪  `updated_at` DateTime, `action` Enum8('none' = 0, 'created' =
↪  1, 'added' = 2, 'edited' = 3, 'deleted' = 4, 'opened' = 5,
↪  'closed' = 6, 'reopened' = 7, 'assigned' = 8, 'unassigned' = 9,
↪  'labeled' = 10, 'unlabeled' = 11, 'review_requested' = 12,
↪  'review_request_removed' = 13, 'synchronize' = 14, 'started' =
↪  15, 'published' = 16, 'update' = 17, 'create' = 18, 'fork' =
↪  19, 'merged' = 20), `comment_id` UInt64, `body` String, `path`
↪  String, `position` Int32, `line` Int32, `ref`
↪  LowCardinality(String), `ref_type` Enum8('none' = 0, 'branch' =
↪  1, 'tag' = 2, 'repository' = 3, 'unknown' = 4),
↪  `creator_user_login` LowCardinality(String), `number` UInt32,
↪  `title` String, `labels` Array(LowCardinality(String)), `state`
↪  Enum8('none' = 0, 'open' = 1, 'closed' = 2), `locked` UInt8,
↪  `assignee` LowCardinality(String), `assignees`
↪  Array(LowCardinality(String)), `comments` UInt32,
↪  `author_association` Enum8('NONE' = 0, 'CONTRIBUTOR' = 1,
↪  'OWNER' = 2, 'COLLABORATOR' = 3, 'MEMBER' = 4, 'MANNEQUIN' =
↪  5), `closed_at` DateTime, `merged_at` DateTime,
↪  `merge_commit_sha` String, `requested_reviewers`
↪  Array(LowCardinality(String)), `requested_teams`
↪  Array(LowCardinality(String)), `head_ref`
↪  LowCardinality(String), `head_sha` String, `base_ref`
↪  LowCardinality(String), `base_sha` String, `merged` UInt8,
↪  `mergeable` UInt8, `rebaseable` UInt8, `mergeable_state`
↪  Enum8('unknown' = 0, 'dirty' = 1, 'clean' = 2, 'unstable' = 3,
↪  'draft' = 4), `merged_by` LowCardinality(String),
↪  `review_comments` UInt32, `maintainer_can_modify` UInt8,
↪  `commits` UInt32, `additions` UInt32, `deletions` UInt32,
↪  `changed_files` UInt32, `diff_hunk` String, `original_position`
↪  UInt32, `commit_id` String, `original_commit_id` String,
↪  `push_size` UInt32, `push_distinct_size` UInt32, `member_login`
↪  LowCardinality(String), `release_tag_name` String,
↪  `release_name` String, `review_state` Enum8('none' = 0,
↪  'approved' = 1, 'changes_requested' = 2, 'commented' = 3,
↪  'dismissed' = 4, 'pending' = 5)) ENGINE = MergeTree ORDER BY
↪  (event_type, repo_name, created_at) SETTINGS index_granularity
↪  = 8192
```

```
# 2022/05/14 11:03:48.535293  warn can't create table
 ↪  'default.github_events2': code: 57, message: Directory for
 ↪  table data store/199/1991bd38-adaf-4691-aa49-c04b27f61485/
 ↪  already exists, will try again backup=2022-05-11T07-25-16
 ↪  operation=restore
# 2022/05/14 11:03:48.535317  info CREATE DATABASE IF NOT EXISTS
 ↪  `default`
```

```
# 2022/05/14 11:03:48.537514  info CREATE TABLE
↪  default.github_events3 UUID
↪  'c403ed92-213a-4245-b98b-408d6fed9377' (`file_time` DateTime,
↪  `event_type` Enum8('CommitCommentEvent' = 1, 'CreateEvent' = 2,
↪  'DeleteEvent' = 3, 'ForkEvent' = 4, 'GollumEvent' = 5,
↪  'IssueCommentEvent' = 6, 'IssuesEvent' = 7, 'MemberEvent' = 8,
↪  'PublicEvent' = 9, 'PullRequestEvent' = 10,
↪  'PullRequestReviewCommentEvent' = 11, 'PushEvent' = 12,
↪  'ReleaseEvent' = 13, 'SponsorshipEvent' = 14, 'WatchEvent' =
↪  15, 'GistEvent' = 16, 'FollowEvent' = 17, 'DownloadEvent' = 18,
↪  'PullRequestReviewEvent' = 19, 'ForkApplyEvent' = 20, 'Event' =
↪  21, 'TeamAddEvent' = 22), `actor_login` LowCardinality(String),
↪  `repo_name` LowCardinality(String), `created_at` DateTime,
↪  `updated_at` DateTime, `action` Enum8('none' = 0, 'created' =
↪  1, 'added' = 2, 'edited' = 3, 'deleted' = 4, 'opened' = 5,
↪  'closed' = 6, 'reopened' = 7, 'assigned' = 8, 'unassigned' = 9,
↪  'labeled' = 10, 'unlabeled' = 11, 'review_requested' = 12,
↪  'review_request_removed' = 13, 'synchronize' = 14, 'started' =
↪  15, 'published' = 16, 'update' = 17, 'create' = 18, 'fork' =
↪  19, 'merged' = 20), `comment_id` UInt64, `body` String, `path`
↪  String, `position` Int32, `line` Int32, `ref`
↪  LowCardinality(String), `ref_type` Enum8('none' = 0, 'branch' =
↪  1, 'tag' = 2, 'repository' = 3, 'unknown' = 4),
↪  `creator_user_login` LowCardinality(String), `number` UInt32,
↪  `title` String, `labels` Array(LowCardinality(String)), `state`
↪  Enum8('none' = 0, 'open' = 1, 'closed' = 2), `locked` UInt8,
↪  `assignee` LowCardinality(String), `assignees`
↪  Array(LowCardinality(String)), `comments` UInt32,
↪  `author_association` Enum8('NONE' = 0, 'CONTRIBUTOR' = 1,
↪  'OWNER' = 2, 'COLLABORATOR' = 3, 'MEMBER' = 4, 'MANNEQUIN' =
↪  5), `closed_at` DateTime, `merged_at` DateTime,
↪  `merge_commit_sha` String, `requested_reviewers`
↪  Array(LowCardinality(String)), `requested_teams`
↪  Array(LowCardinality(String)), `head_ref`
↪  LowCardinality(String), `head_sha` String, `base_ref`
↪  LowCardinality(String), `base_sha` String, `merged` UInt8,
↪  `mergeable` UInt8, `rebaseable` UInt8, `mergeable_state`
↪  Enum8('unknown' = 0, 'dirty' = 1, 'clean' = 2, 'unstable' = 3,
↪  'draft' = 4), `merged_by` LowCardinality(String),
↪  `review_comments` UInt32, `maintainer_can_modify` UInt8,
↪  `commits` UInt32, `additions` UInt32, `deletions` UInt32,
↪  `changed_files` UInt32, `diff_hunk` String, `original_position`
↪  UInt32, `commit_id` String, `original_commit_id` String,
↪  `push_size` UInt32, `push_distinct_size` UInt32, `member_login`
↪  LowCardinality(String), `release_tag_name` String,
↪  `release_name` String, `review_state` Enum8('none' = 0,
↪  'approved' = 1, 'changes_requested' = 2, 'commented' = 3,
↪  'dismissed' = 4, 'pending' = 5)) ENGINE = MergeTree ORDER BY
↪  (event_type, repo_name, created_at) SETTINGS index_granularity
↪  = 8192
```

```
# 2022/05/14 11:03:48.541332  warn can't create table
 ↪  'default.github_events3': code: 57, message: Directory for
 ↪  table data store/c40/c403ed92-213a-4245-b98b-408d6fed9377/
 ↪  already exists, will try again backup=2022-05-11T07-25-16
 ↪  operation=restore
# 2022/05/14 11:03:48.541354  info CREATE DATABASE IF NOT EXISTS
 ↪  `default`
```

```
# 2022/05/14 11:03:48.543374  info CREATE TABLE
↪  default.github_events4 UUID
↪  '6e6a4e02-a9fa-42ad-a0ce-af77e49a2aee' (`file_time` DateTime,
↪  `event_type` Enum8('CommitCommentEvent' = 1, 'CreateEvent' = 2,
↪  'DeleteEvent' = 3, 'ForkEvent' = 4, 'GollumEvent' = 5,
↪  'IssueCommentEvent' = 6, 'IssuesEvent' = 7, 'MemberEvent' = 8,
↪  'PublicEvent' = 9, 'PullRequestEvent' = 10,
↪  'PullRequestReviewCommentEvent' = 11, 'PushEvent' = 12,
↪  'ReleaseEvent' = 13, 'SponsorshipEvent' = 14, 'WatchEvent' =
↪  15, 'GistEvent' = 16, 'FollowEvent' = 17, 'DownloadEvent' = 18,
↪  'PullRequestReviewEvent' = 19, 'ForkApplyEvent' = 20, 'Event' =
↪  21, 'TeamAddEvent' = 22), `actor_login` LowCardinality(String),
↪  `repo_name` LowCardinality(String), `created_at` DateTime,
↪  `updated_at` DateTime, `action` Enum8('none' = 0, 'created' =
↪  1, 'added' = 2, 'edited' = 3, 'deleted' = 4, 'opened' = 5,
↪  'closed' = 6, 'reopened' = 7, 'assigned' = 8, 'unassigned' = 9,
↪  'labeled' = 10, 'unlabeled' = 11, 'review_requested' = 12,
↪  'review_request_removed' = 13, 'synchronize' = 14, 'started' =
↪  15, 'published' = 16, 'update' = 17, 'create' = 18, 'fork' =
↪  19, 'merged' = 20), `comment_id` UInt64, `body` String, `path`
↪  String, `position` Int32, `line` Int32, `ref`
↪  LowCardinality(String), `ref_type` Enum8('none' = 0, 'branch' =
↪  1, 'tag' = 2, 'repository' = 3, 'unknown' = 4),
↪  `creator_user_login` LowCardinality(String), `number` UInt32,
↪  `title` String, `labels` Array(LowCardinality(String)), `state`
↪  Enum8('none' = 0, 'open' = 1, 'closed' = 2), `locked` UInt8,
↪  `assignee` LowCardinality(String), `assignees`
↪  Array(LowCardinality(String)), `comments` UInt32,
↪  `author_association` Enum8('NONE' = 0, 'CONTRIBUTOR' = 1,
↪  'OWNER' = 2, 'COLLABORATOR' = 3, 'MEMBER' = 4, 'MANNEQUIN' =
↪  5), `closed_at` DateTime, `merged_at` DateTime,
↪  `merge_commit_sha` String, `requested_reviewers`
↪  Array(LowCardinality(String)), `requested_teams`
↪  Array(LowCardinality(String)), `head_ref`
↪  LowCardinality(String), `head_sha` String, `base_ref`
↪  LowCardinality(String), `base_sha` String, `merged` UInt8,
↪  `mergeable` UInt8, `rebaseable` UInt8, `mergeable_state`
↪  Enum8('unknown' = 0, 'dirty' = 1, 'clean' = 2, 'unstable' = 3,
↪  'draft' = 4), `merged_by` LowCardinality(String),
↪  `review_comments` UInt32, `maintainer_can_modify` UInt8,
↪  `commits` UInt32, `additions` UInt32, `deletions` UInt32,
↪  `changed_files` UInt32, `diff_hunk` String, `original_position`
↪  UInt32, `commit_id` String, `original_commit_id` String,
↪  `push_size` UInt32, `push_distinct_size` UInt32, `member_login`
↪  LowCardinality(String), `release_tag_name` String,
↪  `release_name` String, `review_state` Enum8('none' = 0,
↪  'approved' = 1, 'changes_requested' = 2, 'commented' = 3,
↪  'dismissed' = 4, 'pending' = 5)) ENGINE = MergeTree ORDER BY
↪  (event_type, repo_name, created_at) SETTINGS index_granularity
↪  = 8192
```

```
# 2022/05/14 11:03:48.546649  warn can't create table
 ↪  'default.github_events4': code: 57, message: Directory for
 ↪  table data store/6e6/6e6a4e02-a9fa-42ad-a0ce-af77e49a2aee/
 ↪  already exists, will try again backup=2022-05-11T07-25-16
 ↪  operation=restore
# 2022/05/14 11:03:48.546680  info CREATE DATABASE IF NOT EXISTS
 ↪  `default`
```

```
# 2022/05/14 11:03:48.548692  info CREATE TABLE
↪  default.github_events5 UUID
↪  '697e13ba-37ac-44ac-a7fc-01b0081b4670' (`file_time` DateTime,
↪  `event_type` Enum8('CommitCommentEvent' = 1, 'CreateEvent' = 2,
↪  'DeleteEvent' = 3, 'ForkEvent' = 4, 'GollumEvent' = 5,
↪  'IssueCommentEvent' = 6, 'IssuesEvent' = 7, 'MemberEvent' = 8,
↪  'PublicEvent' = 9, 'PullRequestEvent' = 10,
↪  'PullRequestReviewCommentEvent' = 11, 'PushEvent' = 12,
↪  'ReleaseEvent' = 13, 'SponsorshipEvent' = 14, 'WatchEvent' =
↪  15, 'GistEvent' = 16, 'FollowEvent' = 17, 'DownloadEvent' = 18,
↪  'PullRequestReviewEvent' = 19, 'ForkApplyEvent' = 20, 'Event' =
↪  21, 'TeamAddEvent' = 22), `actor_login` LowCardinality(String),
↪  `repo_name` LowCardinality(String), `created_at` DateTime,
↪  `updated_at` DateTime, `action` Enum8('none' = 0, 'created' =
↪  1, 'added' = 2, 'edited' = 3, 'deleted' = 4, 'opened' = 5,
↪  'closed' = 6, 'reopened' = 7, 'assigned' = 8, 'unassigned' = 9,
↪  'labeled' = 10, 'unlabeled' = 11, 'review_requested' = 12,
↪  'review_request_removed' = 13, 'synchronize' = 14, 'started' =
↪  15, 'published' = 16, 'update' = 17, 'create' = 18, 'fork' =
↪  19, 'merged' = 20), `comment_id` UInt64, `body` String, `path`
↪  String, `position` Int32, `line` Int32, `ref`
↪  LowCardinality(String), `ref_type` Enum8('none' = 0, 'branch' =
↪  1, 'tag' = 2, 'repository' = 3, 'unknown' = 4),
↪  `creator_user_login` LowCardinality(String), `number` UInt32,
↪  `title` String, `labels` Array(LowCardinality(String)), `state`
↪  Enum8('none' = 0, 'open' = 1, 'closed' = 2), `locked` UInt8,
↪  `assignee` LowCardinality(String), `assignees`
↪  Array(LowCardinality(String)), `comments` UInt32,
↪  `author_association` Enum8('NONE' = 0, 'CONTRIBUTOR' = 1,
↪  'OWNER' = 2, 'COLLABORATOR' = 3, 'MEMBER' = 4, 'MANNEQUIN' =
↪  5), `closed_at` DateTime, `merged_at` DateTime,
↪  `merge_commit_sha` String, `requested_reviewers`
↪  Array(LowCardinality(String)), `requested_teams`
↪  Array(LowCardinality(String)), `head_ref`
↪  LowCardinality(String), `head_sha` String, `base_ref`
↪  LowCardinality(String), `base_sha` String, `merged` UInt8,
↪  `mergeable` UInt8, `rebaseable` UInt8, `mergeable_state`
↪  Enum8('unknown' = 0, 'dirty' = 1, 'clean' = 2, 'unstable' = 3,
↪  'draft' = 4), `merged_by` LowCardinality(String),
↪  `review_comments` UInt32, `maintainer_can_modify` UInt8,
↪  `commits` UInt32, `additions` UInt32, `deletions` UInt32,
↪  `changed_files` UInt32, `diff_hunk` String, `original_position`
↪  UInt32, `commit_id` String, `original_commit_id` String,
↪  `push_size` UInt32, `push_distinct_size` UInt32, `member_login`
↪  LowCardinality(String), `release_tag_name` String,
↪  `release_name` String, `review_state` Enum8('none' = 0,
↪  'approved' = 1, 'changes_requested' = 2, 'commented' = 3,
↪  'dismissed' = 4, 'pending' = 5)) ENGINE = MergeTree ORDER BY
↪  (event_type, repo_name, created_at) SETTINGS index_granularity
↪  = 8192
```

```
# 2022/05/14 11:03:48.552764 error can't create table
 ↪  `default`.`github_events5`: code: 57, message: Directory for
 ↪  table data store/697/697e13ba-37ac-44ac-a7fc-01b0081b4670/
 ↪  already exists after 5 times, please check your schema
 ↪  dependencies
#
# real        0m0.089s
# user        0m0.092s
# sys        0m0.022s
```

The restore fails because some directories apparently already exist. The VM was restarted.

New attempt at restoring from a local backup:

```
time ./clickhouse-backup restore 2022-05-11T07-25-16
# 2022/05/14 11:08:27.188612  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/14 11:08:27.191239  info SELECT * FROM system.disks;
# 2022/05/14 11:08:27.193861  info CREATE DATABASE IF NOT EXISTS
 ↪  default
# ENGINE = Atomic
# 2022/05/14 11:08:27.195504  info SELECT engine FROM
 ↪  system.databases WHERE name = 'default'
# 2022/05/14 11:08:27.198077  info DROP TABLE IF EXISTS
 ↪  `default`.`github_events1` NO DELAY
# 2022/05/14 11:08:27.200105  info SELECT engine FROM
 ↪  system.databases WHERE name = 'default'
# 2022/05/14 11:08:27.203430  info DROP TABLE IF EXISTS
 ↪  `default`.`github_events2` NO DELAY
# 2022/05/14 11:08:27.205062  info SELECT engine FROM
 ↪  system.databases WHERE name = 'default'
# 2022/05/14 11:08:27.207140  info DROP TABLE IF EXISTS
 ↪  `default`.`github_events3` NO DELAY
# 2022/05/14 11:08:27.209092  info SELECT engine FROM
 ↪  system.databases WHERE name = 'default'
# 2022/05/14 11:08:27.211382  info DROP TABLE IF EXISTS
 ↪  `default`.`github_events4` NO DELAY
# 2022/05/14 11:08:27.212707  info SELECT engine FROM
 ↪  system.databases WHERE name = 'default'
# 2022/05/14 11:08:27.215804  info DROP TABLE IF EXISTS
 ↪  `default`.`github_events5` NO DELAY
# 2022/05/14 11:08:27.217320  info CREATE DATABASE IF NOT EXISTS
 ↪  `default`
```

```
# 2022/05/14 11:08:27.219084  info CREATE TABLE
↪  default.github_events1 UUID
↪  '1711c114-1286-4a0b-8f9a-11fbc6d35b89' (`file_time` DateTime,
↪  `event_type` Enum8('CommitCommentEvent' = 1, 'CreateEvent' = 2,
↪  'DeleteEvent' = 3, 'ForkEvent' = 4, 'GollumEvent' = 5,
↪  'IssueCommentEvent' = 6, 'IssuesEvent' = 7, 'MemberEvent' = 8,
↪  'PublicEvent' = 9, 'PullRequestEvent' = 10,
↪  'PullRequestReviewCommentEvent' = 11, 'PushEvent' = 12,
↪  'ReleaseEvent' = 13, 'SponsorshipEvent' = 14, 'WatchEvent' =
↪  15, 'GistEvent' = 16, 'FollowEvent' = 17, 'DownloadEvent' = 18,
↪  'PullRequestReviewEvent' = 19, 'ForkApplyEvent' = 20, 'Event' =
↪  21, 'TeamAddEvent' = 22), `actor_login` LowCardinality(String),
↪  `repo_name` LowCardinality(String), `created_at` DateTime,
↪  `updated_at` DateTime, `action` Enum8('none' = 0, 'created' =
↪  1, 'added' = 2, 'edited' = 3, 'deleted' = 4, 'opened' = 5,
↪  'closed' = 6, 'reopened' = 7, 'assigned' = 8, 'unassigned' = 9,
↪  'labeled' = 10, 'unlabeled' = 11, 'review_requested' = 12,
↪  'review_request_removed' = 13, 'synchronize' = 14, 'started' =
↪  15, 'published' = 16, 'update' = 17, 'create' = 18, 'fork' =
↪  19, 'merged' = 20), `comment_id` UInt64, `body` String, `path`
↪  String, `position` Int32, `line` Int32, `ref`
↪  LowCardinality(String), `ref_type` Enum8('none' = 0, 'branch' =
↪  1, 'tag' = 2, 'repository' = 3, 'unknown' = 4),
↪  `creator_user_login` LowCardinality(String), `number` UInt32,
↪  `title` String, `labels` Array(LowCardinality(String)), `state`
↪  Enum8('none' = 0, 'open' = 1, 'closed' = 2), `locked` UInt8,
↪  `assignee` LowCardinality(String), `assignees`
↪  Array(LowCardinality(String)), `comments` UInt32,
↪  `author_association` Enum8('NONE' = 0, 'CONTRIBUTOR' = 1,
↪  'OWNER' = 2, 'COLLABORATOR' = 3, 'MEMBER' = 4, 'MANNEQUIN' =
↪  5), `closed_at` DateTime, `merged_at` DateTime,
↪  `merge_commit_sha` String, `requested_reviewers`
↪  Array(LowCardinality(String)), `requested_teams`
↪  Array(LowCardinality(String)), `head_ref`
↪  LowCardinality(String), `head_sha` String, `base_ref`
↪  LowCardinality(String), `base_sha` String, `merged` UInt8,
↪  `mergeable` UInt8, `rebaseable` UInt8, `mergeable_state`
↪  Enum8('unknown' = 0, 'dirty' = 1, 'clean' = 2, 'unstable' = 3,
↪  'draft' = 4), `merged_by` LowCardinality(String),
↪  `review_comments` UInt32, `maintainer_can_modify` UInt8,
↪  `commits` UInt32, `additions` UInt32, `deletions` UInt32,
↪  `changed_files` UInt32, `diff_hunk` String, `original_position`
↪  UInt32, `commit_id` String, `original_commit_id` String,
↪  `push_size` UInt32, `push_distinct_size` UInt32, `member_login`
↪  LowCardinality(String), `release_tag_name` String,
↪  `release_name` String, `review_state` Enum8('none' = 0,
↪  'approved' = 1, 'changes_requested' = 2, 'commented' = 3,
↪  'dismissed' = 4, 'pending' = 5)) ENGINE = MergeTree ORDER BY
↪  (event_type, repo_name, created_at) SETTINGS index_granularity
↪  = 8192
```

```
# 2022/05/14 11:08:27.234314  info CREATE DATABASE IF NOT EXISTS
 ↪  `default`
```

```
# 2022/05/14 11:08:27.236603  info CREATE TABLE
↪  default.github_events2 UUID
↪  '1991bd38-adaf-4691-aa49-c04b27f61485' (`file_time` DateTime,
↪  `event_type` Enum8('CommitCommentEvent' = 1, 'CreateEvent' = 2,
↪  'DeleteEvent' = 3, 'ForkEvent' = 4, 'GollumEvent' = 5,
↪  'IssueCommentEvent' = 6, 'IssuesEvent' = 7, 'MemberEvent' = 8,
↪  'PublicEvent' = 9, 'PullRequestEvent' = 10,
↪  'PullRequestReviewCommentEvent' = 11, 'PushEvent' = 12,
↪  'ReleaseEvent' = 13, 'SponsorshipEvent' = 14, 'WatchEvent' =
↪  15, 'GistEvent' = 16, 'FollowEvent' = 17, 'DownloadEvent' = 18,
↪  'PullRequestReviewEvent' = 19, 'ForkApplyEvent' = 20, 'Event' =
↪  21, 'TeamAddEvent' = 22), `actor_login` LowCardinality(String),
↪  `repo_name` LowCardinality(String), `created_at` DateTime,
↪  `updated_at` DateTime, `action` Enum8('none' = 0, 'created' =
↪  1, 'added' = 2, 'edited' = 3, 'deleted' = 4, 'opened' = 5,
↪  'closed' = 6, 'reopened' = 7, 'assigned' = 8, 'unassigned' = 9,
↪  'labeled' = 10, 'unlabeled' = 11, 'review_requested' = 12,
↪  'review_request_removed' = 13, 'synchronize' = 14, 'started' =
↪  15, 'published' = 16, 'update' = 17, 'create' = 18, 'fork' =
↪  19, 'merged' = 20), `comment_id` UInt64, `body` String, `path`
↪  String, `position` Int32, `line` Int32, `ref`
↪  LowCardinality(String), `ref_type` Enum8('none' = 0, 'branch' =
↪  1, 'tag' = 2, 'repository' = 3, 'unknown' = 4),
↪  `creator_user_login` LowCardinality(String), `number` UInt32,
↪  `title` String, `labels` Array(LowCardinality(String)), `state`
↪  Enum8('none' = 0, 'open' = 1, 'closed' = 2), `locked` UInt8,
↪  `assignee` LowCardinality(String), `assignees`
↪  Array(LowCardinality(String)), `comments` UInt32,
↪  `author_association` Enum8('NONE' = 0, 'CONTRIBUTOR' = 1,
↪  'OWNER' = 2, 'COLLABORATOR' = 3, 'MEMBER' = 4, 'MANNEQUIN' =
↪  5), `closed_at` DateTime, `merged_at` DateTime,
↪  `merge_commit_sha` String, `requested_reviewers`
↪  Array(LowCardinality(String)), `requested_teams`
↪  Array(LowCardinality(String)), `head_ref`
↪  LowCardinality(String), `head_sha` String, `base_ref`
↪  LowCardinality(String), `base_sha` String, `merged` UInt8,
↪  `mergeable` UInt8, `rebaseable` UInt8, `mergeable_state`
↪  Enum8('unknown' = 0, 'dirty' = 1, 'clean' = 2, 'unstable' = 3,
↪  'draft' = 4), `merged_by` LowCardinality(String),
↪  `review_comments` UInt32, `maintainer_can_modify` UInt8,
↪  `commits` UInt32, `additions` UInt32, `deletions` UInt32,
↪  `changed_files` UInt32, `diff_hunk` String, `original_position`
↪  UInt32, `commit_id` String, `original_commit_id` String,
↪  `push_size` UInt32, `push_distinct_size` UInt32, `member_login`
↪  LowCardinality(String), `release_tag_name` String,
↪  `release_name` String, `review_state` Enum8('none' = 0,
↪  'approved' = 1, 'changes_requested' = 2, 'commented' = 3,
↪  'dismissed' = 4, 'pending' = 5)) ENGINE = MergeTree ORDER BY
↪  (event_type, repo_name, created_at) SETTINGS index_granularity
↪  = 8192
```

```
# 2022/05/14 11:08:27.241784  info CREATE DATABASE IF NOT EXISTS
 ↪  `default`
```

```
# 2022/05/14 11:08:27.243580  info CREATE TABLE
↪  default.github_events3 UUID
↪  'c403ed92-213a-4245-b98b-408d6fed9377' (`file_time` DateTime,
↪  `event_type` Enum8('CommitCommentEvent' = 1, 'CreateEvent' = 2,
↪  'DeleteEvent' = 3, 'ForkEvent' = 4, 'GollumEvent' = 5,
↪  'IssueCommentEvent' = 6, 'IssuesEvent' = 7, 'MemberEvent' = 8,
↪  'PublicEvent' = 9, 'PullRequestEvent' = 10,
↪  'PullRequestReviewCommentEvent' = 11, 'PushEvent' = 12,
↪  'ReleaseEvent' = 13, 'SponsorshipEvent' = 14, 'WatchEvent' =
↪  15, 'GistEvent' = 16, 'FollowEvent' = 17, 'DownloadEvent' = 18,
↪  'PullRequestReviewEvent' = 19, 'ForkApplyEvent' = 20, 'Event' =
↪  21, 'TeamAddEvent' = 22), `actor_login` LowCardinality(String),
↪  `repo_name` LowCardinality(String), `created_at` DateTime,
↪  `updated_at` DateTime, `action` Enum8('none' = 0, 'created' =
↪  1, 'added' = 2, 'edited' = 3, 'deleted' = 4, 'opened' = 5,
↪  'closed' = 6, 'reopened' = 7, 'assigned' = 8, 'unassigned' = 9,
↪  'labeled' = 10, 'unlabeled' = 11, 'review_requested' = 12,
↪  'review_request_removed' = 13, 'synchronize' = 14, 'started' =
↪  15, 'published' = 16, 'update' = 17, 'create' = 18, 'fork' =
↪  19, 'merged' = 20), `comment_id` UInt64, `body` String, `path`
↪  String, `position` Int32, `line` Int32, `ref`
↪  LowCardinality(String), `ref_type` Enum8('none' = 0, 'branch' =
↪  1, 'tag' = 2, 'repository' = 3, 'unknown' = 4),
↪  `creator_user_login` LowCardinality(String), `number` UInt32,
↪  `title` String, `labels` Array(LowCardinality(String)), `state`
↪  Enum8('none' = 0, 'open' = 1, 'closed' = 2), `locked` UInt8,
↪  `assignee` LowCardinality(String), `assignees`
↪  Array(LowCardinality(String)), `comments` UInt32,
↪  `author_association` Enum8('NONE' = 0, 'CONTRIBUTOR' = 1,
↪  'OWNER' = 2, 'COLLABORATOR' = 3, 'MEMBER' = 4, 'MANNEQUIN' =
↪  5), `closed_at` DateTime, `merged_at` DateTime,
↪  `merge_commit_sha` String, `requested_reviewers`
↪  Array(LowCardinality(String)), `requested_teams`
↪  Array(LowCardinality(String)), `head_ref`
↪  LowCardinality(String), `head_sha` String, `base_ref`
↪  LowCardinality(String), `base_sha` String, `merged` UInt8,
↪  `mergeable` UInt8, `rebaseable` UInt8, `mergeable_state`
↪  Enum8('unknown' = 0, 'dirty' = 1, 'clean' = 2, 'unstable' = 3,
↪  'draft' = 4), `merged_by` LowCardinality(String),
↪  `review_comments` UInt32, `maintainer_can_modify` UInt8,
↪  `commits` UInt32, `additions` UInt32, `deletions` UInt32,
↪  `changed_files` UInt32, `diff_hunk` String, `original_position`
↪  UInt32, `commit_id` String, `original_commit_id` String,
↪  `push_size` UInt32, `push_distinct_size` UInt32, `member_login`
↪  LowCardinality(String), `release_tag_name` String,
↪  `release_name` String, `review_state` Enum8('none' = 0,
↪  'approved' = 1, 'changes_requested' = 2, 'commented' = 3,
↪  'dismissed' = 4, 'pending' = 5)) ENGINE = MergeTree ORDER BY
↪  (event_type, repo_name, created_at) SETTINGS index_granularity
↪  = 8192
```

```
# 2022/05/14 11:08:27.247996  info CREATE DATABASE IF NOT EXISTS
 ↪  `default`
```

```
# 2022/05/14 11:08:27.249709  info CREATE TABLE
↪  default.github_events4 UUID
↪  '6e6a4e02-a9fa-42ad-a0ce-af77e49a2aee' (`file_time` DateTime,
↪  `event_type` Enum8('CommitCommentEvent' = 1, 'CreateEvent' = 2,
↪  'DeleteEvent' = 3, 'ForkEvent' = 4, 'GollumEvent' = 5,
↪  'IssueCommentEvent' = 6, 'IssuesEvent' = 7, 'MemberEvent' = 8,
↪  'PublicEvent' = 9, 'PullRequestEvent' = 10,
↪  'PullRequestReviewCommentEvent' = 11, 'PushEvent' = 12,
↪  'ReleaseEvent' = 13, 'SponsorshipEvent' = 14, 'WatchEvent' =
↪  15, 'GistEvent' = 16, 'FollowEvent' = 17, 'DownloadEvent' = 18,
↪  'PullRequestReviewEvent' = 19, 'ForkApplyEvent' = 20, 'Event' =
↪  21, 'TeamAddEvent' = 22), `actor_login` LowCardinality(String),
↪  `repo_name` LowCardinality(String), `created_at` DateTime,
↪  `updated_at` DateTime, `action` Enum8('none' = 0, 'created' =
↪  1, 'added' = 2, 'edited' = 3, 'deleted' = 4, 'opened' = 5,
↪  'closed' = 6, 'reopened' = 7, 'assigned' = 8, 'unassigned' = 9,
↪  'labeled' = 10, 'unlabeled' = 11, 'review_requested' = 12,
↪  'review_request_removed' = 13, 'synchronize' = 14, 'started' =
↪  15, 'published' = 16, 'update' = 17, 'create' = 18, 'fork' =
↪  19, 'merged' = 20), `comment_id` UInt64, `body` String, `path`
↪  String, `position` Int32, `line` Int32, `ref`
↪  LowCardinality(String), `ref_type` Enum8('none' = 0, 'branch' =
↪  1, 'tag' = 2, 'repository' = 3, 'unknown' = 4),
↪  `creator_user_login` LowCardinality(String), `number` UInt32,
↪  `title` String, `labels` Array(LowCardinality(String)), `state`
↪  Enum8('none' = 0, 'open' = 1, 'closed' = 2), `locked` UInt8,
↪  `assignee` LowCardinality(String), `assignees`
↪  Array(LowCardinality(String)), `comments` UInt32,
↪  `author_association` Enum8('NONE' = 0, 'CONTRIBUTOR' = 1,
↪  'OWNER' = 2, 'COLLABORATOR' = 3, 'MEMBER' = 4, 'MANNEQUIN' =
↪  5), `closed_at` DateTime, `merged_at` DateTime,
↪  `merge_commit_sha` String, `requested_reviewers`
↪  Array(LowCardinality(String)), `requested_teams`
↪  Array(LowCardinality(String)), `head_ref`
↪  LowCardinality(String), `head_sha` String, `base_ref`
↪  LowCardinality(String), `base_sha` String, `merged` UInt8,
↪  `mergeable` UInt8, `rebaseable` UInt8, `mergeable_state`
↪  Enum8('unknown' = 0, 'dirty' = 1, 'clean' = 2, 'unstable' = 3,
↪  'draft' = 4), `merged_by` LowCardinality(String),
↪  `review_comments` UInt32, `maintainer_can_modify` UInt8,
↪  `commits` UInt32, `additions` UInt32, `deletions` UInt32,
↪  `changed_files` UInt32, `diff_hunk` String, `original_position`
↪  UInt32, `commit_id` String, `original_commit_id` String,
↪  `push_size` UInt32, `push_distinct_size` UInt32, `member_login`
↪  LowCardinality(String), `release_tag_name` String,
↪  `release_name` String, `review_state` Enum8('none' = 0,
↪  'approved' = 1, 'changes_requested' = 2, 'commented' = 3,
↪  'dismissed' = 4, 'pending' = 5)) ENGINE = MergeTree ORDER BY
↪  (event_type, repo_name, created_at) SETTINGS index_granularity
↪  = 8192
```

```
# 2022/05/14 11:08:27.254263  info CREATE DATABASE IF NOT EXISTS
 ↪  `default`
```

```
# 2022/05/14 11:08:27.256082  info CREATE TABLE
↪  default.github_events5 UUID
↪  '697e13ba-37ac-44ac-a7fc-01b0081b4670' (`file_time` DateTime,
↪  `event_type` Enum8('CommitCommentEvent' = 1, 'CreateEvent' = 2,
↪  'DeleteEvent' = 3, 'ForkEvent' = 4, 'GollumEvent' = 5,
↪  'IssueCommentEvent' = 6, 'IssuesEvent' = 7, 'MemberEvent' = 8,
↪  'PublicEvent' = 9, 'PullRequestEvent' = 10,
↪  'PullRequestReviewCommentEvent' = 11, 'PushEvent' = 12,
↪  'ReleaseEvent' = 13, 'SponsorshipEvent' = 14, 'WatchEvent' =
↪  15, 'GistEvent' = 16, 'FollowEvent' = 17, 'DownloadEvent' = 18,
↪  'PullRequestReviewEvent' = 19, 'ForkApplyEvent' = 20, 'Event' =
↪  21, 'TeamAddEvent' = 22), `actor_login` LowCardinality(String),
↪  `repo_name` LowCardinality(String), `created_at` DateTime,
↪  `updated_at` DateTime, `action` Enum8('none' = 0, 'created' =
↪  1, 'added' = 2, 'edited' = 3, 'deleted' = 4, 'opened' = 5,
↪  'closed' = 6, 'reopened' = 7, 'assigned' = 8, 'unassigned' = 9,
↪  'labeled' = 10, 'unlabeled' = 11, 'review_requested' = 12,
↪  'review_request_removed' = 13, 'synchronize' = 14, 'started' =
↪  15, 'published' = 16, 'update' = 17, 'create' = 18, 'fork' =
↪  19, 'merged' = 20), `comment_id` UInt64, `body` String, `path`
↪  String, `position` Int32, `line` Int32, `ref`
↪  LowCardinality(String), `ref_type` Enum8('none' = 0, 'branch' =
↪  1, 'tag' = 2, 'repository' = 3, 'unknown' = 4),
↪  `creator_user_login` LowCardinality(String), `number` UInt32,
↪  `title` String, `labels` Array(LowCardinality(String)), `state`
↪  Enum8('none' = 0, 'open' = 1, 'closed' = 2), `locked` UInt8,
↪  `assignee` LowCardinality(String), `assignees`
↪  Array(LowCardinality(String)), `comments` UInt32,
↪  `author_association` Enum8('NONE' = 0, 'CONTRIBUTOR' = 1,
↪  'OWNER' = 2, 'COLLABORATOR' = 3, 'MEMBER' = 4, 'MANNEQUIN' =
↪  5), `closed_at` DateTime, `merged_at` DateTime,
↪  `merge_commit_sha` String, `requested_reviewers`
↪  Array(LowCardinality(String)), `requested_teams`
↪  Array(LowCardinality(String)), `head_ref`
↪  LowCardinality(String), `head_sha` String, `base_ref`
↪  LowCardinality(String), `base_sha` String, `merged` UInt8,
↪  `mergeable` UInt8, `rebaseable` UInt8, `mergeable_state`
↪  Enum8('unknown' = 0, 'dirty' = 1, 'clean' = 2, 'unstable' = 3,
↪  'draft' = 4), `merged_by` LowCardinality(String),
↪  `review_comments` UInt32, `maintainer_can_modify` UInt8,
↪  `commits` UInt32, `additions` UInt32, `deletions` UInt32,
↪  `changed_files` UInt32, `diff_hunk` String, `original_position`
↪  UInt32, `commit_id` String, `original_commit_id` String,
↪  `push_size` UInt32, `push_distinct_size` UInt32, `member_login`
↪  LowCardinality(String), `release_tag_name` String,
↪  `release_name` String, `review_state` Enum8('none' = 0,
↪  'approved' = 1, 'changes_requested' = 2, 'commented' = 3,
↪  'dismissed' = 4, 'pending' = 5)) ENGINE = MergeTree ORDER BY
↪  (event_type, repo_name, created_at) SETTINGS index_granularity
↪  = 8192
```

```
# 2022/05/14 11:08:27.262311  info SELECT count() FROM
 ↪  system.settings WHERE name =
 ↪  'show_table_uuid_in_table_create_query_if_not_nil'
# 2022/05/14 11:08:27.265255  info SELECT name FROM
 ↪  system.databases WHERE engine IN ('MySQL','PostgreSQL')
# 2022/05/14 11:08:27.267740  info
#                 SELECT
#                         countIf(name='data_path')
 ↪  is_data_path_present,
#                         countIf(name='data_paths')
 ↪  is_data_paths_present,
#                         countIf(name='uuid') is_uuid_present,
#                         countIf(name='create_table_query')
 ↪  is_create_table_query_present,
#                         countIf(name='total_bytes')
 ↪  is_total_bytes_present
#                 FROM system.columns WHERE database='system' AND
 ↪  table='tables'
#
# 2022/05/14 11:08:27.271152  info SELECT database, name, engine ,
 ↪  data_paths , uuid , create_table_query , coalesce(total_bytes,
 ↪  0) AS total_bytes   FROM system.tables WHERE is_temporary = 0
 ↪  SETTINGS show_table_uuid_in_table_create_query_if_not_nil=1
# 2022/05/14 11:08:27.279692  info SELECT sum(bytes_on_disk) as
 ↪  size FROM system.parts WHERE database='default' AND
 ↪  table='github_events1' GROUP BY database, table
# 2022/05/14 11:08:27.282537  info SELECT sum(bytes_on_disk) as
 ↪  size FROM system.parts WHERE database='default' AND
 ↪  table='github_events2' GROUP BY database, table
# 2022/05/14 11:08:27.285404  info SELECT sum(bytes_on_disk) as
 ↪  size FROM system.parts WHERE database='default' AND
 ↪  table='github_events3' GROUP BY database, table
# 2022/05/14 11:08:27.288135  info SELECT sum(bytes_on_disk) as
 ↪  size FROM system.parts WHERE database='default' AND
 ↪  table='github_events4' GROUP BY database, table
# 2022/05/14 11:08:27.309797  info SELECT sum(bytes_on_disk) as
 ↪  size FROM system.parts WHERE database='default' AND
 ↪  table='github_events5' GROUP BY database, table
# 2022/05/14 11:08:27.336982  info ALTER TABLE
 ↪  `default`.`github_events1` ATTACH PART 'all_1610_2464_4'
# 2022/05/14 11:08:27.350624  info ALTER TABLE
 ↪  `default`.`github_events1` ATTACH PART 'all_1_673_4'
# 2022/05/14 11:08:27.384390  info ALTER TABLE
 ↪  `default`.`github_events1` ATTACH PART 'all_2465_3440_4'
```

```
# 2022/05/14 11:08:27.396689  info ALTER TABLE
 ↪  `default`.`github_events1` ATTACH PART 'all_3441_4218_4'
# 2022/05/14 11:08:28.992399  info ALTER TABLE
 ↪  `default`.`github_events1` ATTACH PART 'all_4219_5004_4'
# 2022/05/14 11:08:29.030771  info ALTER TABLE
 ↪  `default`.`github_events1` ATTACH PART 'all_5005_5175_3'
# 2022/05/14 11:08:29.040817  info ALTER TABLE
 ↪  `default`.`github_events1` ATTACH PART 'all_5176_5206_2'
# 2022/05/14 11:08:29.045172  info ALTER TABLE
 ↪  `default`.`github_events1` ATTACH PART 'all_5207_5213_1'
# 2022/05/14 11:08:29.049166  info ALTER TABLE
 ↪  `default`.`github_events1` ATTACH PART 'all_5214_5219_1'
# 2022/05/14 11:08:29.052533  info ALTER TABLE
 ↪  `default`.`github_events1` ATTACH PART 'all_5220_5225_1'
# 2022/05/14 11:08:29.056002  info ALTER TABLE
 ↪  `default`.`github_events1` ATTACH PART 'all_5226_5231_1'
# 2022/05/14 11:08:29.059550  info ALTER TABLE
 ↪  `default`.`github_events1` ATTACH PART 'all_674_1609_4'
# 2022/05/14 11:08:29.075561  info done
 ↪  backup=2022-05-11T07-25-16 operation=restore
 ↪  table=default.github_events1
# 2022/05/14 11:08:29.096382  info ALTER TABLE
 ↪  `default`.`github_events2` ATTACH PART 'all_1612_2460_4'
# 2022/05/14 11:08:29.110102  info ALTER TABLE
 ↪  `default`.`github_events2` ATTACH PART 'all_1_700_4'
# 2022/05/14 11:08:29.141549  info ALTER TABLE
 ↪  `default`.`github_events2` ATTACH PART 'all_2461_3426_4'
# 2022/05/14 11:08:29.154434  info ALTER TABLE
 ↪  `default`.`github_events2` ATTACH PART 'all_3427_4340_4'
# 2022/05/14 11:08:29.197997  info ALTER TABLE
 ↪  `default`.`github_events2` ATTACH PART 'all_4341_5037_4'
# 2022/05/14 11:08:29.232228  info ALTER TABLE
 ↪  `default`.`github_events2` ATTACH PART 'all_5038_5070_2'
# 2022/05/14 11:08:29.237555  info ALTER TABLE
 ↪  `default`.`github_events2` ATTACH PART 'all_5071_5223_3'
# 2022/05/14 11:08:29.246467  info ALTER TABLE
 ↪  `default`.`github_events2` ATTACH PART 'all_5224_5229_1'
# 2022/05/14 11:08:29.250524  info ALTER TABLE
 ↪  `default`.`github_events2` ATTACH PART 'all_5230_5230_0'
# 2022/05/14 11:08:29.254688  info ALTER TABLE
 ↪  `default`.`github_events2` ATTACH PART 'all_5231_5231_0'
# 2022/05/14 11:08:29.258458  info ALTER TABLE
 ↪  `default`.`github_events2` ATTACH PART 'all_701_1611_4'
```

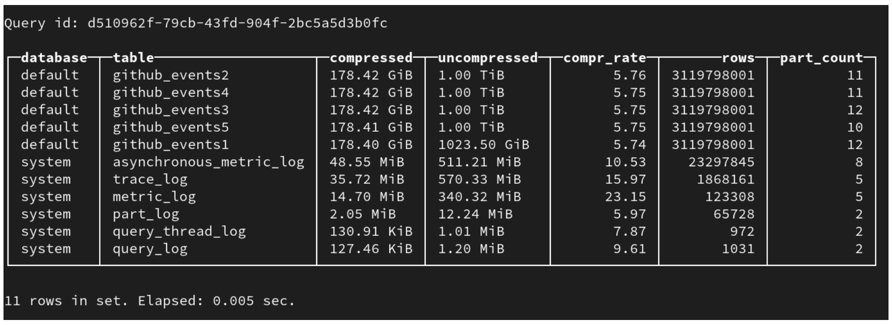
```
# 2022/05/14 11:08:29.273429  info done
 ↪  backup=2022-05-11T07-25-16 operation=restore
 ↪  table=default.github_events2
# 2022/05/14 11:08:29.297371  info ALTER TABLE
 ↪  `default`.`github_events3` ATTACH PART 'all_1600_2557_4'
# 2022/05/14 11:08:29.311401  info ALTER TABLE
 ↪  `default`.`github_events3` ATTACH PART 'all_1_722_4'
# 2022/05/14 11:08:29.342883  info ALTER TABLE
 ↪  `default`.`github_events3` ATTACH PART 'all_2558_3428_4'
# 2022/05/14 11:08:29.355298  info ALTER TABLE
 ↪  `default`.`github_events3` ATTACH PART 'all_3429_4208_4'
# 2022/05/14 11:08:29.392131  info ALTER TABLE
 ↪  `default`.`github_events3` ATTACH PART 'all_4209_5023_4'
# 2022/05/14 11:08:29.432268  info ALTER TABLE
 ↪  `default`.`github_events3` ATTACH PART 'all_5024_5204_3'
# 2022/05/14 11:08:29.444730  info ALTER TABLE
 ↪  `default`.`github_events3` ATTACH PART 'all_5205_5212_1'
# 2022/05/14 11:08:29.449364  info ALTER TABLE
 ↪  `default`.`github_events3` ATTACH PART 'all_5213_5218_1'
# 2022/05/14 11:08:29.453913  info ALTER TABLE
 ↪  `default`.`github_events3` ATTACH PART 'all_5219_5224_1'
# 2022/05/14 11:08:29.457986  info ALTER TABLE
 ↪  `default`.`github_events3` ATTACH PART 'all_5225_5230_1'
# 2022/05/14 11:08:29.461761  info ALTER TABLE
 ↪  `default`.`github_events3` ATTACH PART 'all_5231_5231_0'
# 2022/05/14 11:08:29.464290  info ALTER TABLE
 ↪  `default`.`github_events3` ATTACH PART 'all_723_1599_4'
# 2022/05/14 11:08:29.479531  info done
 ↪  backup=2022-05-11T07-25-16 operation=restore
 ↪  table=default.github_events3
# 2022/05/14 11:08:29.500687  info ALTER TABLE
 ↪  `default`.`github_events4` ATTACH PART 'all_1899_2864_4'
# 2022/05/14 11:08:29.512288  info ALTER TABLE
 ↪  `default`.`github_events4` ATTACH PART 'all_1_1898_5'
# 2022/05/14 11:08:29.559629  info ALTER TABLE
 ↪  `default`.`github_events4` ATTACH PART 'all_2865_4197_5'
# 2022/05/14 11:08:29.598962  info ALTER TABLE
 ↪  `default`.`github_events4` ATTACH PART 'all_4198_5026_4'
# 2022/05/14 11:08:29.636805  info ALTER TABLE
 ↪  `default`.`github_events4` ATTACH PART 'all_5027_5197_3'
# 2022/05/14 11:08:29.648912  info ALTER TABLE
 ↪  `default`.`github_events4` ATTACH PART 'all_5198_5204_1'
# 2022/05/14 11:08:29.652660  info ALTER TABLE
 ↪  `default`.`github_events4` ATTACH PART 'all_5205_5212_1'
```

```
# 2022/05/14 11:08:29.658078  info ALTER TABLE
 ↪  `default`.`github_events4` ATTACH PART 'all_5213_5218_1'
# 2022/05/14 11:08:29.660785  info ALTER TABLE
 ↪  `default`.`github_events4` ATTACH PART 'all_5219_5224_1'
# 2022/05/14 11:08:29.664747  info ALTER TABLE
 ↪  `default`.`github_events4` ATTACH PART 'all_5225_5230_1'
# 2022/05/14 11:08:29.668866  info ALTER TABLE
 ↪  `default`.`github_events4` ATTACH PART 'all_5231_5231_0'
# 2022/05/14 11:08:29.673154  info done
 ↪  backup=2022-05-11T07-25-16 operation=restore
 ↪  table=default.github_events4
# 2022/05/14 11:08:29.692356  info ALTER TABLE
 ↪  `default`.`github_events5` ATTACH PART 'all_1761_2815_4'
# 2022/05/14 11:08:29.705300  info ALTER TABLE
 ↪  `default`.`github_events5` ATTACH PART 'all_1_686_4'
# 2022/05/14 11:08:29.735563  info ALTER TABLE
 ↪  `default`.`github_events5` ATTACH PART 'all_2816_4010_5'
# 2022/05/14 11:08:29.768306  info ALTER TABLE
 ↪  `default`.`github_events5` ATTACH PART 'all_4011_4912_4'
# 2022/05/14 11:08:29.810721  info ALTER TABLE
 ↪  `default`.`github_events5` ATTACH PART 'all_4913_4982_3'
# 2022/05/14 11:08:29.819110  info ALTER TABLE
 ↪  `default`.`github_events5` ATTACH PART 'all_4983_5150_3'
# 2022/05/14 11:08:29.829574  info ALTER TABLE
 ↪  `default`.`github_events5` ATTACH PART 'all_5151_5184_2'
# 2022/05/14 11:08:29.834708  info ALTER TABLE
 ↪  `default`.`github_events5` ATTACH PART 'all_5185_5230_2'
# 2022/05/14 11:08:29.839591  info ALTER TABLE
 ↪  `default`.`github_events5` ATTACH PART 'all_5231_5231_0'
# 2022/05/14 11:08:29.843270  info ALTER TABLE
 ↪  `default`.`github_events5` ATTACH PART 'all_687_1760_4'
# 2022/05/14 11:08:29.860576  info done
 ↪  backup=2022-05-11T07-25-16 operation=restore
 ↪  table=default.github_events5
# 2022/05/14 11:08:29.860615  info done
 ↪  backup=2022-05-11T07-25-16 duration=2.6s operation=restore
# 2022/05/14 11:08:29.860642  info done
 ↪  backup=2022-05-11T07-25-16 operation=restore
#
# real        0m2.687s
# user        0m0.339s
# sys        0m0.191s
```

The restoration appears to be successful. It took 2.687s
Verify table sizes:

```
SELECT
    database,
    table,
    formatReadableSize(sum(data_compressed_bytes) AS size) AS
    ↪  compressed,
    formatReadableSize(sum(data_uncompressed_bytes) AS usize) AS
    ↪  uncompressed,
    round(usize / size, 2) AS compr_rate,
    sum(rows) AS rows,
    count() AS part_count
FROM system.parts
WHERE (active = 1) AND (database LIKE '%') AND (table LIKE '%')
GROUP BY
    database,
    table
ORDER BY size DESC
```

Output (correct):



```
Query id: d510962f-79cb-43fd-904f-2bc5a5d3b0fc

┌─database─┬─table──────────────────┬─compressed─┬─uncompressed─┬─compr_rate─┬───────rows─┬─part_count─┐
│ default  │ github_events2         │ 178.42 GiB │ 1.00 TiB     │       5.76 │ 3119798001 │         11 │
│ default  │ github_events4         │ 178.42 GiB │ 1.00 TiB     │       5.75 │ 3119798001 │         11 │
│ default  │ github_events3         │ 178.42 GiB │ 1.00 TiB     │       5.75 │ 3119798001 │         12 │
│ default  │ github_events5         │ 178.41 GiB │ 1.00 TiB     │       5.75 │ 3119798001 │         10 │
│ default  │ github_events1         │ 178.40 GiB │ 1023.50 GiB  │       5.74 │ 3119798001 │         12 │
│ system   │ asynchronous_metric_log│ 48.55 MiB  │ 511.21 MiB   │      10.53 │   23297845 │          8 │
│ system   │ trace_log              │ 35.72 MiB  │ 570.33 MiB   │      15.97 │    1868161 │          5 │
│ system   │ metric_log             │ 14.70 MiB  │ 340.32 MiB   │      23.15 │     123308 │          5 │
│ system   │ part_log               │ 2.05 MiB   │ 12.24 MiB    │       5.97 │      65728 │          2 │
│ system   │ query_thread_log       │ 130.91 KiB │ 1.01 MiB     │       7.87 │        972 │          2 │
│ system   │ query_log              │ 127.46 KiB │ 1.20 MiB     │       9.61 │       1031 │          2 │
└──────────┴────────────────────────┴────────────┴──────────────┴────────────┴────────────┴────────────┘

11 rows in set. Elapsed: 0.005 sec.
```

New attempt at downloading

Delete local backup:

```
./clickhouse-backup delete local 2022-05-11T07-25-16
# 2022/05/14 11:14:26.277665  info SELECT value FROM
↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/14 11:14:26.280677  info SELECT * FROM system.disks;
# 2022/05/14 11:14:26.327285  info done
↪  backup=2022-05-11T07-25-16 duration=52ms location=local
↪  operation=delete
```

List remote backups

```
./clickhouse-backup list remote --config config.yaml
# 2022/05/14 11:15:16.688656  info SELECT max(toInt64(bytes_on_disk
↪  * 1.02)) AS max_file_size FROM system.parts
```

```
# 2022-05-11T07-25-16   895.29GiB   11/05/2022 10:44:05   remote
↪    tar
# 2022-05-14T10-12-59   2.81KiB     14/05/2022 10:13:31   remote
↪    tar
```

Downloading the remote backup failed once again:

```
time ./clickhouse-backup download 2022-05-11T07-25-16 --config
↪  config.yaml
# 2022/05/14 11:16:23.426432  info SELECT value FROM
↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/14 11:16:23.428319  info SELECT * FROM system.disks;
# 2022/05/14 11:16:23.433846  info SELECT max(toInt64(bytes_on_disk
↪  * 1.02)) AS max_file_size FROM system.parts
# 2022/05/14 11:16:23.506743  info done
↪  backup=2022-05-11T07-25-16 duration=16ms operation=download
↪  size=3.36KiB table_metadata=default.github_events1
# 2022/05/14 11:16:23.512251  info done
↪  backup=2022-05-11T07-25-16 duration=5ms operation=download
↪  size=3.32KiB table_metadata=default.github_events2
# 2022/05/14 11:16:23.517891  info done
↪  backup=2022-05-11T07-25-16 duration=6ms operation=download
↪  size=3.36KiB table_metadata=default.github_events3
# 2022/05/14 11:16:23.522455  info done
↪  backup=2022-05-11T07-25-16 duration=5ms operation=download
↪  size=3.32KiB table_metadata=default.github_events4
# 2022/05/14 11:16:23.527403  info done
↪  backup=2022-05-11T07-25-16 duration=5ms operation=download
↪  size=3.28KiB table_metadata=default.github_events5
# 2022/05/14 11:17:23.532041 error can't acquire semaphore during
↪  downloadTableData: context canceled---------]  70.38% 59s
# 2022/05/14 11:17:23.697884 error can't acquire semaphore during
↪  Download: context canceled backup=2022-05-11T07-25-16
↪  operation=download
# 2022/05/14 11:18:23.730339 error can't acquire semaphore during
↪  downloadTableData: context canceled---------]   9.31% 59s
# 2022/05/14 11:18:23.914406 error one of Download go-routine
↪  return error: one of downloadTableData go-routine return error:
↪  handling file: /all_3441_4218_4/push_distinct_size.bin: context
↪  deadline exceeded
#
# real    2m0.504s
# user    0m11.834s
# sys     0m16.363s
```

A.5 Encrypt ClickHouse and recover from backup

In this experiment, we install `ccrypt` (an encryption tool) in order to encrypt database files and observe the results. Afterwards, we recover the VM using `clickhouse-backup` and Azure Backup respectively. We attempted to encrypt a single file, as well as an entire directory.

This experiment has two main objectives. One is to encrypt data files used by ClickHouse and observe the results. How will database queries be affected? Is it possible to detect a ransomware that encrypts ClickHouse's data files? The other objective is to recover the database after the files have been encrypted. The goal is to learn how to configure and use both `clickhouse-backup` and Azure Backup, as well as evaluating their usability.

A.5.1 Procedure

Files were encrypted by using the tool `ccrypt` with an empty passphrase (example: `ccrypt -e filename`).

SQL queries were run as `azureuser` (the default, non-root user) via `clickhouse-client`.

Bash commands were also run as `azureuser`. Commands were run in `/home/azureuser` unless specified otherwise.

The output of commands and queries are shown in comments (# for bash and PowerShell, `--` for SQL) beneath the command or query. Comments used to explain what a command does are placed above the command. The variables from A.1 need to be declared in order for many commands to work.

A.5.2 Determine which files to encrypt

Before we could encrypt files, we had to find out which files to encrypt.

In order to find out where table data is stored, the path to the `cell_towers` table was fetched from `system.parts`, which, as the name suggests, contains the parts of a database table (when using the MergeTree database engine). A descripton of `system.parts` can be found in the ClickHouse documentation. [63]

Query to fetch path of the `cell_towers` table:

```sql
SELECT
    table,
    disk_name,
    path
FROM system.parts
WHERE table = 'cell_towers'
FORMAT TabSeparated

-- cell_towers     default /var/lib/clickhouse/store/c28/c283470d-9
 ↪  ab3-4be8-bd81-132274c9f9b0/all_1_35_2/
```

```
-- cell_towers     default /var/lib/clickhouse/store/c28/c283470d-9⌋
 ↪  ab3-4be8-bd81-132274c9f9b0/all_36_41_1/
-- cell_towers     default /var/lib/clickhouse/store/c28/c283470d-9⌋
 ↪  ab3-4be8-bd81-132274c9f9b0/all_42_42_0/
```

We navigated to the first path listed and printed the contents of the directory (performed as root):

```
cd /var/lib/clickhouse/store/c28/c283470d-9ab3-4be8-bd81-132274c9f9⌋
 ↪  b0/all_1_35_2/
ls

# area.bin            cell.mrk2        count.txt
 ↪    lat.mrk2  net.bin        range.bin     unit.mrk2
# area.mrk2           changeable.bin   created.bin
 ↪    lon.bin   net.mrk2       range.mrk2    updated.bin
# averageSignal.bin   changeable.mrk2  created.mrk2
 ↪    lon.mrk2  primary.idx    samples.bin   updated.mrk2
# averageSignal.mrk2  checksums.txt
 ↪  default_compression_codec.txt  mcc.bin   radio.bin.cpt
 ↪  samples.mrk2
# cell.bin            columns.txt      lat.bin
 ↪    mcc.mrk2  radio.mrk2     unit.bin
```

In order to determine which files corresponded to columns in the table, we used `DESCRIBE TABLE` to display the columns.

```
DESCRIBE TABLE cell_towers
FORMAT TabSeparated

-- radio   Enum8(\'\' = 0, \'CDMA\' = 1, \'GSM\' = 2, \'LTE\' = 3,
 ↪  \'NR\' = 4, \'UMTS\' = 5)
-- mcc     UInt16
-- net     UInt16
-- area    UInt16
-- cell    UInt64
-- unit    Int16
-- lon     Float64
-- lat     Float64
-- range   UInt32
-- samples UInt32
-- changeable      UInt8
-- created DateTime
-- updated DateTime
-- averageSignal   UInt8
```

Some of the column names displayed matched the names of files in the directory. We made an educated guess that .bin files are the actual data files. A quick Google search indicates that .mrk2 files are related to the indexing of the database.

We decided to encrypt the file radio.bin, since the radio column is used in one of the test queries we will use.

A.5.3 Install ccrypt

```
sudo apt install ccrypt
```

A.5.4 Perform test queries

In order to see how encryption would affect the database, we first performed two test queries and noted the results. The queries are the same as the ones used in the A.1.

Test query 1:

```
SELECT
    radio,
    count() AS c
FROM cell_towers
GROUP BY radio
ORDER BY c DESC
FORMAT TabSeparated

-- UMTS 20686487
-- LTE  12101148
-- GSM  9931312
-- CDMA 556344
-- NR   867
```

Test query 2:

```
SELECT
    mcc,
    count()
FROM cell_towers
GROUP BY mcc
ORDER BY count() DESC
LIMIT 10
FORMAT TabSeparated

-- 310  5024650
-- 262  2622423
-- 250  1953176
-- 208  1891187
-- 724  1836150
-- 404  1729151
-- 234  1618924
-- 510  1353998
-- 440  1343355
-- 311  1332798
```

A.5.5 Encrypt file and repeat test queries

Navigate to directory and encrypt `radio.bin` (performed as root):

```
cd /var/lib/clickhouse/store/c28/c283470d-9ab3-4be8-bd81-132274c9f9↲
↪  b0/all_1_35_2
ccrypt -e radio.bin
```

The test queries from earlier were repeated in `clickhouse-client`. The first test query, which selects data from the `radio` table fails, because the file `radio.bin`, which contains a part of the table used for the query, is encrypted.

Query using `radio`, which fails because `radio.bin` is encrypted:

```
SELECT
    radio,
    count() AS c
FROM cell_towers
GROUP BY radio
ORDER BY c DESC
FORMAT TabSeparated

-- Received exception from server (version 22.4.3):
-- Code: 107. DB::Exception: Received from localhost:9000.
↪  DB::Exception: Cannot open file /var/lib/clickhouse/store/c28/c↲
↪  283470d-9ab3-4be8-bd81-132274c9f9b0/all_1_35_2/radio.bin,
↪  errno: 2, strerror: No such file or directory: While executing
↪  MergeTreeInOrder. (FILE_DOESNT_EXIST)
```

The other test query, which does not use the `radio` table succeeds:

```
SELECT
    mcc,
    count()
FROM cell_towers
GROUP BY mcc
ORDER BY count() DESC
LIMIT 10
FORMAT TabSeparated
```

```
-- 310  5024650
-- 262  2622423
-- 250  1953176
-- 208  1891187
-- 724  1836150
-- 404  1729151
-- 234  1618924
-- 510  1353998
-- 440  1343355
-- 311  1332798
```

A.5.6 Decrypt file and repeat test query

The file `radio.bin.cpt` (the encrypted version of `radio.bin`) was decrypted with `ccrypt -d radio.bin.cpt`, and the test queries were repeated. After decrypting the file, both queries succeeded.

First test query repeated after `radio.bin.cpt` was decrypted:

```
SELECT
    radio,
    count() AS c
FROM cell_towers
GROUP BY radio
ORDER BY c DESC
FORMAT TabSeparated

-- UMTS 20686487
-- LTE  12101148
-- GSM  9931312
-- CDMA 556344
-- NR   867
```

Second test query repeated after `radio.bin.cpt` was decrypted:

```
SELECT
    mcc,
    count()
FROM cell_towers
GROUP BY mcc
ORDER BY count() DESC
LIMIT 10
FORMAT TabSeparated

-- 310 5024650
-- 262 2622423
-- 250 1953176
-- 208 1891187
-- 724 1836150
-- 404 1729151
-- 234 1618924
-- 510 1353998
-- 440 1343355
-- 311 1332798
```

A.5.7 Encrypt all files in `/var/lib/clickhouse/store` and repeat test queries

In order to simulate a ransomware attack where the attacker doesn't encrypt specific files one-by-one, but instead encrypts the entire data directory automatically, we encrypted the directory which stores the data parts for (non-system) ClickHouse tables.

Files in the directory `/var/lib/clickhouse/store`, were encrypted recursively (`-r`) using `ccrypt`. The `-f` (force) flag makes `ccrypt` encrypt write-protected files without asking for confirmation.

Encrypting `store` (performed as root):

```
ccrypt -erf store

-- Enter encryption key:
-- Enter encryption key: (repeat)
-- ccrypt: store/063/0638116a-3105-483f-a2e5-55287d4eaa27/202204_22⌋
↪  95_2350_15: No such file or
↪  directory
-- ccrypt: store/063/0638116a-3105-483f-a2e5-55287d4eaa27/202204_23⌋
↪  55_2355_0: No such file or
↪  directory
-- ccrypt: store/063/0638116a-3105-483f-a2e5-55287d4eaa27/202204_23⌋
↪  51_2351_0: No such file or
↪  directory
-- ccrypt: store/11f/11fafbd1-2c3f-4621-a574-b4f7b5594988/202204_27⌋
↪  94_2794_0: No such file or
↪  directory
-- ccrypt: store/11f/11fafbd1-2c3f-4621-a574-b4f7b5594988/202204_21⌋
↪  40_2793_477: No such file or
↪  directory
```

The contents of one of the encryted directories (performed as root):

```
ls -l /var/lib/clickhouse/store/c28/c283470d-9ab3-4be8-bd81-132274c⌋
↪  9f9b0/all_1_35_2
```

```
# Result:
# total 918640
# -rw-r----- 1 clickhouse clickhouse  41208741 Apr 26 13:09
 ↪  area.bin.cpt
# -rw-r----- 1 clickhouse clickhouse    107576 Apr 26 13:09
 ↪  area.mrk2.cpt
# -rw-r----- 1 clickhouse clickhouse    163548 Apr 26 13:09
 ↪  averageSignal.bin.cpt
# -rw-r----- 1 clickhouse clickhouse    107576 Apr 26 13:09
 ↪  averageSignal.mrk2.cpt
# -rw-r----- 1 clickhouse clickhouse 157679630 Apr 26 13:09
 ↪  cell.bin.cpt
# -rw-r----- 1 clickhouse clickhouse    107576 Apr 26 13:09
 ↪  cell.mrk2.cpt
# -rw-r----- 1 clickhouse clickhouse    163548 Apr 26 13:09
 ↪  changeable.bin.cpt
# -rw-r----- 1 clickhouse clickhouse    107576 Apr 26 13:09
 ↪  changeable.mrk2.cpt
# -rw-r----- 1 clickhouse clickhouse      1183 Apr 26 13:09
 ↪  checksums.txt.cpt
# -rw-r----- 1 clickhouse clickhouse       354 Apr 26 13:09
 ↪  columns.txt.cpt
# -rw-r----- 1 clickhouse clickhouse        40 Apr 26 13:09
 ↪  count.txt.cpt
# -rw-r----- 1 clickhouse clickhouse  65728056 Apr 26 13:09
 ↪  created.bin.cpt
# -rw-r----- 1 clickhouse clickhouse    107576 Apr 26 13:09
 ↪  created.mrk2.cpt
# -rw-r----- 1 clickhouse clickhouse        42 Apr 26 13:09
 ↪  default_compression_codec.txt.cpt
# -rw-r----- 1 clickhouse clickhouse 224774675 Apr 26 13:09
 ↪  lat.bin.cpt
# -rw-r----- 1 clickhouse clickhouse    107576 Apr 26 13:09
 ↪  lat.mrk2.cpt
# -rw-r----- 1 clickhouse clickhouse 227815661 Apr 26 13:09
 ↪  lon.bin.cpt
# -rw-r----- 1 clickhouse clickhouse    107576 Apr 26 13:09
 ↪  lon.mrk2.cpt
# -rw-r----- 1 clickhouse clickhouse    331043 Apr 26 13:09
 ↪  mcc.bin.cpt
# -rw-r----- 1 clickhouse clickhouse    107576 Apr 26 13:09
 ↪  mcc.mrk2.cpt
# -rw-r----- 1 clickhouse clickhouse    345344 Apr 26 13:09
 ↪  net.bin.cpt
```

```
# -rw-r----- 1 clickhouse clickhouse    107576 Apr 26 13:09
 ↪  net.mrk2.cpt
# -rw-r----- 1 clickhouse clickhouse     40361 Apr 26 13:09
 ↪  primary.idx.cpt
# -rw-r----- 1 clickhouse clickhouse    163563 Apr 26 13:09
 ↪  radio.bin.cpt
# -rw-r----- 1 clickhouse clickhouse    107576 Apr 26 13:09
 ↪  radio.mrk2.cpt
# -rw-r----- 1 clickhouse clickhouse  41566798 Apr 26 13:09
 ↪  range.bin.cpt
# -rw-r----- 1 clickhouse clickhouse    107576 Apr 26 13:09
 ↪  range.mrk2.cpt
# -rw-r----- 1 clickhouse clickhouse  66713118 Apr 26 13:09
 ↪  samples.bin.cpt
# -rw-r----- 1 clickhouse clickhouse    107576 Apr 26 13:09
 ↪  samples.mrk2.cpt
# -rw-r----- 1 clickhouse clickhouse   1331731 Apr 26 13:09
 ↪  unit.bin.cpt
# -rw-r----- 1 clickhouse clickhouse    107576 Apr 26 13:09
 ↪  unit.mrk2.cpt
# -rw-r----- 1 clickhouse clickhouse 110977534 Apr 26 13:09
 ↪  updated.bin.cpt
# -rw-r----- 1 clickhouse clickhouse    107576 Apr 26 13:09
 ↪  updated.mrk2.cpt
```

Then the test queries were repeated.
Test query 1 fails:

```
SELECT
    mcc,
    count()
FROM cell_towers
GROUP BY mcc
ORDER BY count() DESC
LIMIT 10
FORMAT TabSeparated

-- Query id: 6d32968b-85bb-4059-9119-ae76a84b8c13
--
-- 0 rows in set. Elapsed: 0.023 sec.
--
-- Received exception from server (version 22.4.3):
```

```
-- Code: 107. DB::Exception: Received from localhost:9000.
↪  DB::Exception: Cannot open file /var/lib/clickhouse/store/c28/c⌋
↪  283470d-9ab3-4be8-bd81-132274c9f9b0/all_1_35_2/mcc.bin, errno:
↪  2, strerror: No such file or directory: While executing
↪  MergeTreeInOrder. (FILE_DOESNT_EXIST)
```

Test query 2 also fails:

```
SELECT
    radio,
    count() AS c
FROM cell_towers
GROUP BY radio
ORDER BY c DESC
FORMAT TabSeparated

-- Query id: 8c913fd5-b0db-4cc2-b176-f389886c3bce
--
-- 0 rows in set. Elapsed: 0.003 sec.
--
-- Received exception from server (version 22.4.3):
-- Code: 107. DB::Exception: Received from localhost:9000.
↪  DB::Exception: Cannot open file /var/lib/clickhouse/store/c28/c⌋
↪  283470d-9ab3-4be8-bd81-132274c9f9b0/all_1_35_2/radio.bin,
↪  errno: 2, strerror: No such file or directory: While executing
↪  MergeTreeInOrder. (FILE_DOESNT_EXIST)
```

Selecting all columns fails:

```
SELECT *
FROM cell_towers
LIMIT 10

-- Query id: 7a96d377-2f5b-4439-96d6-eb554b21fea7
--
-- 0 rows in set. Elapsed: 0.003 sec.
--
-- Received exception from server (version 22.4.3):
-- Code: 107. DB::Exception: Received from localhost:9000.
↪  DB::Exception: Cannot open file /var/lib/clickhouse/store/c28/c⌋
↪  283470d-9ab3-4be8-bd81-132274c9f9b0/all_1_35_2/radio.bin,
↪  errno: 2, strerror: No such file or directory: While executing
↪  MergeTreeInOrder. (FILE_DOESNT_EXIST)
```

Selecting any single column seems to fail:

```
SELECT updated
FROM cell_towers
```

```
LIMIT 10

-- Query id: c983d27d-144d-4296-adb3-ade5ebba3777
--
--
-- 0 rows in set. Elapsed: 0.002 sec.
--
-- Received exception from server (version 22.4.3):
-- Code: 107. DB::Exception: Received from localhost:9000.
 ↪  DB::Exception: Cannot open file /var/lib/clickhouse/store/c28/c⌋
 ↪  283470d-9ab3-4be8-bd81-132274c9f9b0/all_1_35_2/updated.bin,
 ↪  errno: 2, strerror: No such file or directory: While executing
 ↪  MergeTreeInOrder. (FILE_DOESNT_EXIST)
```

A.5.8 Rebuild VM and try to recover

Delete VM

Script to delete the VM and accociated resources (run in Azure Cloud Shell):

```
# Delete the VM
az vm delete --name $CHName --resource-group $RGName --yes

# Get all resources in resource group
$resources = az resource list --resource-group $RGName |
 ↪  ConvertFrom-Json -AsHashtable

# Fetch only the ids of resources with names containing
 ↪  "clickhouse".
# The VM was named "clickhouseVM", which is why this works.
# If the VM had a different name, we would have to grep for
 ↪  something else.
$filtered = foreach($r in $resources) {
    Write-Output $r["id"] | grep clickhouse
}

# Delete the resources
$filtered | % {Remove-AzResource -ResourceId $_ -Force}
$filtered | % {Remove-AzResource -ResourceId $_ -Force}
```

The last command had to be run twice, because some resources could not be deleted while other resources depended on them. Repeating the command had the effect of deleting “child” resources first, so that “parent” resources could be deleted.

Set up VM according to test environment setup

Outputs were skipped for the sake of brevity. Everything went as planned.

Create VM (run from cloud shell):

```
az vm create `
    --resource-group $RGName `
    --name $CHName `
    --image Canonical:UbuntuServer:16.04-LTS:16.04.202109280 `
    --admin-username azureuser `
    --size Standard_B1s `
    --ssh-key-values $SSHPath `
    --public-ip-sku Standard
```

Install clickhouse (run from bash in VM):

```
sudo apt-get install -y apt-transport-https ca-certificates dirmngr
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
 ↪  8919F6BD2B48D754

echo "deb https://packages.clickhouse.com/deb stable main" | sudo
 ↪  tee \
    /etc/apt/sources.list.d/clickhouse.list
sudo apt-get update

sudo apt-get install -y clickhouse-server clickhouse-client

sudo service clickhouse-server start
```

Install clickhouse-backup (run from bash in VM):

```
# Download archive containing binary
wget https://github.com/AlexAkulov/clickhouse-backup/releases/downl⌋
 ↪  oad/v1.3.2/clickhouse-backup-linux-amd64.tar.gz

# Decompress archive
tar -zxvf clickhouse-backup-linux-amd64.tar.gz

# Move binary to home directory
mv build/linux/amd64/clickhouse-backup ~

# Cleanup
rmdir -p build/linux/amd64
rm clickhouse-backup-linux-amd64.tar.gz
```

The `clickhouse-backup` config file was copied from the test environment setup (see A.1) and saved as `~/config.yaml`.

Restore from remote backup

Commands were run from bash on the VM.

List remote backups:

```
sudo ./clickhouse-backup list remote --config config.yaml

# 2022/05/03 10:38:55.223041  info SELECT max(toInt64(bytes_on_disk
↪  * 1.02)) AS max_file_size FROM system.parts
# 2022-05-02T09-48-18   1.07GiB   02/05/2022 09:48:39   remote
↪  tar
```

Restore from the remote backup:

```
sudo ./clickhouse-backup restore_remote 2022-05-02T09-48-18
↪  --config config.yaml

# 2022/05/03 10:40:28.403253  info SELECT value FROM
↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/03 10:40:28.407883  info SELECT * FROM system.disks;
# 2022/05/03 10:40:28.422520  info SELECT max(toInt64(bytes_on_disk
↪  * 1.02)) AS max_file_size FROM system.parts
# 2022/05/03 10:40:28.512501  info done
↪  backup=2022-05-02T09-48-18 duration=10ms operation=download
↪  size=765B table_metadata=default.cell_towers
# 2022/05/03 10:40:48.487522  info done
↪  diff_parts=0 duration=0s operation=downloadDiffParts===========⌋
↪  ===============================================================⌋
↪  =========================================] 100.00%
↪  15s
# 2022/05/03 10:40:49.252289  info done
↪  backup=2022-05-02T09-48-18 duration=20.739s
↪  operation=download_data size=1.07GiB table=default.cell_towers
# 2022/05/03 10:40:49.269557  info done
↪  backup=2022-05-02T09-48-18 duration=20.855s operation=download
↪  size=1.07GiB
# 2022/05/03 10:40:49.326168  info SELECT value FROM
↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/03 10:40:49.420143  info SELECT * FROM system.disks;
# 2022/05/03 10:40:49.429084  info CREATE DATABASE IF NOT EXISTS
↪  default
# ENGINE = Atomic
# 2022/05/03 10:40:49.437031  info SELECT engine FROM
↪  system.databases WHERE name = 'default'
# 2022/05/03 10:40:49.440420  info DROP TABLE IF EXISTS
↪  `default`.`cell_towers` NO DELAY
```

```
# 2022/05/03 10:40:49.444862  info CREATE DATABASE IF NOT EXISTS
 ↪  `default`
# 2022/05/03 10:40:49.448408  info CREATE TABLE default.cell_towers
 ↪  UUID 'f9b9b44f-3af6-4bf5-9458-9ef6a81a3519' (`radio` Enum8('' =
 ↪  0, 'CDMA' = 1, 'GSM' = 2, 'LTE' = 3, 'NR' = 4, 'UMTS' = 5),
 ↪  `mcc` UInt16, `net` UInt16, `area` UInt16, `cell` UInt64,
 ↪  `unit` Int16, `lon` Float64, `lat` Float64, `range` UInt32,
 ↪  `samples` UInt32, `changeable` UInt8, `created` DateTime,
 ↪  `updated` DateTime, `averageSignal` UInt8) ENGINE = MergeTree
 ↪  ORDER BY (radio, mcc, net, created) SETTINGS index_granularity
 ↪  = 8192
# 2022/05/03 10:40:49.498743  info SELECT count() FROM
 ↪  system.settings WHERE name =
 ↪  'show_table_uuid_in_table_create_query_if_not_nil'
# 2022/05/03 10:40:49.519183  info SELECT name FROM
 ↪  system.databases WHERE engine IN ('MySQL','PostgreSQL')
# 2022/05/03 10:40:49.533985  info
#                 SELECT
#                         countIf(name='data_path')
 ↪  is_data_path_present,
#                         countIf(name='data_paths')
 ↪  is_data_paths_present,
#                         countIf(name='uuid') is_uuid_present,
#                         countIf(name='create_table_query')
 ↪  is_create_table_query_present,
#                         countIf(name='total_bytes')
 ↪  is_total_bytes_present
#                 FROM system.columns WHERE database='system' AND
 ↪  table='tables'
#
# 2022/05/03 10:40:49.545657  info SELECT database, name, engine ,
 ↪  data_paths , uuid , create_table_query , coalesce(total_bytes,
 ↪  0) AS total_bytes   FROM system.tables WHERE is_temporary = 0
 ↪  SETTINGS show_table_uuid_in_table_create_query_if_not_nil=1
# 2022/05/03 10:40:49.566479  info SELECT sum(bytes_on_disk) as
 ↪  size FROM system.parts WHERE database='default' AND
 ↪  table='cell_towers' GROUP BY database, table
# 2022/05/03 10:40:49.578319  info ALTER TABLE
 ↪  `default`.`cell_towers` ATTACH PART 'all_1_35_2'
# 2022/05/03 10:40:49.589570  info ALTER TABLE
 ↪  `default`.`cell_towers` ATTACH PART 'all_36_41_1'
# 2022/05/03 10:40:49.596540  info ALTER TABLE
 ↪  `default`.`cell_towers` ATTACH PART 'all_42_42_0'
```

```
# 2022/05/03 10:40:49.601181  info done
↪  backup=2022-05-02T09-48-18 operation=restore
↪  table=default.cell_towers
# 2022/05/03 10:40:49.601380  info done
↪  backup=2022-05-02T09-48-18 duration=109ms operation=restore
# 2022/05/03 10:40:49.601555  info done
↪  backup=2022-05-02T09-48-18 operation=restore
```

Repeat test queries

Test queries were run from `clickhouse-backup` and compared with earlier results to se if they were valid. Both queries returned valid results, indicating that the recovery was successful.

At first, performance was very poor. Sometimes the queries would time out because they took too long (more than 300 seconds). After restarting the VM, queries were as fast as usual (0.072s and 0.2s respectively).

Test query 1:

```
SELECT
    radio,
    count() AS c
FROM cell_towers
GROUP BY radio
ORDER BY c DESC
FORMAT TabSeparated

-- Query id: fea063fa-98e6-4a99-b373-4f51d99be3c1
--
-- UMTS    20686487
-- LTE     12101148
-- GSM     9931312
-- CDMA    556344
-- NR      867
```

Test query 2:

```
SELECT
    mcc,
    count()
FROM cell_towers
GROUP BY mcc
ORDER BY count() DESC
LIMIT 10
FORMAT TabSeparated

-- Query id: e4cd8484-7d4e-4e97-b78f-5373ed61c2b2
```

```
--
-- 310     5024650
-- 262     2622423
-- 250     1953176
-- 208     1891187
-- 724     1836150
-- 404     1729151
-- 234     1618924
-- 510     1353998
-- 440     1343355
-- 311     1332798
```

A.5.9 Repeat encryption and recover database with Azure Backup

Encrypt `store` directory

Run as root in bash on the VM:

```
ccrypt -erf /var/lib/clickhouse/store
```

Delete VM

Deleting the VM:

```
az vm delete --name $CHName --resource-group $RGName --yes
```

When using Azure Backup via the Azure Portal to recover a VM, a virtual network and a subnet have to be provided. We assume the same holds true for recovering via the Azure CLI, even though no commands seem to be asking for the aforementioned resources (at least not directly). This is why we only deleted the VM, and not related resources like network cards and IP adresses as well.

Create a Storage Account

In order to recover the VM, a storage account is required for staging the recovery.

Creating a storage account:

```
az storage account create `
    --name $StagingSAName `
    --resource-group $RGName `
    --location eastus `
    --sku Standard_LRS
```

Output:

```
{
  "accessTier": "Hot",
  "allowBlobPublicAccess": true,
"allowCrossTenantReplication": null,
"allowSharedKeyAccess": null,
"allowedCopyScope": null,
"azureFilesIdentityBasedAuthentication": null,
"blobRestoreStatus": null,
"creationTime": "2022-05-04T07:11:01.403417+00:00",
"customDomain": null,
"defaultToOAuthAuthentication": null,
"dnsEndpointType": null,
"enableHttpsTrafficOnly": true,
"enableNfsV3": null,
"encryption": {
  "encryptionIdentity": null,
  "keySource": "Microsoft.Storage",
  "keyVaultProperties": null,
  "requireInfrastructureEncryption": null,
  "services": {
    "blob": {
      "enabled": true,
      "keyType": "Account",
      "lastEnabledTime": "2022-05-04T07:11:01.528489+00:00"
    },
    "file": {
      "enabled": true,
      "keyType": "Account",
      "lastEnabledTime": "2022-05-04T07:11:01.528489+00:00"
    },
    "queue": null,
    "table": null
  }
},
"extendedLocation": null,
"failoverInProgress": null,
"geoReplicationStats": null,
"id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
↪  ceGroups/testRG/providers/Microsoft.Storage/storageAccounts/s⌋
↪  tagingchsa",
"identity": null,
"immutableStorageWithVersioning": null,
"isHnsEnabled": null,
"isLocalUserEnabled": null,
"isSftpEnabled": null,
"keyCreationTime": {
  "key1": "2022-05-04T07:11:01.528489+00:00",
    "key2": "2022-05-04T07:11:01.528489+00:00"
  },
  "keyPolicy": null,
  "kind": "StorageV2",
  "largeFileSharesState": null,
  "lastGeoFailoverTime": null,
  "location": "eastus",
  "minimumTlsVersion": "TLS1_0",
  "name": "stagingchsa",
  "networkRuleSet": {
    "bypass": "AzureServices",
    "defaultAction": "Allow",
    "ipRules": [],
    "resourceAccessRules": null,
    "virtualNetworkRules": []
  },
  "primaryEndpoints": {
    "blob": "https://stagingchsa.blob.core.windows.net/",
    "dfs": "https://stagingchsa.dfs.core.windows.net/",
    "file": "https://stagingchsa.file.core.windows.net/",
    "internetEndpoints": null,
    "microsoftEndpoints": null,
    "queue": "https://stagingchsa.queue.core.windows.net/",
    "table": "https://stagingchsa.table.core.windows.net/",
    "web": "https://stagingchsa.z13.web.core.windows.net/"
  },
  "primaryLocation": "eastus",
  "privateEndpointConnections": [],
  "provisioningState": "Succeeded",
  "publicNetworkAccess": null,
  "resourceGroup": "testRG",
  "routingPreference": null,
  "sasPolicy": null,
  "secondaryEndpoints": null,
  "secondaryLocation": null,
  "sku": {
    "name": "Standard_LRS",
    "tier": "Standard"
  },
  "statusOfPrimary": "available",
  "statusOfSecondary": null,
  "storageAccountSkuConversionStatus": null,
  "tags": {},
  "type": "Microsoft.Storage/storageAccounts"
```

```
}
```

Recover from Azure Backup

The VM was recovered by modifying instructions in the documentation [64] to suit our use case. Many steps, like creating a Recovery Services vault, had already been performed and were thus skipped. The commands we performed to recover the VM are listed below. The process consists of retrieving information used to list relevant recovery points. Then, a restore job is started in order to recover the VHD. When the restore job is finished, it produces a template which is then used to deploy a new VM.

Initial setup:

```
# Get RSV and set context
$RSV = Get-AzRecoveryServicesVault -Name $RSVName
 ↪  -ResourceGroupName $RGName
Set-AzRecoveryServicesVaultContext -Vault $RSV

# Select VM
$namedContainer = Get-AzRecoveryServicesBackupContainer
 ↪  -ContainerType "AzureVM" -Status "Registered" -FriendlyName
 ↪  $CHName -VaultId $RSV.ID
$backupitem = Get-AzRecoveryServicesBackupItem -Container
 ↪  $namedContainer  -WorkloadType "AzureVM" -VaultId $RSV.ID

# Get start and end date
$startDate = (Get-Date).AddDays(-7)
$endDate = Get-Date

# Store recovery points in variable
$rp = Get-AzRecoveryServicesBackupRecoveryPoint -Item $backupitem
 ↪  -StartDate $startdate.ToUniversalTime() -EndDate
 ↪  $enddate.ToUniversalTime() -VaultId $RSV.ID
```

Show contents of rp:

```
$rp

# RecoveryPointId    RecoveryPointType  RecoveryPointTime
 ↪  ContainerName                        ContainerType
# ---------------    -----------------  -----------------
 ↪  -------------                        -------------
# 201217219254041    CrashConsistent    5/4/2022 9:30:41 AM
 ↪  iaasvmcontainerv2;testrg;clickhouse... AzureVM
# 208189198246820    CrashConsistent    5/2/2022 10:35:59 PM
 ↪  iaasvmcontainerv2;testrg;clickhouse... AzureVM
```

```
# 210069911701143    CrashConsistent    5/2/2022 12:37:15 PM
↪  iaasvmcontainerv2;testrg;clickhouse... AzureVM
```

The recovery point `$rp[1]` (208189198246820) is the one that was triggered manually earlier (see A.1). It was chosen to be used for recovery.

A restore job was started:

```
# Select recovery point
$RecPoint = $rp[1]

# Create a restore job for the backup item
$restorejob = Restore-AzRecoveryServicesBackupItem -RecoveryPoint
↪  $RecPoint -StorageAccountName $StagingSAName
↪  -StorageAccountResourceGroupName $RGName
↪  -TargetResourceGroupName $RGName -VaultId $RSV.ID

# Wait for the restore job to complete
Wait-AzRecoveryServicesBackupJob -Job $restorejob -Timeout 43200

# Get details of the restore job
$restorejob = Get-AzRecoveryServicesBackupJob -Job $restorejob
↪  -VaultId $RSV.ID
$details = Get-AzRecoveryServicesBackupJobDetail -Job $restorejob
↪  -VaultId $RSV.ID

# Print details
$details
# WorkloadName     Operation            Status
↪  StartTime                 EndTime                   JobID
# ------------     ---------            ------
↪  ---------                 -------                   -----
# clickhousevm     Restore              Completed
↪  5/4/2022 9:30:38 AM       5/4/2022 9:51:17 AM
↪  3019de90-58c6-4f40-9157-cd53c7546223
```

We then declared some variables based on the details, to be used for future commands:

```
$properties = $details.properties
$storageAccountName = $properties["Target Storage Account Name"]
$containerName = $properties["Config Blob Container Name"]
$templateBlobURI = $properties["Template Blob Uri"]
```

The contents of `$properties`:

```
Key                            Value
---                            -----
```

```
Job Type                    Recover disks
Target VM Name              vmName
Target Storage Account Name stagingchsa
Recovery point time         5/2/2022 10:35:59 PM
Config Blob Name
 ↪  config-clickhousevm-19ee2306-41c2-49bf-896f-40aa8172d2ea.json
Config Blob Container Name
 ↪  clickhousevm-0edd1295169343bda63beb53c8781b85
Config Blob Uri             https://stagingchsa.blob.core.windows.n⌋
 ↪  et/clickhousevm-0edd1295169343bda63beb53c8781b85/config-clickho⌋
 ↪  usevm-19ee2306-41c2-49bf-896f-40aa8172d2ea.j...
Target resource group       testRG
Template Blob Uri           https://stagingchsa.blob.core.windows.n⌋
 ↪  et/clickhousevm-0edd1295169343bda63beb53c8781b85/azuredeploy19e⌋
 ↪  e2306-41c2-49bf-896f-40aa8172d2ea.json
```

Deploying the VM from the template:

```
# Template name was copied from the "Template Blob Uri"
$templateName =
 ↪  "azuredeploy19ee2306-41c2-49bf-896f-40aa8172d2ea.json"

# Set the storage account
Set-AzCurrentStorageAccount -Name $storageAccountName
 ↪  -ResourceGroupName $RGName

# Generate SAS token
$templateBlobFullURI = New-AzStorageBlobSASToken -Container
 ↪  $containerName -Blob $templateName -Permission r -FullUri

# Deploy VM (VirtualMachineName had to be specified interactively)
New-AzResourceGroupDeployment -Name $CHName -ResourceGroupName
 ↪  $RGName -TemplateUri $templateBlobFullURI
```

Output from the New-AzResourceGroupDeployment (The VirtualMachineName had to be supplied via user input, indentation was adjusted for readability):

```
cmdlet New-AzResourceGroupDeployment at command pipeline position 1
Supply values for the following parameters:
(Type !? for Help.)
VirtualMachineName: clickhouseVM

DeploymentName          : clickhouseVM
ResourceGroupName       : testRG
ProvisioningState       : Succeeded
Timestamp               : 5/4/2022 11:50:42 AM
```

```
Mode                    : Incremental
TemplateLink            :
Uri            : https://stagingchsa.blob.core.windows.net/clickhou⌋
 ↪  sevm-0edd1295169343bda63beb53c8781b85/azuredeploy19ee2306-41c2-⌋
 ↪  49bf-896f-40aa8172d
2ea.json?sv=2021-04-10&se=2022-05-04T12%3A46%3A51Z&sr=b&sp=r&sig=Le⌋
 ↪  %2BGX1JoU%2FSivJi9EpXyk0bPHAXxBsww%2BxY7SpGsP9k%3D
ContentVersion : 1.0.0.0
Parameters              :
Name                           Type                        Value
=============================  ==========================  ==========
virtualMachineName             String
 ↪  "clickhouseVM"
virtualNetwork                 String
 ↪  "clickhouseVMVNET"
virtualNetworkResourceGroup    String                      "testRG"
subnet                         String
 ↪  "clickhouseVMSubnet"
osDiskName                     String
 ↪  "clickhouseVMOSDisk"
networkInterfacePrefixName     String
 ↪  "clickhouseVMRestoredNIC"
publicIpAddressName            String
 ↪  "clickhouseVMRestoredip"

Outputs                 :
DeploymentDebugLogLevel :
```

Repeat test queries

The test queries were repeated in `clickhouse-client`. Both were successful and reasonably fast.

Test query 1:

```sql
SELECT
    radio,
    count() AS c
FROM cell_towers
GROUP BY radio
ORDER BY c DESC
FORMAT TabSeparated

-- Query id: fa3847f4-e564-4c8b-b2be-e85e2d4db87e
--
-- UMTS        20686487
-- LTE        12101148
-- GSM        9931312
-- CDMA        556344
-- NR        867
--
-- 5 rows in set. Elapsed: 0.094 sec. Processed 43.28 million rows,
↪  43.28 MB (462.68 million rows/s., 462.68 MB/s.)
```

Test query 2:

```
SELECT
    mcc,
    count()
FROM cell_towers
GROUP BY mcc
ORDER BY count() DESC
LIMIT 10
FORMAT TabSeparated

-- Query id: f944c5ad-937c-45fa-a9f1-d173d12f7dbc
--
-- 310        5024650
-- 262        2622423
-- 250        1953176
-- 208        1891187
-- 724        1836150
-- 404        1729151
-- 234        1618924
-- 510        1353998
-- 440        1343355
-- 311        1332798
--
-- 10 rows in set. Elapsed: 0.392 sec. Processed 43.28 million
↪  rows, 86.55 MB (110.54 million rows/s., 221.07 MB/s.)
```

A.6 Encrypt local backups

In this experiment, we encrypted local backups made with `clickhouse-backup` and observed the results.

A.6.1 Create and encrypt a local backup

A local backup was created:

```
sudo ./clickhouse-backup create
# 2022/05/06 13:33:41.900950  info SELECT name, engine FROM
 ↪  system.databases WHERE name NOT IN ('system',
 ↪  'INFORMATION_SCHEMA', 'information_schema')
# 2022/05/06 13:33:41.910260  info SHOW CREATE DATABASE `default`
# 2022/05/06 13:33:41.922356  info SELECT count() FROM
 ↪  system.settings WHERE name =
 ↪  'show_table_uuid_in_table_create_query_if_not_nil'
# 2022/05/06 13:33:41.928718  info SELECT name FROM
 ↪  system.databases WHERE engine IN ('MySQL','PostgreSQL')
# 2022/05/06 13:33:41.934157  info
#                  SELECT
#                          countIf(name='data_path')
 ↪  is_data_path_present,
#                          countIf(name='data_paths')
 ↪  is_data_paths_present,
#                          countIf(name='uuid') is_uuid_present,
#                          countIf(name='create_table_query')
 ↪  is_create_table_query_present,
#                          countIf(name='total_bytes')
 ↪  is_total_bytes_present
#                  FROM system.columns WHERE database='system' AND
 ↪  table='tables'
#
# 2022/05/06 13:33:41.945031  info SELECT database, name, engine ,
 ↪  data_paths , uuid , create_table_query , coalesce(total_bytes,
 ↪  0) AS total_bytes   FROM system.tables WHERE is_temporary = 0
 ↪  SETTINGS show_table_uuid_in_table_create_query_if_not_nil=1
# 2022/05/06 13:33:41.969735  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/06 13:33:41.975802  info SELECT * FROM system.disks;
# 2022/05/06 13:33:41.981654  info ALTER TABLE
 ↪  `default`.`cell_towers` FREEZE WITH NAME
 ↪  '46510daab8304f1bb845cc0a3378dc81';
# 2022/05/06 13:33:42.094131  info done
 ↪  backup=2022-05-06T13-33-41 operation=create
 ↪  table=default.cell_towers
# 2022/05/06 13:33:42.094551  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_DESCRIBE'
# 2022/05/06 13:33:42.103890  info done
 ↪  backup=2022-05-06T13-33-41 duration=206ms operation=create
```

The `backup` directory was encrypted (as `root`) with an empty passphrase (output truncated):

```
ccrypt -erf /var/lib/clickhouse/backup
```

```
# Enter encryption key:
# Enter encryption key: (repeat)
# ccrypt: warning: backup/2022-05-06T13-33-41/shadow/default/cell_t⌋
↪  owers/default/all_36_41_1/updated.mrk2 has 2
↪  links
# ccrypt: warning: backup/2022-05-06T13-33-41/shadow/default/cell_t⌋
↪  owers/default/all_36_41_1/lon.bin has 2
↪  links
# ccrypt: warning: backup/2022-05-06T13-33-41/shadow/default/cell_t⌋
↪  owers/default/all_36_41_1/checksums.txt has 2
↪  links
# ccrypt: warning: backup/2022-05-06T13-33-41/shadow/default/cell_t⌋
↪  owers/default/all_36_41_1/default_compression_codec.txt has 2
↪  links
# ccrypt: warning: backup/2022-05-06T13-33-41/shadow/default/cell_t⌋
↪  owers/default/all_36_41_1/radio.bin has 2
↪  links
# ccrypt: warning: backup/2022-05-06T13-33-41/shadow/default/cell_t⌋
↪  owers/default/all_36_41_1/count.txt has 2
↪  links
# ccrypt: warning: backup/2022-05-06T13-33-41/shadow/default/cell_t⌋
↪  owers/default/all_36_41_1/range.bin has 2
↪  links
# ccrypt: warning: backup/2022-05-06T13-33-41/shadow/default/cell_t⌋
↪  owers/default/all_36_41_1/area.bin has 2
↪  links
# ccrypt: warning: backup/2022-05-06T13-33-41/shadow/default/cell_t⌋
↪  owers/default/all_36_41_1/lon.mrk2 has 2
↪  links
# ccrypt: warning: backup/2022-05-06T13-33-41/shadow/default/cell_t⌋
↪  owers/default/all_36_41_1/primary.idx has 2
↪  links
# ccrypt: warning: backup/2022-05-06T13-33-41/shadow/default/cell_t⌋
↪  owers/default/all_36_41_1/mcc.mrk2 has 2
↪  links
# ccrypt: warning: backup/2022-05-06T13-33-41/shadow/default/cell_t⌋
↪  owers/default/all_36_41_1/columns.txt has 2
↪  links
# ccrypt: warning: backup/2022-05-06T13-33-41/shadow/default/cell_t⌋
↪  owers/default/all_36_41_1/created.bin has 2
↪  links
```

Listing backups via `clickhouse-backup`:

```
./clickhouse-backup list
```

```
# 2022/05/06 13:39:10.487501  info SELECT value FROM
↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/06 13:39:10.601852  info SELECT * FROM system.disks;
# 2022-05-06T13-33-41   ???   06/05/2022 13:37:27   local
```

Trying to restore from the encrypted backup:

```
./clickhouse-backup restore local 2022-05-06T13-33-41
# 2022/05/06 13:40:23.945646  info SELECT value FROM
↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/06 13:40:23.950380  info SELECT * FROM system.disks;
# 2022/05/06 13:40:23.956845 error stat
↪  /var/lib/clickhouse/backup/local/metadata: no such file or
↪  directory
```

We also discovered that the `clickhouse-client` would not start while `/var/lib/clickhouse/backup` was encrypted.

A.7 Delete backups via `clickhouse-backup`

In this experiment, local and remote backups were deleted using `clickhouse-backup`. Attempts were then made to recover the deleted remote files by undeleting them using the Azure CLI. The first attempt failed, because soft delete had not been enabled correctly. Soft delete was then enabled for blobs, and the experiment was repeated.

A.7.1 Preparation

The following variables were declared:

```
$AccKey = "NdbW07WlHBf5zcpMXundwkP88Ie2SO1Ad+84VD8moaUg1ihIeRR7cEdy⌋
↪  4FXIgHRvIQwPIMc7eD2q+ASt6EqxWg==" # Storage Account Access
↪  Key
$StorageAccount = Get-AzStorageAccount | Where-Object {
↪  $_.StorageAccountName -eq $SAName} # clickhouse-backup storage
↪  account
```

A.7.2 Delete local backups

Listing local backups:

```
sudo ./clickhouse-backup list
# 2022/05/06 06:20:48.820972  info SELECT value FROM
↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/06 06:20:48.830191  info SELECT * FROM system.disks;
# 2022-05-02T09-34-00   1.07GiB   02/05/2022 09:34:01   local
# 2022-05-02T09-48-03   1.07GiB   02/05/2022 09:48:03   local
# 2022-05-02T09-48-18   1.07GiB   02/05/2022 09:48:18   local
```

Deleting all the local backups:

```
sudo ./clickhouse-backup delete local 2022-05-02T09-34-00
# 2022/05/06 06:51:05.560887  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/06 06:51:05.567336  info SELECT * FROM system.disks;
# 2022/05/06 06:51:05.585788  info done
 ↪  backup=2022-05-02T09-34-00 duration=29ms location=local
 ↪  operation=delete

sudo ./clickhouse-backup delete local 2022-05-02T09-48-03
# 2022/05/06 06:51:05.630403  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/06 06:51:05.635599  info SELECT * FROM system.disks;
# 2022/05/06 06:51:05.645033  info done
 ↪  backup=2022-05-02T09-48-03 duration=18ms location=local
 ↪  operation=delete

sudo ./clickhouse-backup delete local 2022-05-02T09-48-18
# 2022/05/06 06:51:05.677436  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/06 06:51:05.683594  info SELECT * FROM system.disks;
# 2022/05/06 06:51:05.696499  info done
 ↪  backup=2022-05-02T09-48-18 duration=23ms location=local
 ↪  operation=delete
```

Trying to list the backups (no results):

```
sudo ./clickhouse-backup list
# 2022/05/06 06:52:09.307718  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/06 06:52:09.313018  info SELECT * FROM system.disks;
```

A.7.3 Delete remote backups via `clickhouse-backup` and restore via Azure CLI (first attempt)

Create remote backup

Creating a remote backup:

```
sudo ./clickhouse-backup create_remote --config config.yaml
# 2022/05/06 12:08:24.906957  info SELECT name, engine FROM
 ↪  system.databases WHERE name NOT IN ('system',
 ↪  'INFORMATION_SCHEMA', 'information_schema')
```

```
# 2022/05/06 12:08:24.916734  info SHOW CREATE DATABASE `default`
# 2022/05/06 12:08:24.928118  info SELECT count() FROM
 ↪  system.settings WHERE name =
 ↪  'show_table_uuid_in_table_create_query_if_not_nil'
# 2022/05/06 12:08:24.934948  info SELECT name FROM
 ↪  system.databases WHERE engine IN ('MySQL','PostgreSQL')
# 2022/05/06 12:08:24.940334  info
#                 SELECT
#                         countIf(name='data_path')
 ↪  is_data_path_present,
#                         countIf(name='data_paths')
 ↪  is_data_paths_present,
#                         countIf(name='uuid') is_uuid_present,
#                         countIf(name='create_table_query')
 ↪  is_create_table_query_present,
#                         countIf(name='total_bytes')
 ↪  is_total_bytes_present
#                 FROM system.columns WHERE database='system' AND
 ↪  table='tables'
#
# 2022/05/06 12:08:24.947893  info SELECT database, name, engine ,
 ↪  data_paths , uuid , create_table_query , coalesce(total_bytes,
 ↪  0) AS total_bytes   FROM system.tables WHERE is_temporary = 0
 ↪  SETTINGS show_table_uuid_in_table_create_query_if_not_nil=1
# 2022/05/06 12:08:24.984565  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/06 12:08:24.991468  info SELECT * FROM system.disks;
# 2022/05/06 12:08:24.996567  info ALTER TABLE
 ↪  `default`.`cell_towers` FREEZE WITH NAME
 ↪  '247d916bc61f4c05af26280ad8f3be9f';
# 2022/05/06 12:08:25.032979  info done
 ↪  backup=2022-05-06T12-08-24 operation=create
 ↪  table=default.cell_towers
# 2022/05/06 12:08:25.034527  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_DESCRIBE'
# 2022/05/06 12:08:25.041405  info done
 ↪  backup=2022-05-06T12-08-24 duration=142ms operation=create
# 2022/05/06 12:08:25.047661  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/06 12:08:25.051625  info SELECT * FROM system.disks;
# 2022/05/06 12:08:25.059594  info SELECT max(toInt64(bytes_on_disk
 ↪  * 1.02)) AS max_file_size FROM system.parts
```

```
# 2022/05/06 12:10:21.914725  info done
↪  backup=2022-05-06T12-08-24 duration=1m56.772s operation=upload
↪  size=1.07GiB table=default.cell_towers
# 2022/05/06 12:10:21.932113  info done
↪  backup=2022-05-06T12-08-24 duration=1m56.89s operation=upload
↪  size=1.07GiB
```

Listing remote backups:

```
sudo ./clickhouse-backup list remote --config config.yaml
# 2022/05/06 12:13:23.147970  info SELECT max(toInt64(bytes_on_disk
↪  * 1.02)) AS max_file_size FROM system.parts
# 2022-05-06T12-08-24   1.07GiB   06/05/2022 12:10:21   remote
↪  tar
```

Delete remote backup

Deleting remote backups:

```
sudo ./clickhouse-backup delete remote 2022-05-06T12-08-24 --config
↪  config.yaml
# 2022/05/06 12:15:02.656213  info SELECT max(toInt64(bytes_on_disk
↪  * 1.02)) AS max_file_size FROM system.parts
# 2022/05/06 12:15:02.777130  info done
↪  backup=2022-05-06T12-08-24 duration=124ms location=remote
↪  operation=delete
```

Trying to list remote backups (no result):

```
sudo ./clickhouse-backup list remote --config config.yaml
# 2022/05/06 12:15:19.106515  info SELECT max(toInt64(bytes_on_disk
↪  * 1.02)) AS max_file_size FROM system.parts
```

Undelete remote backups via Azure CLI

Trying to list remote backups (blobs) via the Azure Cloud Shell (including deleted blobs):

```
az storage blob list --account-name $SAName --container-name
↪  $ContainerName --account-key $AccKey --include d
```

The result was an empty array (`[]`). The container was still visible in the Azure Portal, though.

It appears that `clickhouse-backup` bypasses container soft delete in a sense, by deleting the contents of the container, but not the container itself. Thus, the container (and the backups) cannot be undeleted, since it was never deleted in the first place. To verify this, the container was deleted via the Azure Portal and then undeleted.

A.7.4 Enable soft delete for blobs

Since the previous attempt at recovering remote `clickhouse-backups` via soft delete failed, because we used the “wrong” type of soft delete, we decided to try again. This time, we enabled soft delete for the blobs themselves.

Enable soft delete for Blobs:

```
az storage account blob-service-properties update --account-name
 ↪  $SAName `
    --resource-group $RGName `
    --enable-delete-retention true `
    --delete-retention-days 7
```

```
{
  "automaticSnapshotPolicyEnabled": null,
  "changeFeed": null,
  "containerDeleteRetentionPolicy": {
    "allowPermanentDelete": null,
    "days": 7,
    "enabled": true
  },
  "cors": {
    "corsRules": []
  },
  "defaultServiceVersion": null,
  "deleteRetentionPolicy": {
    "allowPermanentDelete": null,
    "days": 7,
    "enabled": true
  },
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
  ↪  ceGroups/testRG/providers/Microsoft.Storage/storageAccounts/c⌋
  ↪  hbksa/blobServices/default",
  "isVersioningEnabled": null,
  "lastAccessTimeTrackingPolicy": null,
  "name": "default",
  "resourceGroup": "testRG",
  "restorePolicy": null,
  "sku": null,
  "type": "Microsoft.Storage/storageAccounts/blobServices"
}
```

A.7.5 Delete remote backups via `clickhouse-backup` and restore via Azure CLI (second attempt)

Create remote backup

A remote backup was created:

```
sudo ./clickhouse-backup create_remote --config config.yaml
# 2022/05/06 12:41:19.110753  info SELECT name, engine FROM
 ↪  system.databases WHERE name NOT IN ('system',
 ↪  'INFORMATION_SCHEMA', 'information_schema')
# 2022/05/06 12:41:19.185348  info SHOW CREATE DATABASE `default`
# 2022/05/06 12:41:19.192509  info SELECT count() FROM
 ↪  system.settings WHERE name =
 ↪  'show_table_uuid_in_table_create_query_if_not_nil'
# 2022/05/06 12:41:19.199675  info SELECT name FROM
 ↪  system.databases WHERE engine IN ('MySQL','PostgreSQL')
# 2022/05/06 12:41:19.205297  info
#                 SELECT
#                         countIf(name='data_path')
 ↪  is_data_path_present,
#                         countIf(name='data_paths')
 ↪  is_data_paths_present,
#                         countIf(name='uuid') is_uuid_present,
#                         countIf(name='create_table_query')
 ↪  is_create_table_query_present,
#                         countIf(name='total_bytes')
 ↪  is_total_bytes_present
#                 FROM system.columns WHERE database='system' AND
 ↪  table='tables'
#
# 2022/05/06 12:41:19.212594  info SELECT database, name, engine ,
 ↪  data_paths , uuid , create_table_query , coalesce(total_bytes,
 ↪  0) AS total_bytes   FROM system.tables WHERE is_temporary = 0
 ↪  SETTINGS show_table_uuid_in_table_create_query_if_not_nil=1
# 2022/05/06 12:41:19.231869  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/06 12:41:19.237796  info SELECT * FROM system.disks;
# 2022/05/06 12:41:19.243268  info ALTER TABLE
 ↪  `default`.`cell_towers` FREEZE WITH NAME
 ↪  '8fac6ff911f04cf68521c2d87f778dd7';
# 2022/05/06 12:41:19.320599  info done
 ↪  backup=2022-05-06T12-41-19 operation=create
 ↪  table=default.cell_towers
# 2022/05/06 12:41:19.321056  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_DESCRIBE'
```

```
# 2022/05/06 12:41:19.330933  info done
 ↪  backup=2022-05-06T12-41-19 duration=227ms operation=create
# 2022/05/06 12:41:19.338082  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/06 12:41:19.342402  info SELECT * FROM system.disks;
# 2022/05/06 12:41:19.350151  info SELECT max(toInt64(bytes_on_disk
 ↪  * 1.02)) AS max_file_size FROM system.parts
# 2022/05/06 12:41:40.804639  info done
 ↪  backup=2022-05-06T12-41-19 duration=21.379s operation=upload
 ↪  size=1.07GiB table=default.cell_towers
# 2022/05/06 12:41:40.822260  info done
 ↪  backup=2022-05-06T12-41-19 duration=21.491s operation=upload
 ↪  size=1.07GiB
```

Listing the remote backup:

```
sudo ./clickhouse-backup list remote --config config.yaml
# 2022/05/06 12:43:35.718005  info SELECT max(toInt64(bytes_on_disk
 ↪  * 1.02)) AS max_file_size FROM system.parts
# 2022-05-06T12-41-19   1.07GiB   06/05/2022 12:41:40   remote
 ↪  tar
```

Delete remote backup

Deleting the remote backup:

```
sudo ./clickhouse-backup delete remote 2022-05-06T12-41-19 --config
 ↪  config.yaml
# 2022/05/06 12:44:26.871214  info SELECT max(toInt64(bytes_on_disk
 ↪  * 1.02)) AS max_file_size FROM system.parts
# 2022/05/06 12:44:26.977566  info done
 ↪  backup=2022-05-06T12-41-19 duration=110ms location=remote
 ↪  operation=delete
```

Listing the remote backup (no result):

```
sudo ./clickhouse-backup list remote --config config.yaml
# 2022/05/06 12:44:54.497519  info SELECT max(toInt64(bytes_on_disk
 ↪  * 1.02)) AS max_file_size FROM system.parts
```

Listing the deleted blobs in the Azure CLI:

```
az storage blob list --account-name $SAName --container-name
 ↪  $ContainerName --account-key $AccKey --include d
```

Output (notice that `deleted` is `true` for the blobs):

```
[
  {
    "container": "chbkcontainer",
    "content": "",
    "deleted": true,
    "encryptedMetadata": null,
    "encryptionKeySha256": null,
    "encryptionScope": null,
    "hasLegalHold": null,
    "hasVersionsOnly": null,
    "immutabilityPolicy": {
      "expiryTime": null,
      "policyMode": null
    },
    "isAppendBlobSealed": null,
    "isCurrentVersion": null,
    "lastAccessedOn": null,
    "metadata": {},
    "name": "https:/chbksa.blob.core.windows.net/chbkcontainer/2022-
     ↪  05-06T12-41-19/metadata.json",
    "objectReplicationDestinationPolicy": null,
    "objectReplicationSourceProperties": [],
    "properties": {
      "appendBlobCommittedBlockCount": null,
      "blobTier": "Hot",
      "blobTierChangeTime": null,
      "blobTierInferred": true,
      "blobType": "BlockBlob",
      "contentLength": 518,
      "contentRange": null,
      "contentSettings": {
        "cacheControl": null,
        "contentDisposition": null,
        "contentEncoding": null,
        "contentLanguage": null,
        "contentMd5": null,
        "contentType": "application/octet-stream"
      },
      "copy": {
        "completionTime": null,
        "destinationSnapshot": null,
        "id": null,
        "incrementalCopy": null,
        "progress": null,
        "source": null,
        "status": null,
        "statusDescription": null
      },
      "creationTime": "2022-05-06T12:41:40+00:00",
      "deletedTime": "2022-05-07T12:27:14+00:00",
      "etag": "0x8DA2F5DC4E522F7",
      "lastModified": "2022-05-06T12:41:40+00:00",
      "lease": {
        "duration": null,
        "state": null,
        "status": null
      },
      "pageBlobSequenceNumber": null,
      "pageRanges": null,
      "rehydrationStatus": null,
      "remainingRetentionDays": 6,
      "serverEncrypted": true
    },
    "rehydratePriority": null,
    "requestServerEncrypted": null,
    "snapshot": null,
    "tagCount": null,
    "tags": null,
    "versionId": null
  },
  {
    "container": "chbkcontainer",
    "content": "",
    "deleted": true,
    "encryptedMetadata": null,
    "encryptionKeySha256": null,
    "encryptionScope": null,
    "hasLegalHold": null,
    "hasVersionsOnly": null,
    "immutabilityPolicy": {
      "expiryTime": null,
      "policyMode": null
    },
    "isAppendBlobSealed": null,
    "isCurrentVersion": null,
    "lastAccessedOn": null,
    "metadata": {},
    "name": "https:/chbksa.blob.core.windows.net/chbkcontainer/2022-
     ↪  05-06T12-41-19/metadata/default/cell_towers.json",
    "objectReplicationDestinationPolicy": null,
    "objectReplicationSourceProperties": [],
    "properties": {
      "appendBlobCommittedBlockCount": null,
      "blobTier": "Hot",
      "blobTierChangeTime": null,
      "blobTierInferred": true,
      "blobType": "BlockBlob",
      "contentLength": 888,
      "contentRange": null,
      "contentSettings": {
        "cacheControl": null,
        "contentDisposition": null,
        "contentEncoding": null,
        "contentLanguage": null,
        "contentMd5": null,
        "contentType": "application/octet-stream"
      },
      "copy": {
        "completionTime": null,
        "destinationSnapshot": null,
        "id": null,
        "incrementalCopy": null,
        "progress": null,
        "source": null,
        "status": null,
        "statusDescription": null
      },
      "creationTime": "2022-05-06T12:41:40+00:00",
      "deletedTime": "2022-05-07T12:27:14+00:00",
      "etag": "0x8DA2F5DC4E23D31",
      "lastModified": "2022-05-06T12:41:40+00:00",
      "lease": {
        "duration": null,
        "state": null,
        "status": null
      },
      "pageBlobSequenceNumber": null,
      "pageRanges": null,
      "rehydrationStatus": null,
      "remainingRetentionDays": 6,
      "serverEncrypted": true
```

```
  },
  "rehydratePriority": null,
  "requestServerEncrypted": null,
  "snapshot": null,
  "tagCount": null,
  "tags": null,
  "versionId": null
},
{
  "container": "chbkcontainer",
  "content": "",
  "deleted": true,
  "encryptedMetadata": null,
  "encryptionKeySha256": null,
  "encryptionScope": null,
  "hasLegalHold": null,
  "hasVersionsOnly": null,
  "immutabilityPolicy": {
    "expiryTime": null,
    "policyMode": null
  },
  "isAppendBlobSealed": null,
  "isCurrentVersion": null,
  "lastAccessedOn": null,
  "metadata": {},
  "name": "https:/chbksa.blob.core.windows.net/chbkcontainer/2022-
   ↪  05-06T12-41-19/shadow/default/cell_towers/default_all_1_35_
   ↪  2.tar",
  "objectReplicationDestinationPolicy": null,
  "objectReplicationSourceProperties": [],
  "properties": {
    "appendBlobCommittedBlockCount": null,
    "blobTier": "Hot",
    "blobTierChangeTime": null,
    "blobTierInferred": true,
    "blobType": "BlockBlob",
    "contentLength": 940539392,
    "contentRange": null,
    "contentSettings": {
      "cacheControl": null,
      "contentDisposition": null,
      "contentEncoding": null,
      "contentLanguage": null,
      "contentMd5": null,
        "contentType": "application/octet-stream"
      },
      "copy": {
        "completionTime": null,
        "destinationSnapshot": null,
        "id": null,
        "incrementalCopy": null,
        "progress": null,
        "source": null,
        "status": null,
        "statusDescription": null
      },
      "creationTime": "2022-05-06T12:41:36+00:00",
      "deletedTime": "2022-05-07T12:27:14+00:00",
      "etag": "0x8DA2F5DC241DE8B",
      "lastModified": "2022-05-06T12:41:36+00:00",
      "lease": {
        "duration": null,
        "state": null,
        "status": null
      },
      "pageBlobSequenceNumber": null,
      "pageRanges": null,
      "rehydrationStatus": null,
      "remainingRetentionDays": 6,
      "serverEncrypted": true
    },
    "rehydratePriority": null,
    "requestServerEncrypted": null,
    "snapshot": null,
    "tagCount": null,
    "tags": null,
    "versionId": null
  },
  {
    "container": "chbkcontainer",
    "content": "",
    "deleted": true,
    "encryptedMetadata": null,
    "encryptionKeySha256": null,
    "encryptionScope": null,
    "hasLegalHold": null,
    "hasVersionsOnly": null,
    "immutabilityPolicy": {
```

```
    "expiryTime": null,
    "policyMode": null
  },
  "isAppendBlobSealed": null,
  "isCurrentVersion": null,
  "lastAccessedOn": null,
  "metadata": {},
  "name": "https:/chbksa.blob.core.windows.net/chbkcontainer/2022-
   ↪  05-06T12-41-19/shadow/default/cell_towers/default_all_36_41
   ↪  _1.tar",
  "objectReplicationDestinationPolicy": null,
  "objectReplicationSourceProperties": [],
  "properties": {
    "appendBlobCommittedBlockCount": null,
    "blobTier": "Hot",
    "blobTierChangeTime": null,
    "blobTierInferred": true,
    "blobType": "BlockBlob",
    "contentLength": 201757696,
    "contentRange": null,
    "contentSettings": {
      "cacheControl": null,
      "contentDisposition": null,
      "contentEncoding": null,
      "contentLanguage": null,
      "contentMd5": null,
      "contentType": "application/octet-stream"
    },
    "copy": {
      "completionTime": null,
      "destinationSnapshot": null,
      "id": null,
      "incrementalCopy": null,
      "progress": null,
      "source": null,
      "status": null,
      "statusDescription": null
    },
    "creationTime": "2022-05-06T12:41:40+00:00",
    "deletedTime": "2022-05-07T12:27:14+00:00",
    "etag": "0x8DA2F5DC4B0D44C",
    "lastModified": "2022-05-06T12:41:40+00:00",
    "lease": {
      "duration": null,
        "state": null,
        "status": null
      },
      "pageBlobSequenceNumber": null,
      "pageRanges": null,
      "rehydrationStatus": null,
      "remainingRetentionDays": 6,
      "serverEncrypted": true
    },
    "rehydratePriority": null,
    "requestServerEncrypted": null,
    "snapshot": null,
    "tagCount": null,
    "tags": null,
    "versionId": null
  },
  {
    "container": "chbkcontainer",
    "content": "",
    "deleted": true,
    "encryptedMetadata": null,
    "encryptionKeySha256": null,
    "encryptionScope": null,
    "hasLegalHold": null,
    "hasVersionsOnly": null,
    "immutabilityPolicy": {
      "expiryTime": null,
      "policyMode": null
    },
    "isAppendBlobSealed": null,
    "isCurrentVersion": null,
    "lastAccessedOn": null,
    "metadata": {},
    "name": "https:/chbksa.blob.core.windows.net/chbkcontainer/2022-
    ↪  05-06T12-41-19/shadow/default/cell_towers/default_all_42_42⌋
    ↪  _0.tar",
    "objectReplicationDestinationPolicy": null,
    "objectReplicationSourceProperties": [],
    "properties": {
      "appendBlobCommittedBlockCount": null,
      "blobTier": "Hot",
      "blobTierChangeTime": null,
      "blobTierInferred": true,
      "blobType": "BlockBlob",
      "contentLength": 8671232,
      "contentRange": null,
      "contentSettings": {
        "cacheControl": null,
        "contentDisposition": null,
        "contentEncoding": null,
        "contentLanguage": null,
        "contentMd5": null,
        "contentType": "application/octet-stream"
      },
      "copy": {
        "completionTime": null,
        "destinationSnapshot": null,
        "id": null,
        "incrementalCopy": null,
        "progress": null,
        "source": null,
        "status": null,
        "statusDescription": null
      },
      "creationTime": "2022-05-06T12:41:40+00:00",
      "deletedTime": "2022-05-07T12:27:14+00:00",
      "etag": "0x8DA2F5DC4DEE251",
      "lastModified": "2022-05-06T12:41:40+00:00",
      "lease": {
        "duration": null,
        "state": null,
        "status": null
      },
      "pageBlobSequenceNumber": null,
      "pageRanges": null,
      "rehydrationStatus": null,
      "remainingRetentionDays": 6,
      "serverEncrypted": true
    },
    "rehydratePriority": null,
    "requestServerEncrypted": null,
    "snapshot": null,
    "tagCount": null,
    "tags": null,
    "versionId": null
  }
]
```

Undelete remote backups via Azure CLI

Undeleting the deleted backups:

```
# List all blobs and retrieve deleted blobs from list
$Blobs = az storage blob list --account-name $SAName
 ↪  --container-name $ContainerName --account-key $AccKey --include
 ↪  d | ConvertFrom-Json -AsHashtable
$DeletedBlobs = $($Blobs | Where-Object { $_.deleted })

# Undelete deleted blobs
foreach($Blob in $DeletedBlobs) {
    az storage blob undelete --container-name $ContainerName --name
     ↪  $Blob.name --account-key $AccKey --account-name $SAName
}
```

Output:

```
{
  "undeleted": null
}
{
  "undeleted": null
}
{
  "undeleted": null
}
{
  "undeleted": null
}
{
  "undeleted": null
}
```

Verify results

Listing the Blobs in the Azure Cloud Shell:

```
az storage blob list --account-name $SAName --container-name
 ↪  $ContainerName --account-key $AccKey --include d
```

Output (notice that `deleted` is `null`):

```
[
  {
    "container": "chbkcontainer",
    "content": "",
  "deleted": null,
  "encryptedMetadata": null,
  "encryptionKeySha256": null,
  "encryptionScope": null,
  "hasLegalHold": null,
  "hasVersionsOnly": null,
  "immutabilityPolicy": {
    "expiryTime": null,
    "policyMode": null
  },
  "isAppendBlobSealed": null,
  "isCurrentVersion": null,
  "lastAccessedOn": null,
  "metadata": {},
  "name": "https:/chbksa.blob.core.windows.net/chbkcontainer/2022-
   ↪  05-06T12-41-19/metadata.json",
  "objectReplicationDestinationPolicy": null,
  "objectReplicationSourceProperties": [],
  "properties": {
    "appendBlobCommittedBlockCount": null,
    "blobTier": "Hot",
    "blobTierChangeTime": null,
    "blobTierInferred": true,
    "blobType": "BlockBlob",
    "contentLength": 518,
    "contentRange": null,
    "contentSettings": {
      "cacheControl": null,
      "contentDisposition": null,
      "contentEncoding": null,
      "contentLanguage": null,
      "contentMd5": null,
      "contentType": "application/octet-stream"
    },
    "copy": {
      "completionTime": null,
      "destinationSnapshot": null,
      "id": null,
      "incrementalCopy": null,
      "progress": null,
      "source": null,
      "status": null,
      "statusDescription": null
    },
      "creationTime": "2022-05-06T12:41:40+00:00",
      "deletedTime": null,
      "etag": "0x8DA2F5DC4E522F7",
      "lastModified": "2022-05-06T12:41:40+00:00",
      "lease": {
        "duration": null,
        "state": "available",
        "status": "unlocked"
      },
      "pageBlobSequenceNumber": null,
      "pageRanges": null,
      "rehydrationStatus": null,
      "remainingRetentionDays": null,
      "serverEncrypted": true
    },
    "rehydratePriority": null,
    "requestServerEncrypted": null,
    "snapshot": null,
    "tagCount": null,
    "tags": null,
    "versionId": null
  },
  {
    "container": "chbkcontainer",
    "content": "",
    "deleted": null,
    "encryptedMetadata": null,
    "encryptionKeySha256": null,
    "encryptionScope": null,
    "hasLegalHold": null,
    "hasVersionsOnly": null,
    "immutabilityPolicy": {
      "expiryTime": null,
      "policyMode": null
    },
    "isAppendBlobSealed": null,
    "isCurrentVersion": null,
    "lastAccessedOn": null,
    "metadata": {},
    "name": "https:/chbksa.blob.core.windows.net/chbkcontainer/2022-
     ↪  05-06T12-41-19/metadata/default/cell_towers.json",
    "objectReplicationDestinationPolicy": null,
    "objectReplicationSourceProperties": [],
    "properties": {
```

```
    "appendBlobCommittedBlockCount": null,
    "blobTier": "Hot",
    "blobTierChangeTime": null,
    "blobTierInferred": true,
    "blobType": "BlockBlob",
    "contentLength": 888,
    "contentRange": null,
    "contentSettings": {
      "cacheControl": null,
      "contentDisposition": null,
      "contentEncoding": null,
      "contentLanguage": null,
      "contentMd5": null,
      "contentType": "application/octet-stream"
    },
    "copy": {
      "completionTime": null,
      "destinationSnapshot": null,
      "id": null,
      "incrementalCopy": null,
      "progress": null,
      "source": null,
      "status": null,
      "statusDescription": null
    },
    "creationTime": "2022-05-06T12:41:40+00:00",
    "deletedTime": null,
    "etag": "0x8DA2F5DC4E23D31",
    "lastModified": "2022-05-06T12:41:40+00:00",
    "lease": {
      "duration": null,
      "state": "available",
      "status": "unlocked"
    },
    "pageBlobSequenceNumber": null,
    "pageRanges": null,
    "rehydrationStatus": null,
    "remainingRetentionDays": null,
    "serverEncrypted": true
  },
  "rehydratePriority": null,
  "requestServerEncrypted": null,
  "snapshot": null,
  "tagCount": null,
    "tags": null,
    "versionId": null
  },
  {
    "container": "chbkcontainer",
    "content": "",
    "deleted": null,
    "encryptedMetadata": null,
    "encryptionKeySha256": null,
    "encryptionScope": null,
    "hasLegalHold": null,
    "hasVersionsOnly": null,
    "immutabilityPolicy": {
      "expiryTime": null,
      "policyMode": null
    },
    "isAppendBlobSealed": null,
    "isCurrentVersion": null,
    "lastAccessedOn": null,
    "metadata": {},
    "name": "https:/chbksa.blob.core.windows.net/chbkcontainer/2022-
     ↪  05-06T12-41-19/shadow/default/cell_towers/default_all_1_35_
     ↪  2.tar",
    "objectReplicationDestinationPolicy": null,
    "objectReplicationSourceProperties": [],
    "properties": {
      "appendBlobCommittedBlockCount": null,
      "blobTier": "Hot",
      "blobTierChangeTime": null,
      "blobTierInferred": true,
      "blobType": "BlockBlob",
      "contentLength": 940539392,
      "contentRange": null,
      "contentSettings": {
        "cacheControl": null,
        "contentDisposition": null,
        "contentEncoding": null,
        "contentLanguage": null,
        "contentMd5": null,
        "contentType": "application/octet-stream"
      },
      "copy": {
        "completionTime": null,
        "destinationSnapshot": null,
        "id": null,
        "incrementalCopy": null,
        "progress": null,
        "source": null,
        "status": null,
        "statusDescription": null
      },
      "creationTime": "2022-05-06T12:41:36+00:00",
      "deletedTime": null,
      "etag": "0x8DA2F5DC241DE8B",
      "lastModified": "2022-05-06T12:41:36+00:00",
      "lease": {
        "duration": null,
        "state": "available",
        "status": "unlocked"
      },
      "pageBlobSequenceNumber": null,
      "pageRanges": null,
      "rehydrationStatus": null,
      "remainingRetentionDays": null,
      "serverEncrypted": true
    },
    "rehydratePriority": null,
    "requestServerEncrypted": null,
    "snapshot": null,
    "tagCount": null,
    "tags": null,
    "versionId": null
  },
  {
    "container": "chbkcontainer",
    "content": "",
    "deleted": null,
    "encryptedMetadata": null,
    "encryptionKeySha256": null,
    "encryptionScope": null,
    "hasLegalHold": null,
    "hasVersionsOnly": null,
    "immutabilityPolicy": {
      "expiryTime": null,
      "policyMode": null
    },
    "isAppendBlobSealed": null,
    "isCurrentVersion": null,
    "lastAccessedOn": null,
    "metadata": {},
    "name": "https:/chbksa.blob.core.windows.net/chbkcontainer/2022-
    ↪  05-06T12-41-19/shadow/default/cell_towers/default_all_36_41
    ↪  _1.tar",
    "objectReplicationDestinationPolicy": null,
    "objectReplicationSourceProperties": [],
    "properties": {
      "appendBlobCommittedBlockCount": null,
      "blobTier": "Hot",
      "blobTierChangeTime": null,
      "blobTierInferred": true,
      "blobType": "BlockBlob",
      "contentLength": 201757696,
      "contentRange": null,
      "contentSettings": {
        "cacheControl": null,
        "contentDisposition": null,
        "contentEncoding": null,
        "contentLanguage": null,
        "contentMd5": null,
        "contentType": "application/octet-stream"
      },
      "copy": {
        "completionTime": null,
        "destinationSnapshot": null,
        "id": null,
        "incrementalCopy": null,
        "progress": null,
        "source": null,
        "status": null,
        "statusDescription": null
      },
      "creationTime": "2022-05-06T12:41:40+00:00",
      "deletedTime": null,
      "etag": "0x8DA2F5DC4B0D44C",
      "lastModified": "2022-05-06T12:41:40+00:00",
      "lease": {
        "duration": null,
        "state": "available",
        "status": "unlocked"
      },
      "pageBlobSequenceNumber": null,
      "pageRanges": null,
      "rehydrationStatus": null,
      "remainingRetentionDays": null,
      "serverEncrypted": true
    },
    "rehydratePriority": null,
    "requestServerEncrypted": null,
    "snapshot": null,
    "tagCount": null,
    "tags": null,
    "versionId": null
  },
  {
    "container": "chbkcontainer",
    "content": "",
    "deleted": null,
    "encryptedMetadata": null,
    "encryptionKeySha256": null,
    "encryptionScope": null,
    "hasLegalHold": null,
    "hasVersionsOnly": null,
    "immutabilityPolicy": {
      "expiryTime": null,
      "policyMode": null
    },
    "isAppendBlobSealed": null,
    "isCurrentVersion": null,
    "lastAccessedOn": null,
    "metadata": {},
    "name": "https:/chbksa.blob.core.windows.net/chbkcontainer/2022-
    ↪  05-06T12-41-19/shadow/default/cell_towers/default_all_42_42
    ↪  _0.tar",
    "objectReplicationDestinationPolicy": null,
    "objectReplicationSourceProperties": [],
    "properties": {
      "appendBlobCommittedBlockCount": null,
      "blobTier": "Hot",
      "blobTierChangeTime": null,
      "blobTierInferred": true,
      "blobType": "BlockBlob",
      "contentLength": 8671232,
      "contentRange": null,
      "contentSettings": {
        "cacheControl": null,
        "contentDisposition": null,
        "contentEncoding": null,
        "contentLanguage": null,
        "contentMd5": null,
        "contentType": "application/octet-stream"
      },
      "copy": {
        "completionTime": null,
        "destinationSnapshot": null,
        "id": null,
        "incrementalCopy": null,
        "progress": null,
        "source": null,
        "status": null,
        "statusDescription": null
      },
      "creationTime": "2022-05-06T12:41:40+00:00",
      "deletedTime": null,
      "etag": "0x8DA2F5DC4DEE251",
      "lastModified": "2022-05-06T12:41:40+00:00",
      "lease": {
        "duration": null,
        "state": "available",
        "status": "unlocked"
      },
      "pageBlobSequenceNumber": null,
      "pageRanges": null,
      "rehydrationStatus": null,
      "remainingRetentionDays": null,
      "serverEncrypted": true
    },
    "rehydratePriority": null,
    "requestServerEncrypted": null,
    "snapshot": null,
    "tagCount": null,
    "tags": null,
    "versionId": null
  }
]
```

Listing undeleted Blobs in `clickhouse-backup`:

```
sudo ./clickhouse-backup list remote --config config.yaml
# 2022/05/07 12:26:27.136458  info SELECT max(toInt64(bytes_on_disk
 ↪  * 1.02)) AS max_file_size FROM system.parts
```

```
# 2022-05-06T12-41-19   1.07GiB   06/05/2022 12:41:40   remote
 ↪  tar
```

Restoring the database from the undeleted remote backup:

```
sudo ./clickhouse-backup restore_remote 2022-05-06T12-41-19
 ↪  --config config.yaml
# 2022/05/07 12:30:32.003711  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/07 12:30:32.007101  info SELECT * FROM system.disks;
# 2022/05/07 12:30:32.018499  info SELECT max(toInt64(bytes_on_disk
 ↪  * 1.02)) AS max_file_size FROM system.parts
# 2022/05/07 12:30:32.094614  info done
 ↪  backup=2022-05-06T12-41-19 duration=7ms operation=download
 ↪  size=765B table_metadata=default.cell_towers
# 2022/05/07 12:30:51.137543  info done
 ↪  diff_parts=0 duration=0s operation=downloadDiffParts100.00% 0s
# 2022/05/07 12:30:51.137700  info done
 ↪  backup=2022-05-06T12-41-19 duration=19.043s
 ↪  operation=download_data size=1.07GiB table=default.cell_towers
# 2022/05/07 12:30:51.146800  info done
 ↪  backup=2022-05-06T12-41-19 duration=19.135s operation=download
 ↪  size=1.07GiB
# 2022/05/07 12:30:52.193726  info SELECT value FROM
 ↪  `system`.`build_options` where name='VERSION_INTEGER'
# 2022/05/07 12:30:53.438041  info SELECT * FROM system.disks;
# 2022/05/07 12:30:53.441231  info CREATE DATABASE IF NOT EXISTS
 ↪  default
# ENGINE = Atomic
# 2022/05/07 12:30:53.458792  info SELECT engine FROM
 ↪  system.databases WHERE name = 'default'
# 2022/05/07 12:30:53.483831  info DROP TABLE IF EXISTS
 ↪  `default`.`cell_towers` NO DELAY
# 2022/05/07 12:30:53.509936  info CREATE DATABASE IF NOT EXISTS
 ↪  `default`
# 2022/05/07 12:30:53.517094  info CREATE TABLE default.cell_towers
 ↪  UUID 'f9b9b44f-3af6-4bf5-9458-9ef6a81a3519' (`radio` Enum8('' =
 ↪  0, 'CDMA' = 1, 'GSM' = 2, 'LTE' = 3, 'NR' = 4, 'UMTS' = 5),
 ↪  `mcc` UInt16, `net` UInt16, `area` UInt16, `cell` UInt64,
 ↪  `unit` Int16, `lon` Float64, `lat` Float64, `range` UInt32,
 ↪  `samples` UInt32, `changeable` UInt8, `created` DateTime,
 ↪  `updated` DateTime, `averageSignal` UInt8) ENGINE = MergeTree
 ↪  ORDER BY (radio, mcc, net, created) SETTINGS index_granularity
 ↪  = 8192
```

```
# 2022/05/07 12:30:53.764502  info SELECT count() FROM
↪  system.settings WHERE name =
↪  'show_table_uuid_in_table_create_query_if_not_nil'
# 2022/05/07 12:30:53.771475  info SELECT name FROM
↪  system.databases WHERE engine IN ('MySQL','PostgreSQL')
# 2022/05/07 12:30:53.780565  info
#                 SELECT
#                         countIf(name='data_path')
↪  is_data_path_present,
#                         countIf(name='data_paths')
↪  is_data_paths_present,
#                         countIf(name='uuid') is_uuid_present,
#                         countIf(name='create_table_query')
↪  is_create_table_query_present,
#                         countIf(name='total_bytes')
↪  is_total_bytes_present
#                 FROM system.columns WHERE database='system' AND
↪  table='tables'
#
# 2022/05/07 12:30:53.791797  info SELECT database, name, engine ,
↪  data_paths , uuid , create_table_query , coalesce(total_bytes,
↪  0) AS total_bytes   FROM system.tables WHERE is_temporary = 0
↪  SETTINGS show_table_uuid_in_table_create_query_if_not_nil=1
# 2022/05/07 12:30:53.816765  info SELECT sum(bytes_on_disk) as
↪  size FROM system.parts WHERE database='default' AND
↪  table='cell_towers' GROUP BY database, table
# 2022/05/07 12:30:53.832002  info ALTER TABLE
↪  `default`.`cell_towers` ATTACH PART 'all_1_35_2'
# 2022/05/07 12:30:53.844536  info ALTER TABLE
↪  `default`.`cell_towers` ATTACH PART 'all_36_41_1'
# 2022/05/07 12:30:53.849068  info ALTER TABLE
↪  `default`.`cell_towers` ATTACH PART 'all_42_42_0'
# 2022/05/07 12:30:53.852548  info done
↪  backup=2022-05-06T12-41-19 operation=restore
↪  table=default.cell_towers
# 2022/05/07 12:30:53.852691  info done
↪  backup=2022-05-06T12-41-19 duration=94ms operation=restore
# 2022/05/07 12:30:53.852822  info done
↪  backup=2022-05-06T12-41-19 operation=restore
```

The test queries were repeated in `clickhouse-client`. The queries were hanging at first, so the VM was restarted. Both were successful.

Test query 1:

```
SELECT
    radio,
    count() AS c
FROM cell_towers
GROUP BY radio
ORDER BY c DESC
FORMAT TabSeparated

-- Query id: 4d363e44-943b-47c0-b926-abd323e4dd59
--
-- UMTS        20686487
-- LTE        12101148
-- GSM        9931312
-- CDMA        556344
-- NR        867
--
-- 5 rows in set. Elapsed: 0.055 sec. Processed 43.28 million rows,
↪  43.28 MB (781.15 million rows/s., 781.15 MB/s.)
```

Test query 2:

```
SELECT
    mcc,
    count()
FROM cell_towers
GROUP BY mcc
ORDER BY count() DESC
LIMIT 10
FORMAT TabSeparated

-- Query id: ca9261f1-bb07-44f1-b83d-c95aab614203
--
-- 310        5024650
-- 262        2622423
-- 250        1953176
-- 208        1891187
-- 724        1836150
-- 404        1729151
-- 234        1618924
-- 510        1353998
-- 440        1343355
-- 311        1332798
--
-- 10 rows in set. Elapsed: 0.197 sec. Processed 43.28 million
↪  rows, 86.55 MB (220.03 million rows/s., 440.05 MB/s.)
```

A.8 Delete backups stored in Azure blob storage via Azure CLI

This experiment is similar to the previous one (A.7), except that the blobs were deleted via the Azure CLI, instead of using `clickhouse-backup`. Afterwards, the Blobs were restored.

A.8.1 Preparation

The following variables were declared:

```
$AccKey = "NdbW07WlHBf5zcpMXundwkP88Ie2SO1Ad+84VD8moaUg1ihIeRR7cEdy⌋
 ↪  4FXIgHRvIQwPIMc7eD2q+ASt6EqxWg==" # Storage Account Access
 ↪  Key
$StorageAccount = Get-AzStorageAccount | Where-Object {
 ↪  $_.StorageAccountName -eq $SAName} # clickhouse-backup storage
 ↪  account
```

A.8.2 Delete the blobs

Listing the blobs in Azure Cloud Shell:

```
az storage blob list --account-name $SAName --container-name
 ↪  $ContainerName --account-key $AccKey | ConvertFrom-Json
 ↪  -AsHashtable | % {$_.name}
# https:/chbksa.blob.core.windows.net/chbkcontainer/2022-05-06T12-4⌋
 ↪  1-19/metadata.json
# https:/chbksa.blob.core.windows.net/chbkcontainer/2022-05-06T12-4⌋
 ↪  1-19/metadata/default/cell_towers.json
# https:/chbksa.blob.core.windows.net/chbkcontainer/2022-05-06T12-4⌋
 ↪  1-19/shadow/default/cell_towers/default_all_1_35_2.tar
# https:/chbksa.blob.core.windows.net/chbkcontainer/2022-05-06T12-4⌋
 ↪  1-19/shadow/default/cell_towers/default_all_36_41_1.tar
# https:/chbksa.blob.core.windows.net/chbkcontainer/2022-05-06T12-4⌋
 ↪  1-19/shadow/default/cell_towers/default_all_42_42_0.tar
```

Deleting the blobs:

```
# Store the list of blob names in a variable
$BlobNames = az storage blob list --account-name $SAName
 ↪  --container-name $ContainerName --account-key $AccKey |
 ↪  ConvertFrom-Json -AsHashtable | % {$_.name}

foreach($name in $BlobNames) {
    echo "Deleting $name"
    az storage blob delete --account-name $SAName --account-key
     ↪  $AccKey --container-name $ContainerName --name $name
}
# Deleting https:/chbksa.blob.core.windows.net/chbkcontainer/2022-0⌋
 ↪  5-06T12-41-19/metadata.json
# Deleting https:/chbksa.blob.core.windows.net/chbkcontainer/2022-0⌋
 ↪  5-06T12-41-19/metadata/default/cell_towers.json
# Deleting https:/chbksa.blob.core.windows.net/chbkcontainer/2022-0⌋
 ↪  5-06T12-41-19/shadow/default/cell_towers/default_all_1_35_2.tar
# Deleting https:/chbksa.blob.core.windows.net/chbkcontainer/2022-0⌋
 ↪  5-06T12-41-19/shadow/default/cell_towers/default_all_36_41_1.tar
# Deleting https:/chbksa.blob.core.windows.net/chbkcontainer/2022-0⌋
 ↪  5-06T12-41-19/shadow/default/cell_towers/default_all_42_42_0.tar
```

After this, the blobs were no longer visible in the Azure portal (unless “Show deleted blobs” was enabled). We decided to also verify this via `clickhouse-backup`.

Trying to list the blobs via `clickhouse-backup` (no result):

```
sudo ./clickhouse-backup list remote --config config.yaml
# 2022/05/07 12:54:09.423781  info SELECT max(toInt64(bytes_on_disk
 ↪  * 1.02)) AS max_file_size FROM system.parts
```

A.8.3 Restore the blobs

The process for restoring blobs is the same as in section A.7.5.

Script to undelete deleted blobs:

```
# List all blobs and retrieve deleted blobs from list
$Blobs = az storage blob list --account-name $SAName
 ↪  --container-name $ContainerName --account-key $AccKey --include
 ↪  d | ConvertFrom-Json -AsHashtable
$DeletedBlobs = $($Blobs | Where-Object { $_.deleted })

# Undelete deleted blobs
foreach($Blob in $DeletedBlobs) {
    az storage blob undelete --container-name $ContainerName --name
     ↪  $Blob.name --account-key $AccKey --account-name $SAName
}
```

Listing the blobs:

```
az storage blob list --account-name $SAName --container-name
 ↪  $ContainerName --account-key $AccKey | ConvertFrom-Json
 ↪  -AsHashtable | % {$_.name}
# https:/chbksa.blob.core.windows.net/chbkcontainer/2022-05-06T12-4⌋
 ↪  1-19/metadata.json
```

```
# https:/chbksa.blob.core.windows.net/chbkcontainer/2022-05-06T12-4
↪  1-19/metadata/default/cell_towers.json
# https:/chbksa.blob.core.windows.net/chbkcontainer/2022-05-06T12-4
↪  1-19/shadow/default/cell_towers/default_all_1_35_2.tar
# https:/chbksa.blob.core.windows.net/chbkcontainer/2022-05-06T12-4
↪  1-19/shadow/default/cell_towers/default_all_36_41_1.tar
# https:/chbksa.blob.core.windows.net/chbkcontainer/2022-05-06T12-4
↪  1-19/shadow/default/cell_towers/default_all_42_42_0.tar
```

A.9 Delete backups in Azure Backup

In this experiment, backups were stopped for the ClickHouse VM, and the backups were deleted. Then the backups were restored from their soft deleted state, and the backups were resumed.

A.9.1 Delete Azure Backups via CLI

Get the IDs of all backup items:

```
$itemIds = az backup item list --resource-group $RGName
↪  --vault-name $RSVName | ConvertFrom-Json -AsHashtable | %
↪  {$_["id"]}
```

Disable protection for and delete all backup items:

```
foreach($id in $itemIds) {
    az backup protection disable --backup-management-type
     ↪  AzureIaasVM --delete-backup-data true --ids $id
}
```

Output:

```
Are you sure you want to perform this operation? (y/n): y
{
  "eTag": null,
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour
  ↪  ceGroups/testRG/providers/Microsoft.RecoveryServices/vaults/m
  ↪  yRSV/backupJobs/bd6f9157-f5d1-4f67-9435-69737c6ab221",
  "location": null,
  "name": "bd6f9157-f5d1-4f67-9435-69737c6ab221",
  "properties": {
    "actionsInfo": null,
    "activityId": "4054af82-cd28-11ec-8904-0a580af40e30",
    "backupManagementType": "AzureIaasVM",
    "containerName": "iaasvmcontainerv2;testrg;clickhousevm",
    "duration": "0:00:11.138772",
    "endTime": "2022-05-06T10:35:45.276307+00:00",
    "entityFriendlyName": "clickhouseVM",
    "errorDetails": null,
    "extendedInfo": {
      "dynamicErrorMessage": null,
      "estimatedRemainingDuration": null,
      "internalPropertyBag": null,
      "progressPercentage": null,
      "propertyBag": {
        "Number of Recovery Points": "4",
        "VM Name": "clickhouseVM"
      },
      "tasksList": []
    },
    "isUserTriggered": null,
    "jobType": "AzureIaaSVMJob",
    "operation": "DeleteBackupData",
    "startTime": "2022-05-06T10:35:34.137536+00:00",
    "status": "Completed",
    "virtualMachineVersion": "Compute"
  },
  "resourceGroup": "testRG",
  "tags": null,
  "type": "Microsoft.RecoveryServices/vaults/backupJobs"
}
```

This triggered an alert from Azure Monitor (“Delete Backup Data” alert). Listing the backup items:

```
az backup item list --resource-group $RGName --vault-name $RSVName
```

Output:

```
[
  {
    "eTag": null,
    "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/reso⌋
    ↪  urceGroups/testRG/providers/Microsoft.RecoveryServices/vaul⌋
    ↪  ts/myRSV/backupFabrics/Azure/protectionContainers/IaasVMCon⌋
    ↪  tainer;iaasvmcontainerv2;testrg;clickhousevm/protectedItems⌋
    ↪  /VM;iaasvmcontainerv2;testrg;clickhousevm",
    "location": null,
    "name": "VM;iaasvmcontainerv2;testrg;clickhousevm",
    "properties": {
      "backupManagementType": "AzureIaasVM",
      "backupSetName": null,
"containerName": "iaasvmcontainerv2;testrg;clickhousevm",
"createMode": null,
"deferredDeleteTimeInUtc": "2022-05-06T10:35:34.137536+00:00",
"deferredDeleteTimeRemaining": "13.23:48:44.6226505",
"extendedInfo": null,
"extendedProperties": {
  "diskExclusionProperties": null,
  "linuxVmApplicationName": ""
},
"friendlyName": "clickhouseVM",
"healthDetails": [
  {
    "code": 400239,
    "message": "Backup pre-check status of this virtual
    ↪  machine is OK.",
    "recommendations": [],
    "title": "IaasVmHealthGreenDefault"
  }
],
"healthStatus": "Passed",
"isArchiveEnabled": false,
"isDeferredDeleteScheduleUpcoming": null,
"isRehydrate": null,
"isScheduledForDeferredDelete": true,
"kpisHealths": {
  "BackupOperationStatusKPI": {
    "resourceHealthDetails": [
      {
        "code": 0,
        "message": "",
        "recommendations": [],
        "title": "Success"
      }
    ],
    "resourceHealthStatus": "Healthy"
  },
  "RestoreOperationStatusKPI": {
    "resourceHealthDetails": [
      {
        "code": 0,
        "message": "",
        "recommendations": [],
        "title": "Success"
      }
```

```
          ],
          "resourceHealthStatus": "Healthy"
        }
      },
      "lastBackupStatus": "Completed",
      "lastBackupTime": "2022-05-05T22:41:02.855511+00:00",
      "lastRecoveryPoint": "2022-05-05T22:41:06.408480+00:00",
      "policyId": "",
      "policyName": "",
      "protectedItemDataId": "123146170808998",
      "protectedItemType": "Microsoft.Compute/virtualMachines",
      "protectionState": "ProtectionStopped",
      "protectionStatus": "Healthy",
      "resourceGuardOperationRequests": null,
      "sourceResourceId": "/subscriptions/4b48eb85-91f3-4902-b74b-e
       ↪  84641fb6785/resourceGroups/testRG/providers/Microsoft.Com
       ↪  pute/virtualMachines/clickhouseVM",
      "virtualMachineId": "/subscriptions/4b48eb85-91f3-4902-b74b-e
       ↪  84641fb6785/resourceGroups/testRG/providers/Microsoft.Com
       ↪  pute/virtualMachines/clickhouseVM",
      "workloadType": "VM"
    },
    "resourceGroup": "testRG",
    "tags": null,
    "type": "Microsoft.RecoveryServices/vaults/backupFabrics/protec
     ↪  tionContainers/protectedItems"
  }
]
```

Notice that in the above list of backup items `protectionState` is set to `ProtectionStopped`, but the backup item still exists. Because of soft delete, it will be retained for 14 days before being permanently deleted.

A.9.2 Undelete soft deleted backup items

Undelete the backup item (the script assumes there is only one backup item in the vault, and that it is deleted):

```
# Store the deleted backup
$DeletedBackup = az backup item list --resource-group $RGName
 ↪  --vault-name $RSVName | ConvertFrom-Json -AsHashtable

# Store the container
$Container = az backup container list --resource-group $RGName
 ↪  --vault-name $RSVName --backup-management-type AzureIaasVM |
 ↪  ConvertFrom-Json -AsHashtable
```

```
# Undelete the deleted backup
az backup protection undelete --container-name $Container.name
 ↪  --item-name $DeletedBackup.name --resource-group $RGName
 ↪  --vault-name $RSVName --backup-management-type AzureIaasVM
 ↪  --workload-type VM
```

Output:

```
{
  "eTag": null,
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
   ↪  ceGroups/testRG/providers/Microsoft.RecoveryServices/vaults/m⌋
   ↪  yRSV/backupJobs/5b7ed38c-051b-419e-9257-5a709bd17d91",
  "location": null,
  "name": "5b7ed38c-051b-419e-9257-5a709bd17d91",
  "properties": {
    "actionsInfo": null,
    "activityId": "13481c16-ceba-11ec-8f9e-0a580af44d46",
    "backupManagementType": "AzureIaasVM",
    "containerName": "iaasvmcontainerv2;testrg;clickhousevm",
    "duration": "0:00:11.188451",
    "endTime": "2022-05-08T10:32:05.635967+00:00",
    "entityFriendlyName": "clickhouseVM",
    "errorDetails": null,
    "extendedInfo": {
      "dynamicErrorMessage": null,
      "estimatedRemainingDuration": null,
      "internalPropertyBag": null,
      "progressPercentage": null,
      "propertyBag": {
        "Undelete flag": "True",
        "VM Name": "clickhouseVM"
      },
      "tasksList": []
    },
    "isUserTriggered": null,
    "jobType": "AzureIaaSVMJob",
    "operation": "Undelete",
    "startTime": "2022-05-08T10:31:54.447516+00:00",
    "status": "Completed",
    "virtualMachineVersion": "Compute"
  },
  "resourceGroup": "testRG",
  "tags": null,
  "type": "Microsoft.RecoveryServices/vaults/backupJobs"
}
```

A.9.3 Re-enable backup

In order to delete the backup, backup had to be disabled for the VM. Now it is time to re-enable backup for the VM.

Enable backup for VM (script assumes there is one VM):

```
# Get the backup items
$itemId = az backup item list --resource-group $RGName --vault-name
 ↪  $RSVName | ConvertFrom-Json -AsHashtable | % {$_["id"]}

# Resume backup protection
az backup protection resume --policy-name $PolicyName --ids $itemId
```

Output:

```
{
  "eTag": null,
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
   ↪  ceGroups/testRG/providers/Microsoft.RecoveryServices/vaults/m⌋
   ↪  yRSV/backupJobs/5b12880d-3229-4fb6-82b8-273cbfee84e1",
  "location": null,
  "name": "5b12880d-3229-4fb6-82b8-273cbfee84e1",
  "properties": {
    "actionsInfo": null,
    "activityId": "f8984456-cebc-11ec-bebc-0a580af44d46",
    "backupManagementType": "AzureIaasVM",
    "containerName": "iaasvmcontainerv2;testrg;clickhousevm",
    "duration": "0:00:11.105737",
    "endTime": "2022-05-08T10:52:50.612509+00:00",
    "entityFriendlyName": "clickhouseVM",
    "errorDetails": null,
    "extendedInfo": {
      "dynamicErrorMessage": null,
      "estimatedRemainingDuration": null,
      "internalPropertyBag": null,
      "progressPercentage": null,
      "propertyBag": {
        "Policy Name": "DefaultPolicy",
        "VM Name": "clickhouseVM"
      },
      "tasksList": []
    },
    "isUserTriggered": null,
    "jobType": "AzureIaaSVMJob",
    "operation": "ConfigureBackup",
    "startTime": "2022-05-08T10:52:39.506772+00:00",
    "status": "Completed",
    "virtualMachineVersion": "Compute"
  },
  "resourceGroup": "testRG",
  "tags": null,
  "type": "Microsoft.RecoveryServices/vaults/backupJobs"
}
```

Verifying the results:

```
az backup item list --resource-group $RGName --vault-name $RSVName
```

Output:

```
[
  {
    "eTag": null,
    "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/reso⌟
    ↪  urceGroups/testRG/providers/Microsoft.RecoveryServices/vaul⌟
    ↪  ts/myRSV/backupFabrics/Azure/protectionContainers/IaasVMCon⌟
    ↪  tainer;iaasvmcontainerv2;testrg;clickhousevm/protectedItems⌟
    ↪  /VM;iaasvmcontainerv2;testrg;clickhousevm",
    "location": null,
    "name": "VM;iaasvmcontainerv2;testrg;clickhousevm",
    "properties": {
      "backupManagementType": "AzureIaasVM",
      "backupSetName": null,
      "containerName": "iaasvmcontainerv2;testrg;clickhousevm",
      "createMode": null,
      "deferredDeleteTimeInUtc": null,
      "deferredDeleteTimeRemaining": null,
      "extendedInfo": null,
      "extendedProperties": {
        "diskExclusionProperties": null,
        "linuxVmApplicationName": ""
      },
      "friendlyName": "clickhouseVM",
      "healthDetails": [
        {
          "code": 400239,
          "message": "Backup pre-check status of this virtual
          ↪  machine is OK.",
          "recommendations": [],
        "title": "IaasVmHealthGreenDefault"
      }
    ],
    "healthStatus": "Passed",
    "isArchiveEnabled": false,
    "isDeferredDeleteScheduleUpcoming": null,
    "isRehydrate": null,
    "isScheduledForDeferredDelete": null,
    "kpisHealths": {
      "BackupOperationStatusKPI": {
        "resourceHealthDetails": [
          {
            "code": 0,
            "message": "",
            "recommendations": [],
            "title": "Success"
          }
        ],
        "resourceHealthStatus": "Healthy"
      },
      "RestoreOperationStatusKPI": {
        "resourceHealthDetails": [
          {
            "code": 0,
            "message": "",
            "recommendations": [],
            "title": "Success"
          }
        ],
        "resourceHealthStatus": "Healthy"
      }
    },
    "lastBackupStatus": "Completed",
    "lastBackupTime": "2022-05-05T22:41:02.855511+00:00",
    "lastRecoveryPoint": "2022-05-05T22:41:06.408480+00:00",
    "policyId": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6⌋
    ↪  785/resourceGroups/testRG/providers/Microsoft.RecoverySer⌋
    ↪  vices/vaults/myRSV/backupPolicies/DefaultPolicy",
    "policyName": "DefaultPolicy",
    "protectedItemDataId": "123146170808998",
    "protectedItemType": "Microsoft.Compute/virtualMachines",
    "protectionState": "Protected",
    "protectionStatus": "Healthy",
    "resourceGuardOperationRequests": null,
      "sourceResourceId": "/subscriptions/4b48eb85-91f3-4902-b74b-e
       ↪  84641fb6785/resourceGroups/testRG/providers/Microsoft.Com
       ↪  pute/virtualMachines/clickhouseVM",
      "virtualMachineId": "/subscriptions/4b48eb85-91f3-4902-b74b-e
       ↪  84641fb6785/resourceGroups/testRG/providers/Microsoft.Com
       ↪  pute/virtualMachines/clickhouseVM",
      "workloadType": "VM"
    },
    "resourceGroup": "testRG",
    "tags": null,
    "type": "Microsoft.RecoveryServices/vaults/backupFabrics/protec
     ↪  tionContainers/protectedItems"
  }
]
```

`protectionState` is `Protected` and `protectionStatus` is `Healthy`. This was also verified in the Azure Portal.

A.10 Disable soft delete and delete backups

In this experiment, soft delete is disabled before deleting backup items. This makes it impossible to recover the deleted backups by undeleting them.

A.10.1 Disable soft delete

Disabling soft delete for the RSV:

```
# Get RSV
$RSV = Get-AzRecoveryServicesVault -Name $RSVName
 ↪  -ResourceGroupName $RGName

Set-AzRecoveryServicesVaultProperty -VaultId $RSV.ID
 ↪  -SoftDeleteFeatureState Disable
# StorageModelType                 :
# StorageType                      :
# StorageTypeState                 :
# EnhancedSecurityState            : Enabled
# SoftDeleteFeatureState           : Disabled
# ResourceGuardOperationRequests   :
# IsSoftDeleteFeatureStateEditable : True
```

A.10.2 Disable protection and delete backup items

Disabling protection and deleting all backup itemes in vault:

```
# List backup items
$itemIds = az backup item list --resource-group $RGName
 ↪  --vault-name $RSVName | ConvertFrom-Json -AsHashtable | %
 ↪  {$_["id"]}

# Delete each item
foreach($id in $itemIds) {
    az backup protection disable --backup-management-type
     ↪  AzureIaasVM --delete-backup-data true --ids $id
}

# Are you sure you want to perform this operation? (y/n): y
```

Output:

```
{
  "eTag": null,
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
   ↪  ceGroups/testRG/providers/Microsoft.RecoveryServices/vaults/m⌋
   ↪  yRSV/backupJobs/e3489fcc-c117-45c1-98e4-1f26521d7486",
  "location": null,
  "name": "e3489fcc-c117-45c1-98e4-1f26521d7486",
  "properties": {
    "actionsInfo": null,
    "activityId": "9a528994-cec4-11ec-b1cb-0a580af44d46",
    "backupManagementType": "AzureIaasVM",
    "containerName": "iaasvmcontainerv2;testrg;clickhousevm",
    "duration": "0:01:51.796137",
    "endTime": "2022-05-08T11:49:08.863143+00:00",
    "entityFriendlyName": "clickhouseVM",
    "errorDetails": null,
    "extendedInfo": {
      "dynamicErrorMessage": null,
      "estimatedRemainingDuration": null,
      "internalPropertyBag": null,
      "progressPercentage": null,
      "propertyBag": {
        "Number of Recovery Points": "4",
        "VM Name": "clickhouseVM"
      },
      "tasksList": []
    },
    "isUserTriggered": null,
    "jobType": "AzureIaaSVMJob",
    "operation": "DeleteBackupData",
    "startTime": "2022-05-08T11:47:17.067006+00:00",
    "status": "Completed",
    "virtualMachineVersion": "Compute"
  },
  "resourceGroup": "testRG",
  "tags": null,
  "type": "Microsoft.RecoveryServices/vaults/backupJobs"
}
```

A.10.3 Verify deletion

Listing all backups in vault:

```
az backup item list --resource-group $RGName --vault-name $RSVName
# []
```

The result is an empty array, meaning there are no backup items to list. The backup item was also gone when viewed from the Azure Portal.

A.11 Delete Recovery Services vault

Microsoft provides a script to delete an entire Recovery Services vault. The script turns off soft delete and deletes all backup items. Backup items in a soft deleted state are first undeleted and then deleted again. In this experiment, we try to run the script, in order to delete the RSV.

A.11.1 Make a backup in the RSV

Since our backups were deleted in the previous experiment, backup protection for the ClickHouse VM had to be re-enabled.

Enabling backup protection for the VM:

```
az backup protection enable-for-vm `
 --resource-group $RGName `
 --vault-name $RSVName `
 --vm $CHName `
 --policy-name $PolicyName
```

Output:

```
{
  "eTag": null,
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour
  ↪  ceGroups/testRG/providers/Microsoft.RecoveryServices/vaults/m
  ↪  yRSV/backupJobs/f1c62078-9756-4407-b97f-8dad9124d6e3",
  "location": null,
  "name": "f1c62078-9756-4407-b97f-8dad9124d6e3",
  "properties": {
    "actionsInfo": null,
    "activityId": "c7b7513a-cece-11ec-a490-0a580af4687d",
    "backupManagementType": "AzureIaasVM",
    "containerName": "iaasvmcontainerv2;testrg;clickhousevm",
    "duration": "0:00:31.147828",
    "endTime": "2022-05-08T13:00:51.346337+00:00",
    "entityFriendlyName": "clickhousevm",
    "errorDetails": null,
    "extendedInfo": {
      "dynamicErrorMessage": null,
      "estimatedRemainingDuration": null,
      "internalPropertyBag": null,
      "progressPercentage": null,
      "propertyBag": {
        "Policy Name": "DefaultPolicy",
        "VM Name": "clickhousevm"
      },
      "tasksList": []
    },
    "isUserTriggered": null,
    "jobType": "AzureIaaSVMJob",
    "operation": "ConfigureBackup",
    "startTime": "2022-05-08T13:00:20.198509+00:00",
    "status": "Completed",
    "virtualMachineVersion": "Compute"
  },
  "resourceGroup": "testRG",
  "tags": null,
  "type": "Microsoft.RecoveryServices/vaults/backupJobs"
}
```

Trigger a backup job immediately:

```
az backup protection backup-now `
    --resource-group $RGName `
    --vault-name $RSVName `
    --container-name $CHName `
    --item-name $CHName `
    --backup-management-type AzureIaaSVM `
    --retain-until 12-05-2022
```

Output:

```
{
  "eTag": null,
"id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
↪  ceGroups/testRG/providers/Microsoft.RecoveryServices/vaults/m⌋
↪  yRSV/backupJobs/5dcacb3b-0c69-4849-a239-5b5bf935cb08",
"location": null,
"name": "5dcacb3b-0c69-4849-a239-5b5bf935cb08",
"properties": {
  "actionsInfo": [
    "1"
  ],
  "activityId": "ee140e7c-cece-11ec-aa04-0a580af4687d",
  "backupManagementType": "AzureIaasVM",
  "containerName": "iaasvmcontainerv2;testrg;clickhousevm",
  "duration": "0:00:01.551140",
  "endTime": null,
  "entityFriendlyName": "clickhousevm",
  "errorDetails": null,
  "extendedInfo": {
    "dynamicErrorMessage": null,
    "estimatedRemainingDuration": null,
    "internalPropertyBag": {
      "IsInstantRpJob": "True"
    },
    "progressPercentage": null,
    "propertyBag": {
      "Recovery Point Expiry Time in UTC": "5/12/2022 12:00:00
       ↪  AM",
      "VM Name": "clickhousevm"
    },
    "tasksList": [
      {
        "duration": "0:00:00",
        "endTime": null,
        "instanceId": null,
        "progressPercentage": null,
        "startTime": null,
        "status": "InProgress",
        "taskExecutionDetails": null,
        "taskId": "Take Snapshot"
      },
      {
        "duration": "0:00:00",
        "endTime": null,
        "instanceId": null,
        "progressPercentage": null,
          "startTime": null,
          "status": "NotStarted",
          "taskExecutionDetails": null,
          "taskId": "Transfer data to vault"
        }
      ]
    },
    "isUserTriggered": null,
    "jobType": "AzureIaaSVMJob",
    "operation": "Backup",
    "startTime": "2022-05-08T13:01:11.106429+00:00",
    "status": "InProgress",
    "virtualMachineVersion": "Compute"
  },
  "resourceGroup": "testRG",
  "tags": null,
  "type": "Microsoft.RecoveryServices/vaults/backupJobs"
}
```

List backups

```
az backup item list --resource-group $RGName --vault-name $RSVName
```

Output:

```
[
  {
    "eTag": null,
    "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/reso⌟
    ↪  urceGroups/testRG/providers/Microsoft.RecoveryServices/vaul⌟
    ↪  ts/myRSV/backupFabrics/Azure/protectionContainers/IaasVMCon⌟
    ↪  tainer;iaasvmcontainerv2;testrg;clickhousevm/protectedItems⌟
    ↪  /VM;iaasvmcontainerv2;testrg;clickhousevm",
    "location": null,
    "name": "VM;iaasvmcontainerv2;testrg;clickhousevm",
    "properties": {
      "backupManagementType": "AzureIaasVM",
      "backupSetName": null,
      "containerName": "iaasvmcontainerv2;testrg;clickhousevm",
      "createMode": null,
      "deferredDeleteTimeInUtc": null,
      "deferredDeleteTimeRemaining": null,
      "extendedInfo": null,
      "extendedProperties": null,
      "friendlyName": "clickhouseVM",
      "healthDetails": null,
      "healthStatus": "Passed",
      "isArchiveEnabled": false,
      "isDeferredDeleteScheduleUpcoming": null,
      "isRehydrate": null,
      "isScheduledForDeferredDelete": null,
      "kpisHealths": {},
      "lastBackupStatus": "",
      "lastBackupTime": "2001-01-01T00:00:00+00:00",
      "lastRecoveryPoint": "2022-05-08T13:01:15.411726+00:00",
      "policyId": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6⌋
       ↪  785/resourceGroups/testRG/providers/Microsoft.RecoverySer⌋
       ↪  vices/vaults/myRSV/backupPolicies/DefaultPolicy",
      "policyName": "DefaultPolicy",
      "protectedItemDataId": "123146169525886",
      "protectedItemType": "Microsoft.Compute/virtualMachines",
      "protectionState": "Protected",
      "protectionStatus": "Healthy",
      "resourceGuardOperationRequests": null,
      "sourceResourceId": "/subscriptions/4b48eb85-91f3-4902-b74b-e⌋
       ↪  84641fb6785/resourceGroups/testRG/providers/Microsoft.Com⌋
       ↪  pute/virtualMachines/clickhouseVM",
      "virtualMachineId": "/subscriptions/4b48eb85-91f3-4902-b74b-e⌋
       ↪  84641fb6785/resourceGroups/testRG/providers/Microsoft.Com⌋
       ↪  pute/virtualMachines/clickhouseVM",
      "workloadType": "VM"
    },
    "resourceGroup": "testRG",
    "tags": null,
    "type": "Microsoft.RecoveryServices/vaults/backupFabrics/protec⌋
     ↪  tionContainers/protectedItems"
  }
]
```

A.11.2 Download script

The following script was copied from the Microsoft documentation and saved as `delete-rsv.ps1` in the home directory in the Azure Cloud Shell [65]. The four variables at the start of the script were modified to work with our test environment. In addition, the first line (`Connect-AzAccount`) was commented out, because the script was run directly in the Cloud Shell, and thus did not need to be authenticated.

The script was run by executing `./delete-rsv.ps1`.

Script to delete RSV:

```
# Connect-AzAccount
```

```
$VaultName = $RSVName #enter vault name
$Subscription = "Azure for Students" #enter Subscription name
$ResourceGroup = $RGName #enter Resource group name
$SubscriptionId = $subscription #enter Subscription ID

Select-AzSubscription $Subscription
$VaultToDelete = Get-AzRecoveryServicesVault -Name $VaultName
 ↪  -ResourceGroupName $ResourceGroup
Set-AzRecoveryServicesAsrVaultContext -Vault $VaultToDelete

Set-AzRecoveryServicesVaultProperty -Vault $VaultToDelete.ID
 ↪  -SoftDeleteFeatureState Disable #disable soft delete
Write-Host "Soft delete disabled for the vault" $VaultName
$containerSoftDelete = Get-AzRecoveryServicesBackupItem
 ↪  -BackupManagementType AzureVM -WorkloadType AzureVM -VaultId
 ↪  $VaultToDelete.ID | Where-Object {$_.DeleteState -eq
 ↪  "ToBeDeleted"} #fetch backup items in soft delete state
foreach ($softitem in $containerSoftDelete)
{
    Undo-AzRecoveryServicesBackupItemDeletion -Item $softitem
     ↪  -VaultId $VaultToDelete.ID -Force #undelete items in soft
     ↪  delete state
}
#Invoking API to disable enhanced security
$azProfile = [Microsoft.Azure.Commands.Common.Authentication.Abstra⌋
 ↪  ctions.AzureRmProfileProvider]::Instance.Profile
$profileClient = New-Object -TypeName
 ↪  Microsoft.Azure.Commands.ResourceManager.Common.RMProfileClient
 ↪  -ArgumentList ($azProfile)
$accesstoken = Get-AzAccessToken
$token = $accesstoken.Token
$authHeader = @{
    'Content-Type'='application/json'
    'Authorization'='Bearer ' + $token
}
$body = @{properties=@{enhancedSecurityState= "Disabled"}}
$restUri = 'https://management.azure.com/subscriptions/'+$Subscript⌋
 ↪  ionId+'/resourcegroups/'+$ResourceGroup+'/providers/Microsoft.R⌋
 ↪  ecoveryServices/vaults/'+$VaultName+'/backupconfig/vaultconfig?⌋
 ↪  api-version=2019-05-13' #Replace "management.azure.com" with
 ↪  "management.usgovcloudapi.net" if your subscription is in USGov.
$response = Invoke-RestMethod -Uri $restUri -Headers $authHeader
 ↪  -Body ($body | ConvertTo-JSON -Depth 9) -Method PATCH
```

```
#Fetch all protected items and servers
$backupItemsVM = Get-AzRecoveryServicesBackupItem
↪  -BackupManagementType AzureVM -WorkloadType AzureVM -VaultId
↪  $VaultToDelete.ID
$backupItemsSQL = Get-AzRecoveryServicesBackupItem
↪  -BackupManagementType AzureWorkload -WorkloadType MSSQL
↪  -VaultId $VaultToDelete.ID
$backupItemsAFS = Get-AzRecoveryServicesBackupItem
↪  -BackupManagementType AzureStorage -WorkloadType AzureFiles
↪  -VaultId $VaultToDelete.ID
$backupItemsSAP = Get-AzRecoveryServicesBackupItem
↪  -BackupManagementType AzureWorkload -WorkloadType
↪  SAPHanaDatabase -VaultId $VaultToDelete.ID
$backupContainersSQL = Get-AzRecoveryServicesBackupContainer
↪  -ContainerType AzureVMAppContainer -Status Registered -VaultId
↪  $VaultToDelete.ID | Where-Object {$_.ExtendedInfo.WorkloadType
↪  -eq "SQL"}
$protectableItemsSQL = Get-AzRecoveryServicesBackupProtectableItem
↪  -WorkloadType MSSQL -VaultId $VaultToDelete.ID | Where-Object
↪  {$_.IsAutoProtected -eq $true}
$backupContainersSAP = Get-AzRecoveryServicesBackupContainer
↪  -ContainerType AzureVMAppContainer -Status Registered -VaultId
↪  $VaultToDelete.ID | Where-Object {$_.ExtendedInfo.WorkloadType
↪  -eq "SAPHana"}
$StorageAccounts = Get-AzRecoveryServicesBackupContainer
↪  -ContainerType AzureStorage -Status Registered -VaultId
↪  $VaultToDelete.ID
$backupServersMARS = Get-AzRecoveryServicesBackupContainer
↪  -ContainerType "Windows" -BackupManagementType MAB -VaultId
↪  $VaultToDelete.ID
$backupServersMABS = Get-AzRecoveryServicesBackupManagementServer
↪  -VaultId $VaultToDelete.ID| Where-Object {
↪  $_.BackupManagementType -eq "AzureBackupServer" }
$backupServersDPM = Get-AzRecoveryServicesBackupManagementServer
↪  -VaultId $VaultToDelete.ID | Where-Object {
↪  $_.BackupManagementType-eq "SCDPM" }
$pvtendpoints = Get-AzPrivateEndpointConnection
↪  -PrivateLinkResourceId $VaultToDelete.ID

foreach($item in $backupItemsVM)
    {
        Disable-AzRecoveryServicesBackupProtection -Item $item
         ↪  -VaultId $VaultToDelete.ID -RemoveRecoveryPoints -Force
         ↪  #stop backup and delete Azure VM backup items
    }
Write-Host "Disabled and deleted Azure VM backup items"

foreach($item in $backupItemsSQL)
    {
        Disable-AzRecoveryServicesBackupProtection -Item $item
         ↪  -VaultId $VaultToDelete.ID -RemoveRecoveryPoints -Force
         ↪  #stop backup and delete SQL Server in Azure VM backup
         ↪  items
    }
Write-Host "Disabled and deleted SQL Server backup items"

foreach($item in $protectableItems)
    {
        Disable-AzRecoveryServicesBackupAutoProtection
         ↪  -BackupManagementType AzureWorkload -WorkloadType MSSQL
         ↪  -InputItem $item -VaultId $VaultToDelete.ID #disable
         ↪  auto-protection for SQL
    }
Write-Host "Disabled auto-protection and deleted SQL protectable
 ↪  items"

foreach($item in $backupContainersSQL)
    {
        Unregister-AzRecoveryServicesBackupContainer -Container
         ↪  $item -Force -VaultId $VaultToDelete.ID #unregister SQL
         ↪  Server in Azure VM protected server
    }
Write-Host "Deleted SQL Servers in Azure VM containers"

foreach($item in $backupItemsSAP)
    {
        Disable-AzRecoveryServicesBackupProtection -Item $item
         ↪  -VaultId $VaultToDelete.ID -RemoveRecoveryPoints -Force
         ↪  #stop backup and delete SAP HANA in Azure VM backup
         ↪  items
    }
Write-Host "Disabled and deleted SAP HANA backup items"

foreach($item in $backupContainersSAP)
    {
        Unregister-AzRecoveryServicesBackupContainer -Container
         ↪  $item -Force -VaultId $VaultToDelete.ID #unregister SAP
         ↪  HANA in Azure VM protected server
    }
Write-Host "Deleted SAP HANA in Azure VM containers"

foreach($item in $backupItemsAFS)
    {
        Disable-AzRecoveryServicesBackupProtection -Item $item
         ↪  -VaultId $VaultToDelete.ID -RemoveRecoveryPoints -Force
         ↪  #stop backup and delete Azure File Shares backup items
    }
Write-Host "Disabled and deleted Azure File Share backups"

foreach($item in $StorageAccounts)
    {
        Unregister-AzRecoveryServicesBackupContainer -container
         ↪  $item -Force -VaultId $VaultToDelete.ID #unregister
         ↪  storage accounts
    }
Write-Host "Unregistered Storage Accounts"

foreach($item in $backupServersMARS)
    {
            Unregister-AzRecoveryServicesBackupContainer -Container
             ↪  $item -Force -VaultId $VaultToDelete.ID #unregister
             ↪  MARS servers and delete corresponding backup items
    }
Write-Host "Deleted MARS Servers"

foreach($item in $backupServersMABS)
    {
            Unregister-AzRecoveryServicesBackupManagementServer
             ↪  -AzureRmBackupManagementServer $item -VaultId
             ↪  $VaultToDelete.ID #unregister MABS servers and
             ↪  delete corresponding backup items
    }
Write-Host "Deleted MAB Servers"

foreach($item in $backupServersDPM)
    {
            Unregister-AzRecoveryServicesBackupManagementServer
             ↪  -AzureRmBackupManagementServer $item -VaultId
             ↪  $VaultToDelete.ID #unregister DPM servers and
             ↪  delete corresponding backup items
    }
Write-Host "Deleted DPM Servers"

#Deletion of ASR Items

$fabricObjects = Get-AzRecoveryServicesAsrFabric
if ($null -ne $fabricObjects) {
        # First DisableDR all VMs.
        foreach ($fabricObject in $fabricObjects) {
                $containerObjects =
                 ↪  Get-AzRecoveryServicesAsrProtectionContainer
                 ↪  -Fabric $fabricObject
                foreach ($containerObject in $containerObjects) {
                        $protectedItems = Get-AzRecoveryServicesAsr⌋
                         ↪  ReplicationProtectedItem
                         ↪  -ProtectionContainer $containerObject
                        # DisableDR all protected items
                        foreach ($protectedItem in $protectedItems)
                         ↪  {
                                Write-Host "Triggering
                                 ↪  DisableDR(Purge) for item:"
                                 ↪  $protectedItem.Name
                                Remove-AzRecoveryServicesAsrReplica⌋
                                 ↪  tionProtectedItem -InputObject
                                 ↪  $protectedItem -Force
                                Write-Host "DisableDR(Purge)
                                 ↪  completed"
                        }

                        $containerMappings = Get-AzRecoveryServices⌋
                         ↪  AsrProtectionContainerMapping
                         ↪  `
                                -ProtectionContainer
                                 ↪  $containerObject
                        # Remove all Container Mappings
                        foreach ($containerMapping in
                         ↪  $containerMappings) {
                                Write-Host "Triggering Remove
                                 ↪  Container Mapping: "
                                 ↪  $containerMapping.Name
                                Remove-AzRecoveryServicesAsrProtect⌋
                                 ↪  ionContainerMapping
                                 ↪  -ProtectionContainerMapping
                                 ↪  $containerMapping -Force
                                Write-Host "Removed Container
                                 ↪  Mapping."
                        }
                }
                $NetworkObjects = Get-AzRecoveryServicesAsrNetwork
                 ↪  -Fabric $fabricObject
                foreach ($networkObject in $NetworkObjects)
                {
                        #Get the PrimaryNetwork
                        $PrimaryNetwork =
                         ↪  Get-AzRecoveryServicesAsrNetwork
                         ↪  -Fabric $fabricObject -FriendlyName
                         ↪  $networkObject
                        $NetworkMappings =
                         ↪  Get-AzRecoveryServicesAsrNetworkMapping
                         ↪  -Network $PrimaryNetwork
                        foreach ($networkMappingObject in
                         ↪  $NetworkMappings)
                        {
                                #Get the Neetwork Mappings
                                $NetworkMapping = Get-AzRecoverySer⌋
                                 ↪  vicesAsrNetworkMapping -Name
                                 ↪  $networkMappingObject.Name
                                 ↪  -Network $PrimaryNetwork
                                Remove-AzRecoveryServicesAsrNetwork⌋
                                 ↪  Mapping -InputObject
                                 ↪  $NetworkMapping
                        }
                }
                # Remove Fabric
                Write-Host "Triggering Remove Fabric:"
                 ↪  $fabricObject.FriendlyName
                Remove-AzRecoveryServicesAsrFabric -InputObject
                 ↪  $fabricObject -Force
                Write-Host "Removed Fabric."
        }
}

foreach($item in $pvtendpoints)
        {
                $penamesplit = $item.Name.Split(".")
                $pename = $penamesplit[0]
                Remove-AzPrivateEndpointConnection -ResourceId
                 ↪  $item.PrivateEndpoint.Id -Force #remove private
                 ↪  endpoint connections
                Remove-AzPrivateEndpoint -Name $pename
                 ↪  -ResourceGroupName $ResourceGroup -Force
                 ↪  #remove private endpoints
        }
Write-Host "Removed Private Endpoints"

#Recheck ASR items in vault
$fabricCount = 0
$ASRProtectedItems = 0
$ASRPolicyMappings = 0
$fabricObjects = Get-AzRecoveryServicesAsrFabric
if ($null -ne $fabricObjects) {
        foreach ($fabricObject in $fabricObjects) {
                $containerObjects =
                 ↪  Get-AzRecoveryServicesAsrProtectionContainer
                 ↪  -Fabric $fabricObject
                foreach ($containerObject in $containerObjects) {
                        $protectedItems = Get-AzRecoveryServicesAsr⌋
                         ↪  ReplicationProtectedItem
                         ↪  -ProtectionContainer $containerObject
                        foreach ($protectedItem in $protectedItems)
                         ↪  {
                                $ASRProtectedItems++
                        }
                        $containerMappings = Get-AzRecoveryServices⌋
                         ↪  AsrProtectionContainerMapping
                         ↪  `
                                -ProtectionContainer
                                 ↪  $containerObject
                        foreach ($containerMapping in
                         ↪  $containerMappings) {
                                $ASRPolicyMappings++
                        }
                }
                $fabricCount++
        }
}
#Recheck presence of backup items in vault
```

```
$backupItemsVMFin = Get-AzRecoveryServicesBackupItem
↪  -BackupManagementType AzureVM -WorkloadType AzureVM -VaultId
↪  $VaultToDelete.ID
$backupItemsSQLFin = Get-AzRecoveryServicesBackupItem
↪  -BackupManagementType AzureWorkload -WorkloadType MSSQL
↪  -VaultId $VaultToDelete.ID
$backupContainersSQLFin = Get-AzRecoveryServicesBackupContainer
↪  -ContainerType AzureVMAppContainer -Status Registered -VaultId
↪  $VaultToDelete.ID | Where-Object {$_.ExtendedInfo.WorkloadType
↪  -eq "SQL"}
$protectableItemsSQLFin =
↪  Get-AzRecoveryServicesBackupProtectableItem -WorkloadType MSSQL
↪  -VaultId $VaultToDelete.ID | Where-Object {$_.IsAutoProtected
↪  -eq $true}
$backupItemsSAPFin = Get-AzRecoveryServicesBackupItem
↪  -BackupManagementType AzureWorkload -WorkloadType
↪  SAPHanaDatabase -VaultId $VaultToDelete.ID
$backupContainersSAPFin = Get-AzRecoveryServicesBackupContainer
↪  -ContainerType AzureVMAppContainer -Status Registered -VaultId
↪  $VaultToDelete.ID | Where-Object {$_.ExtendedInfo.WorkloadType
↪  -eq "SAPHana"}
$backupItemsAFSFin = Get-AzRecoveryServicesBackupItem
↪  -BackupManagementType AzureStorage -WorkloadType AzureFiles
↪  -VaultId $VaultToDelete.ID
$StorageAccountsFin = Get-AzRecoveryServicesBackupContainer
↪  -ContainerType AzureStorage -Status Registered -VaultId
↪  $VaultToDelete.ID
$backupServersMARSFin = Get-AzRecoveryServicesBackupContainer
↪  -ContainerType "Windows" -BackupManagementType MAB -VaultId
↪  $VaultToDelete.ID
$backupServersMABSFin =
↪  Get-AzRecoveryServicesBackupManagementServer -VaultId
↪  $VaultToDelete.ID| Where-Object { $_.BackupManagementType -eq
↪  "AzureBackupServer" }
$backupServersDPMFin = Get-AzRecoveryServicesBackupManagementServer
↪  -VaultId $VaultToDelete.ID | Where-Object {
↪  $_.BackupManagementType-eq "SCDPM" }
$pvtendpointsFin = Get-AzPrivateEndpointConnection
↪  -PrivateLinkResourceId $VaultToDelete.ID
```

```
Write-Host "Number of backup items left in the vault and which need
 ↪  to be deleted:" $backupItemsVMFin.count "Azure VMs"
 ↪  $backupItemsSQLFin.count "SQL Server Backup Items"
 ↪  $backupContainersSQLFin.count "SQL Server Backup Containers"
 ↪  $protectableItemsSQLFin.count "SQL Server Instances"
 ↪  $backupItemsSAPFin.count "SAP HANA backup items"
 ↪  $backupContainersSAPFin.count "SAP HANA Backup Containers"
 ↪  $backupItemsAFSFin.count "Azure File Shares"
 ↪  $StorageAccountsFin.count "Storage Accounts"
 ↪  $backupServersMARSFin.count "MARS Servers"
 ↪  $backupServersMABSFin.count "MAB Servers"
 ↪  $backupServersDPMFin.count "DPM Servers" $pvtendpointsFin.count
 ↪  "Private endpoints"
Write-Host "Number of ASR items left in the vault and which need to
 ↪  be deleted:" $ASRProtectedItems "ASR protected items"
 ↪  $ASRPolicyMappings "ASR policy mappings" $fabricCount "ASR
 ↪  Fabrics" $pvtendpointsFin.count "Private endpoints. Warning:
 ↪  This script will only remove the replication configuration from
 ↪  Azure Site Recovery and not from the source. Please cleanup the
 ↪  source manually. Visit
 ↪  https://go.microsoft.com/fwlink/?linkid=2182781 to learn more"
Remove-AzRecoveryServicesVault -Vault $VaultToDelete
#Finish
```

Running the script:

```
./delete-rsv.ps1
# Name                                    Account
 ↪                              SubscriptionName
 ↪                 Environment
 ↪  TenantId
# ----                                    -------
 ↪                              ----------------
 ↪                 -----------
 ↪  --------
# Azure for Students (4b48eb85-91f3-4902-... MSI@50342
 ↪                                Azure for Students
 ↪                   AzureCloud
 ↪    09a10672-822f-4467-a5ba-5bb375967c05
#
# ResourceName      : myRSV
# ResourceGroupName : testRG
# ResourceNamespace : Microsoft.RecoveryServices
# ResouceType       : vaults
#
```

```
#
# StorageModelType                  :
# StorageType                       :
# StorageTypeState                  :
# EnhancedSecurityState             : Enabled
# SoftDeleteFeatureState            : Disabled
# ResourceGuardOperationRequests    :
# IsSoftDeleteFeatureStateEditable : True
#
# Soft delete disabled for the vault myRSV
# Invoke-RestMethod: /home/torstein/delete-rsv.ps1:30
# Line |
#   30 |  $response = Invoke-RestMethod -Uri $restUri -Headers
 ↪  $authHeader -Bod ...
#      |
 ↪  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#      | {"error":{"code":"InvalidSubscriptionId","message":"The
 ↪  provided subscription identifier 'Azure for Students' is
 ↪  malformed or invalid."}}
#
#
# DynamicErrorMessage  :
# Properties           : {[VM Name, clickhouseVM], [Number of
 ↪  Recovery Points, 1]}
# SubTasks             : {}
# VmVersion            : Compute
# IsCancellable        : False
# IsRetriable          : False
# ErrorDetails         :
# ActivityId           : 7640f777-a528-43a7-b618-fea2ad3b0a52
# JobId                : 88685b46-5b5f-477a-a755-9632751dac21
# Operation            : DeleteBackupData
# Status               : Completed
# WorkloadName         : clickhouseVM
# StartTime            : 5/8/2022 1:22:27 PM
# EndTime              : 5/8/2022 1:24:19 PM
# Duration             : 00:01:51.8479521
# BackupManagementType : AzureVM
#
# Disabled and deleted Azure VM backup items
# Disabled and deleted SQL Server backup items
# Disabled auto-protection and deleted SQL protectable items
# Deleted SQL Servers in Azure VM containers
# Disabled and deleted SAP HANA backup items
```

```
# Deleted SAP HANA in Azure VM containers
# Disabled and deleted Azure File Share backups
# Unregistered Storage Accounts
# Deleted MARS Servers
# Deleted MAB Servers
# Deleted DPM Servers
# Removed Private Endpoints
# Number of backup items left in the vault and which need to be
 ↪  deleted: 0 Azure VMs 0 SQL Server Backup Items 0 SQL Server
 ↪  Backup Containers 0 SQL Server Instances 0 SAP HANA backup
 ↪  items 0 SAP HANA Backup Containers 0 Azure File Shares 0
 ↪  Storage Accounts 0 MARS Servers 0 MAB Servers 0 DPM Servers 0
 ↪  Private endpoints
# Number of ASR items left in the vault and which need to be
 ↪  deleted: 0 ASR protected items 0 ASR policy mappings 0 ASR
 ↪  Fabrics 0 Private endpoints. Warning: This script will only
 ↪  remove the replication configuration from Azure Site Recovery
 ↪  and not from the source. Please cleanup the source manually.
 ↪  Visit https://go.microsoft.com/fwlink/?linkid=2182781 to learn
 ↪  more
#
# Response : Vault has been deleted
```

It seems one of the lines in the script failed. This does not appear to have affected the deletion of the vault, though.

A “Soft Delete disabled for Vault” alert was fired by Azure Monitor.

A.11.3 Verify deletion

Trying to list backup items:

```
az backup item list --resource-group $RGName --vault-name $RSVName
# (ResourceNotFound) The Resource
 ↪  'Microsoft.RecoveryServices/vaults/myRSV' under resource group
 ↪  'testRG' was not found. For more details please go to
 ↪  https://aka.ms/ARMResourceNotFoundFix
# Code: ResourceNotFound
# Message: The Resource 'Microsoft.RecoveryServices/vaults/myRSV'
 ↪  under resource group 'testRG' was not found. For more details
 ↪  please go to https://aka.ms/ARMResourceNotFoundFix
```

The vault is completely gone!

A.12 Applying MUA to vault and attempting deletion

A.12.1 Creating a resource guard

In order to enable Multi User Authentication, a resource guard must be created in a different subscription or directory than the backup vault, but they must be in the same region.

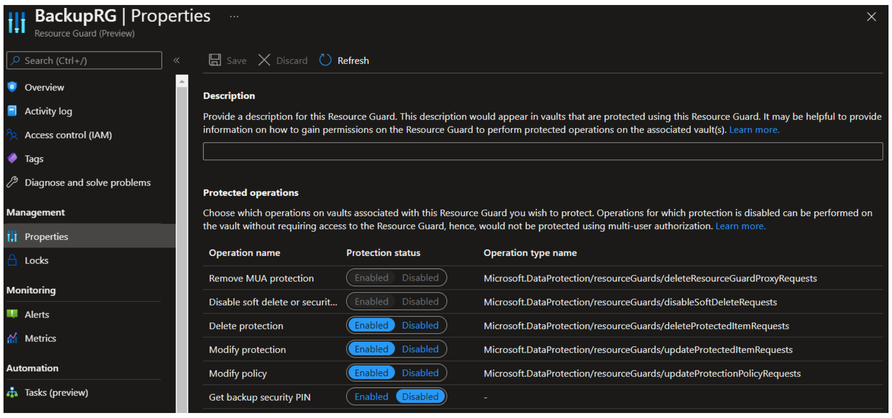
In a different Azure subscription than the one containing the Revocery Services vault, the security admin creates a resource guard. This can be done with the following command in powershell:

```
New-AzResource -Location "North Europe" -ResourceName "BackupRG"
 ↪  -ResourceType "Microsoft.DataProtection/resourceGuards"
 ↪  -ResourceGroupName "bProject"
```

```
which gives the following output:

Name              : BackupRG
ResourceId        : /subscriptions/c29c75c2-44ed-4c7e-a49f-12ebe976⌋
 ↪  37dd/resourceGroups/bProject/providers/Microsoft.DataProtection⌋
 ↪  /resourceGuards/BackupRG
ResourceName      : BackupRG
ResourceType      : Microsoft.DataProtection/resourceGuards
ResourceGroupName : bProject
Location          : northeurope
SubscriptionId    : c29c75c2-44ed-4c7e-a49f-12ebe97637dd
Properties        : @{provisioningState=Succeeded;
 ↪  resourceGuardOperations=System.Object[];
 ↪  vaultCriticalOperationExclusionList=System.Object[];
                    allowAutoApprovals=True}
```

By default the resource guard has all protected operations enabled by default, but the security admin can choose to disable some protections. Disable soft delete and Remove MUA protection cannot be disabled. See the following image for how this looks in the Azure GUI.



Description

Provide a description for this Resource Guard. This description would appear in vaults that are protected using this Resource Guard. It may be helpful to provide information on how to gain permissions on the Resource Guard to perform protected operations on the associated vault(s). Learn more.

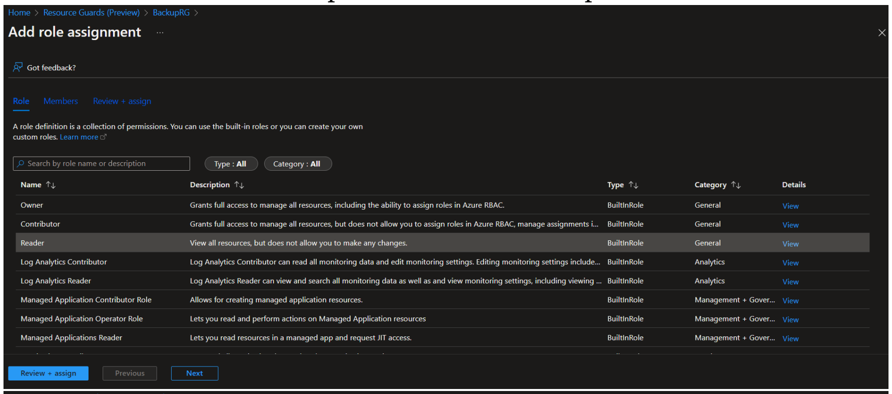
Protected operations

Choose which operations on vaults associated with this Resource Guard you wish to protect. Operations for which protection is disabled can be performed on the vault without requiring access to the Resource Guard, hence, would not be protected using multi-user authorization. Learn more.

Operation name	Protection status	Operation type name
Remove MUA protection	Enabled Disabled	Microsoft.DataProtection/resourceGuards/deleteResourceGuardProxyRequests
Disable soft delete or securit...	Enabled Disabled	Microsoft.DataProtection/resourceGuards/disableSoftDeleteRequests
Delete protection	[✓ Enabled] Disabled	Microsoft.DataProtection/resourceGuards/deleteProtectedItemRequests
Modify protection	[✓ Enabled] Disabled	Microsoft.DataProtection/resourceGuards/updateProtectedItemRequests
Modify policy	[✓ Enabled] Disabled	Microsoft.DataProtection/resourceGuards/updateProtectionPolicyRequests
Get backup security PIN	Enabled [✓ Disabled]	-

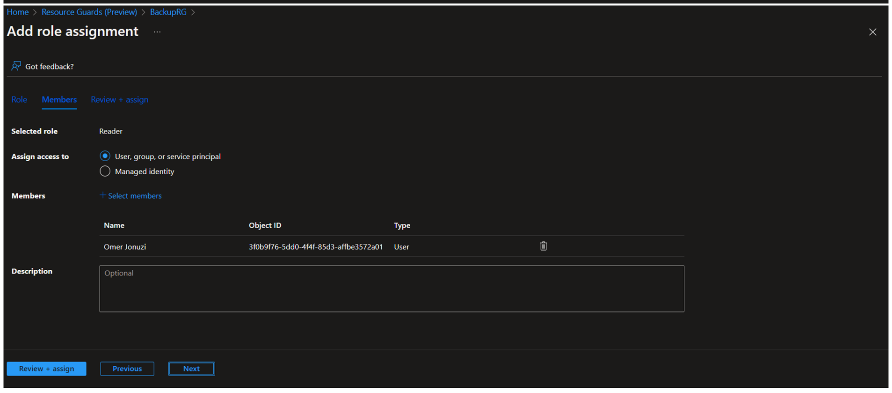
A.12.2 Assigning reader role to backup admin

The next step is to assign the reader role on the Resource Guard to the backup administrator account. In the GUI we can do this through the Access control(IAM)-blads. In the following images we add the reader role on the Resource guard to the user that acts as the backup admin in this example.



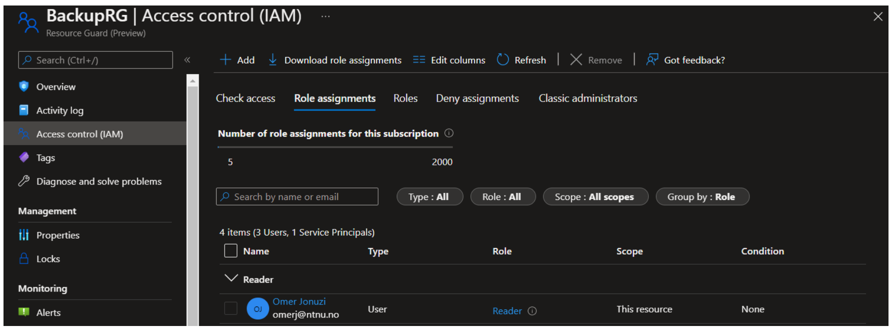
A role definition is a collection of permissions. You can use the built-in roles or you can create your own custom roles. Learn more

Name	Description	Type	Category	Details
Owner	Grants full access to manage all resources, including the ability to assign roles in Azure RBAC.	BuiltInRole	General	View
Contributor	Grants full access to manage all resources, but does not allow you to assign roles in Azure RBAC, manage assignments i...	BuiltInRole	General	View
Reader	View all resources, but does not allow you to make any changes.	BuiltInRole	General	View
Log Analytics Contributor	Log Analytics Contributor can read all monitoring data and edit monitoring settings. Editing monitoring settings include...	BuiltInRole	Analytics	View
Log Analytics Reader	Log Analytics Reader can view and search all monitoring data as well as and view monitoring settings, including viewing ...	BuiltInRole	Analytics	View
Managed Application Contributor Role	Allows for creating managed application resources.	BuiltInRole	Management + Gover...	View
Managed Application Operator Role	Lets you read and perform actions on Managed Application resources	BuiltInRole	Management + Gover...	View
Managed Applications Reader	Lets you read resources in a managed app and request JIT access.	BuiltInRole	Management + Gover...	View



Selected role	Reader
Assign access to	[✓ User, group, or service principal] Managed identity
Members	
Description	Optional

Name	Object ID	Type
Omer Jonuzi	3f0b9f76-5dd0-4f4f-85d3-affbe3572a01	User



BackupRG | Access control (IAM)
Resource Guard (Preview)

Number of role assignments for this subscription: 5 / 2000

Type : All | Role : All | Scope : All scopes | Group by : Role

4 items (3 Users, 1 Service Principals)

Name	Type	Role	Scope	Condition
Reader				
Omer Jonuzi omerj@ntnu.no	User	Reader	This resource	None

After the reader role has been granted, the backup administrator can choose to protect any Recovery Service vault with multi-user authorization. To do this they must choose which resource guard to protect it with, which is why the reader role is required. If they do, any protected operation, such as disabling soft-delete or turning MUA off again, should fail as they do not have the necessary permissions on the resource guard.

A.12.3 Applying MUA to vault

We created a Recovery Services vault, added a resource guard and attempted to disable soft delete.

Create a Recovery Services vault Since the RSV was deleted in the last experiment, we made a new one.

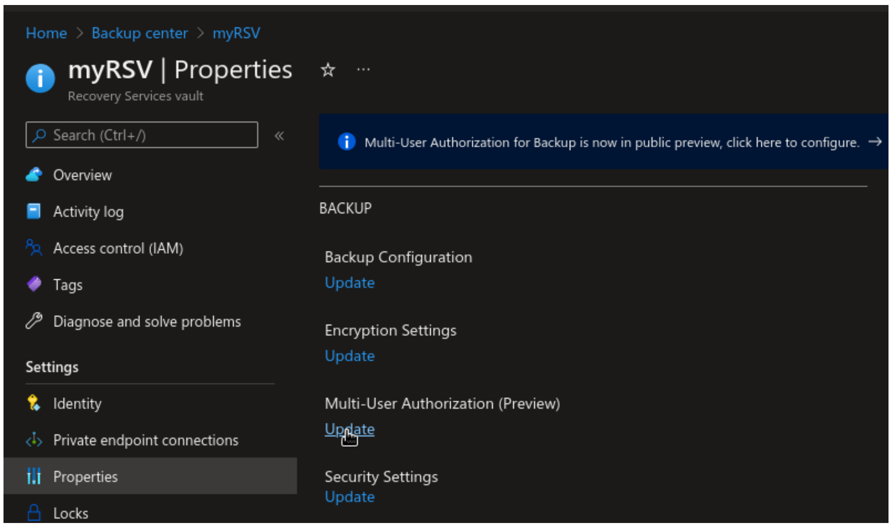
Create RSV:

```
az backup vault create --location $location --name $RSVName
↪  --resource-group $RGName
```

Output:

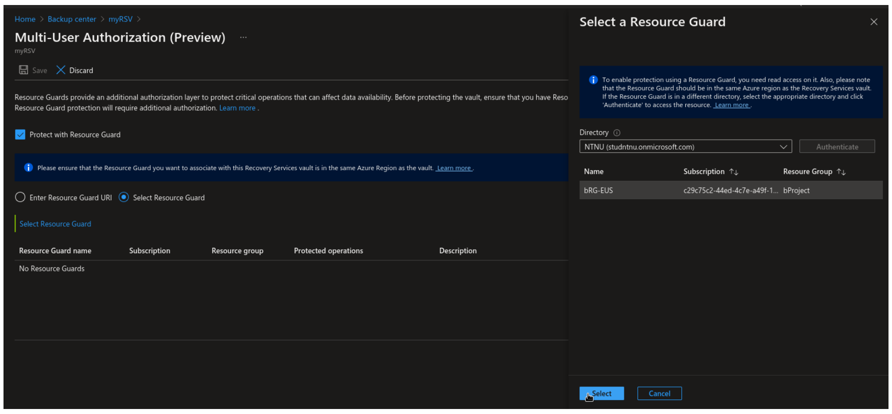
```
{
  "etag": "W/\"datetime'2022-05-13T13%3A13%3A28.4632743Z'\"",
  "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resour⌋
  ↪  ceGroups/testRG/providers/Microsoft.RecoveryServices/vaults/m⌋
  ↪  yRSV",
  "identity": null,
  "location": "eastus",
  "name": "myRSV",
  "properties": {
    "encryption": null,
    "privateEndpointConnections": null,
    "privateEndpointStateForBackup": "None",
    "privateEndpointStateForSiteRecovery": "None",
    "provisioningState": "Succeeded",
    "upgradeDetails": null
  },
  "resourceGroup": "testRG",
  "sku": {
    "name": "Standard",
    "tier": null
  },
  "systemData": null,
  "tags": null,
  "type": "Microsoft.RecoveryServices/vaults"
}
```

Add resource guard Press update MUA settings:



Multi-User Authorization for Backup is now in public preview, click here to configure.

Select resource guard:



Resource Guards provide an additional authorization layer to protect critical operations that can affect data availability. Before protecting the vault, ensure that you have Reso
Resource Guard protection will require additional authorization. Learn more .

Please ensure that the Resource Guard you want to associate with this Recovery Services vault is in the same Azure Region as the vault. Learn more .

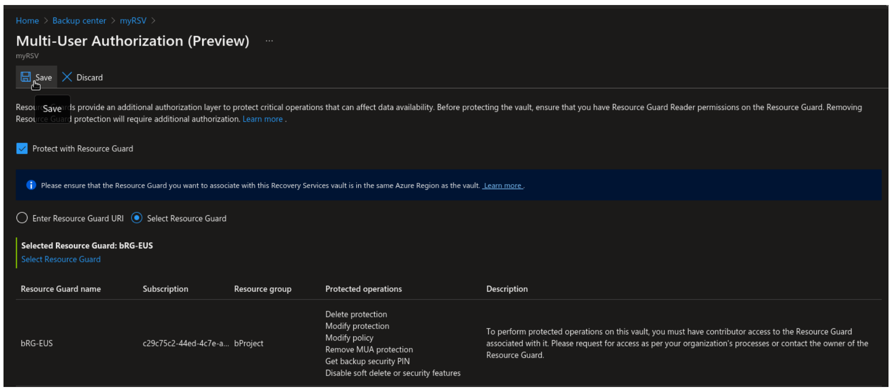
Resource Guard name	Subscription	Resource group	Protected operations	Description
No Resource Guards				

To enable protection using a Resource Guard, you need read access on it. Also, please note that the Resource Guard should be in the same Azure region as the Recovery Services vault. If the Resource Guard is in a different directory, select the appropriate directory and click 'Authenticate' to access the resource. Learn more .

Directory: NTNU (studntnu.onmicrosoft.com)

Name	Subscription ↑↓	Resoure Group ↑↓
bRG-EUS	c29c75c2-44ed-4c7e-a49f-1...	bProject

Save settings:



Multi-User Authorization (Preview)

myRSV

Resou ds provide an additional authorization layer to protect critical operations that can affect data availability. Before protecting the vault, ensure that you have Resource Guard Reader permissions on the Resource Guard. Removing Resou d protection will require additional authorization. Learn more .

☑ Protect with Resource Guard

Please ensure that the Resource Guard you want to associate with this Recovery Services vault is in the same Azure Region as the vault. Learn more .

○ Enter Resource Guard URI ● Select Resource Guard

Selected Resource Guard: bRG-EUS
Select Resource Guard

Resource Guard name	Subscription	Resource group	Protected operations	Description
bRG-EUS	c29c75c2-44ed-4c7e-a...	bProject	Delete protection Modify protection Modify policy Remove MUA protection Get backup security PIN Disable soft delete or security features	To perform protected operations on this vault, you must have contributor access to the Resource Guard associated with it. Please request for access as per your organization's processes or contact the owner of the Resource Guard.

A.12.4 Testing protected action

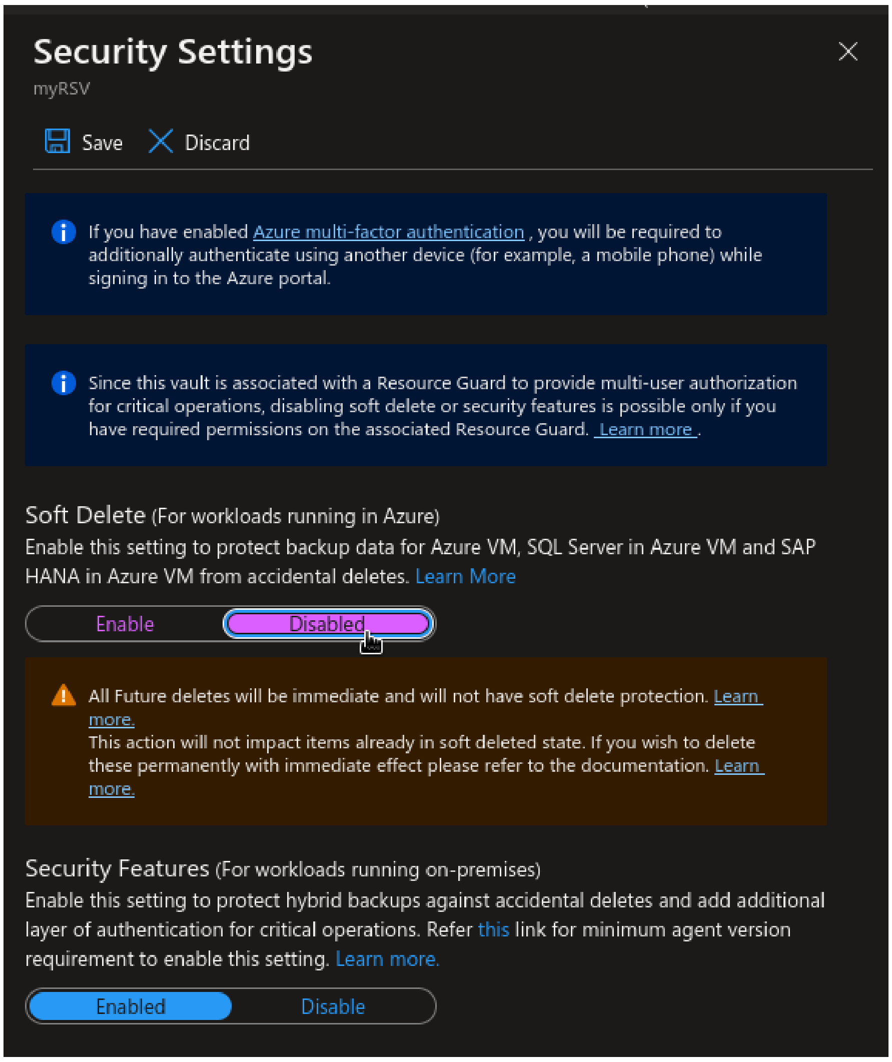
Try to disable soft delete via Azure CLI An attempt was made at disabling soft delete via the Azure CLI. The request failed with a generic “BadRequest” status code, presumably because of MUA.

```
# Get RSV
$RSV = Get-AzRecoveryServicesVault -Name $RSVName
 ↪  -ResourceGroupName $RGName

# Disable soft delete
Set-AzRecoveryServicesVaultProperty -VaultId $RSV.ID
 ↪  -SoftDeleteFeatureState Disable
# Set-AzRecoveryServicesVaultProperty: One or more errors occurred.
 ↪  (Operation returned an invalid status code 'BadRequest')
```

Try to disable soft delete via Azure Portal We attempted to disable soft delete via the Azure Portal.

Disabling soft delete for the vault:



Security Settings
myRSV

If you have enabled Azure multi-factor authentication , you will be required to additionally authenticate using another device (for example, a mobile phone) while signing in to the Azure portal.

Since this vault is associated with a Resource Guard to provide multi-user authorization for critical operations, disabling soft delete or security features is possible only if you have required permissions on the associated Resource Guard. Learn more .

Soft Delete (For workloads running in Azure)
Enable this setting to protect backup data for Azure VM, SQL Server in Azure VM and SAP HANA in Azure VM from accidental deletes. Learn More

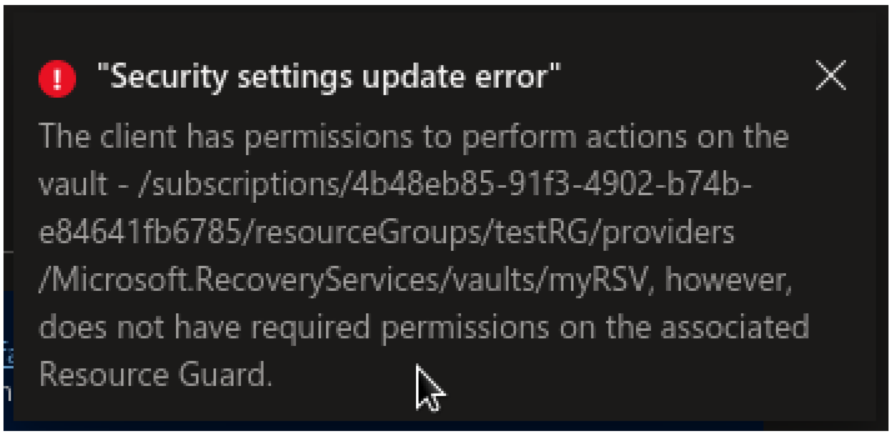
Enable [✓ Disabled]

All Future deletes will be immediate and will not have soft delete protection. Learn more.
This action will not impact items already in soft deleted state. If you wish to delete these permanently with immediate effect please refer to the documentation. Learn more.

Security Features (For workloads running on-premises)
Enable this setting to protect hybrid backups against accidental deletes and add additional layer of authentication for critical operations. Refer this link for minimum agent version requirement to enable this setting. Learn more.

[✓ Enabled] Disable

An error appeared in the notifications:



"Security settings update error"

The client has permissions to perform actions on the vault - /subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/resourceGroups/testRG/providers/Microsoft.RecoveryServices/vaults/myRSV, however, does not have required permissions on the associated Resource Guard.

Try to run RSV deletion script The script from [cref] was run once again. It appears it was able to delete the vault. We believe this is because the vault is empty. We attempted a second time with a vault containing backup data.

Output from script:

```
Name                                    Account
 ↪          SubscriptionName                Environment
 ↪             TenantId
----                                    -------
 ↪          ----------------                -----------
 ↪             --------
Azure for Students (4b48eb85-91f3-4902-... MSI@50342
 ↪            Azure for Students               AzureCloud
 ↪              09a10672-822f-4467-a5ba-5bb3759...

ResourceName      : myRSV
ResourceGroupName : testRG
ResourceNamespace : Microsoft.RecoveryServices
ResouceType       : vaults

Set-AzRecoveryServicesVaultProperty:
 ↪  /home/torstein/delete-rsv.ps1:12
Line |
  12 |  Set-AzRecoveryServicesVaultProperty -Vault
   ↪  $VaultToDelete.ID -SoftDel ...
     |  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~↵
      ↪  ~~~~~~~~~~
     | One or more errors occurred. (Operation returned an invalid
      ↪  status code 'BadRequest')

Soft delete disabled for the vault myRSV
Invoke-RestMethod: /home/torstein/delete-rsv.ps1:30
Line |
  30 |  $response = Invoke-RestMethod -Uri $restUri -Headers
   ↪  $authHeader -Bod ...
     |
      ↪  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     | {"error":{"code":"InvalidSubscriptionId","message":"The
      ↪  provided subscription identifier 'Azure for Students' is
      ↪  malformed or invalid."}}

Disabled and deleted Azure VM backup items
Disabled and deleted SQL Server backup items
Disabled auto-protection and deleted SQL protectable items
Deleted SQL Servers in Azure VM containers
```

```
Disabled and deleted SAP HANA backup items
Deleted SAP HANA in Azure VM containers
Disabled and deleted Azure File Share backups
Unregistered Storage Accounts
Deleted MARS Servers
Deleted MAB Servers
Deleted DPM Servers
Removed Private Endpoints
Number of backup items left in the vault and which need to be
 ↪  deleted: 0 Azure VMs 0 SQL Server Backup Items 0 SQL Server
 ↪  Backup Containers 0 SQL Server Instances 0 SAP HANA backup
 ↪  items 0 SAP HANA Backup Containers 0 Azure File Shares 0
 ↪  Storage Accounts 0 MARS Servers 0 MAB Servers 0 DPM Servers 0
 ↪  Private endpoints
Number of ASR items left in the vault and which need to be deleted:
 ↪  0 ASR protected items 0 ASR policy mappings 0 ASR Fabrics 0
 ↪  Private endpoints. Warning: This script will only remove the
 ↪  replication configuration from Azure Site Recovery and not from
 ↪  the source. Please cleanup the source manually. Visit
 ↪  https://go.microsoft.com/fwlink/?linkid=2182781 to learn more

Response : Vault has been deleted
```

List RSVs:

```
az backup vault list
```

```
[
  {
    "etag": "W/\"datetime'2022-05-10T09%3A58%3A14.2637541Z'\"",
    "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/reso⌋
     ↪  urceGroups/perfRG/providers/Microsoft.RecoveryServices/vaul⌋
     ↪  ts/perfRSV",
    "identity": null,
    "location": "eastus",
    "name": "perfRSV",
    "properties": {
      "encryption": null,
      "privateEndpointConnections": null,
      "privateEndpointStateForBackup": "None",
      "privateEndpointStateForSiteRecovery": "None",
      "provisioningState": "Succeeded",
      "upgradeDetails": null
    },
    "resourceGroup": "perfRG",
    "sku": {
      "name": "Standard",
      "tier": null
    },
    "systemData": null,
    "tags": null,
    "type": "Microsoft.RecoveryServices/vaults"
  }
]
```

"myRSV" is not present in the list.

create new RSV and back up an item Create RSV:

```
az backup vault create --location $location --name $RSVName
 ↪  --resource-group $RGName
```

Back up VM:

```
az backup protection enable-for-vm `
 --resource-group $RGName `
 --vault-name $RSVName `
 --vm $CHName `
 --policy-name $PolicyName
```

Add resource guard The same procedure was followed as in Add resource guard.

Try to run RSV deletion script The script was run once again. This time, we got a few more errors. The script still claims that soft delete was disabled, and that VMs were deleted, but this appears not to be the case.

Output from script:

```
Name                                     Account
 ↪          SubscriptionName                 Environment
 ↪             TenantId
----                                     -------
 ↪          ----------------                 -----------
 ↪             --------
Azure for Students (4b48eb85-91f3-4902-... MSI@50342
 ↪            Azure for Students              AzureCloud
 ↪               09a10672-822f-4467-a5ba-5bb3759...

ResourceName      : myRSV
ResourceGroupName : testRG
ResourceNamespace : Microsoft.RecoveryServices
```

```
ResouceType       : vaults

Set-AzRecoveryServicesVaultProperty:
 ↪  /home/torstein/delete-rsv.ps1:12
Line |
  12 |  Set-AzRecoveryServicesVaultProperty -Vault
   ↪  $VaultToDelete.ID -SoftDel ...
     |  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~↲
      ↪  ~~~~~~~~~~
     | One or more errors occurred. (Operation returned an invalid
      ↪  status code 'BadRequest')

Soft delete disabled for the vault myRSV
Invoke-RestMethod: /home/torstein/delete-rsv.ps1:30
Line |
  30 |  $response = Invoke-RestMethod -Uri $restUri -Headers
   ↪  $authHeader -Bod ...
     |
      ↪  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     | {"error":{"code":"InvalidSubscriptionId","message":"The
      ↪  provided subscription identifier 'Azure for Students' is
      ↪  malformed or invalid."}}

Disable-AzRecoveryServicesBackupProtection:
 ↪  /home/torstein/delete-rsv.ps1:49
Line |
  49 |          Disable-AzRecoveryServicesBackupProtection -Item
   ↪  $item -Vault ...
     |          ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~↲
      ↪  ~~~~~~~~~~
     | Unlock privilege access is needed to delete the
      ↪  ResourceGuard proxy

Disabled and deleted Azure VM backup items
Disabled and deleted SQL Server backup items
Disabled auto-protection and deleted SQL protectable items
Deleted SQL Servers in Azure VM containers
Disabled and deleted SAP HANA backup items
Deleted SAP HANA in Azure VM containers
Disabled and deleted Azure File Share backups
Unregistered Storage Accounts
Deleted MARS Servers
Deleted MAB Servers
Deleted DPM Servers
```

```
Removed Private Endpoints
Number of backup items left in the vault and which need to be
 ↪  deleted: 1 Azure VMs 0 SQL Server Backup Items 0 SQL Server
 ↪  Backup Containers 0 SQL Server Instances 0 SAP HANA backup
 ↪  items 0 SAP HANA Backup Containers 0 Azure File Shares 0
 ↪  Storage Accounts 0 MARS Servers 0 MAB Servers 0 DPM Servers 0
 ↪  Private endpoints
Number of ASR items left in the vault and which need to be deleted:
 ↪  0 ASR protected items 0 ASR policy mappings 0 ASR Fabrics 0
 ↪  Private endpoints. Warning: This script will only remove the
 ↪  replication configuration from Azure Site Recovery and not from
 ↪  the source. Please cleanup the source manually. Visit
 ↪  https://go.microsoft.com/fwlink/?linkid=2182781 to learn more
Remove-AzRecoveryServicesVault: /home/torstein/delete-rsv.ps1:204
Line |
 204 |  Remove-AzRecoveryServicesVault -Vault $VaultToDelete
     |  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     | Operation failed. ClientRequestId:
      ↪  9e041d38-0531-4dbb-8db8-63ec5f59f341-2022-05-13
      ↪  13:41:03Z-P One or more errors occurred. (Recovery
      ↪  Services Vault cannot
     | be deleted as there are existing resources within the vault.
      ↪   : clickhouseVM. Please ensure all containers have been
      ↪  unregistered from the vault and all
     | private endpoints associated with the vault have been
      ↪  deleted, and retry operation. For more details, see
      ↪  https://aka.ms/AB-AA4ecq5)
```

List RSVs:

```
az backup vault list
```

Output:

```
[
  {
    "etag": "W/\"datetime'2022-05-10T09%3A58%3A14.2637541Z'\"",
    "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/reso
     ↪  urceGroups/perfRG/providers/Microsoft.RecoveryServices/vaul
     ↪  ts/perfRSV",
    "identity": null,
    "location": "eastus",
    "name": "perfRSV",
    "properties": {
      "encryption": null,
      "privateEndpointConnections": null,
```

```
      "privateEndpointStateForBackup": "None",
      "privateEndpointStateForSiteRecovery": "None",
      "provisioningState": "Succeeded",
      "upgradeDetails": null
    },
    "resourceGroup": "perfRG",
    "sku": {
      "name": "Standard",
      "tier": null
    },
    "systemData": null,
    "tags": null,
    "type": "Microsoft.RecoveryServices/vaults"
  },
  {
    "etag": "W/\"datetime'2022-05-13T13%3A39%3A48.7614362Z'\"",
    "id": "/subscriptions/4b48eb85-91f3-4902-b74b-e84641fb6785/reso⌋
     ↪  urceGroups/testRG/providers/Microsoft.RecoveryServices/vaul⌋
     ↪  ts/myRSV",
    "identity": null,
    "location": "eastus",
    "name": "myRSV",
    "properties": {
      "encryption": null,
      "privateEndpointConnections": null,
      "privateEndpointStateForBackup": "None",
      "privateEndpointStateForSiteRecovery": "None",
      "provisioningState": "Succeeded",
      "upgradeDetails": null
    },
    "resourceGroup": "testRG",
    "sku": {
      "name": "Standard",
      "tier": null
    },
    "systemData": null,
    "tags": null,
    "type": "Microsoft.RecoveryServices/vaults"
  }
]
```

The vault (“myRSV”) is still present. In other words the backup administrator was not permitted to delete the vault as long as that involved performing protected actions.

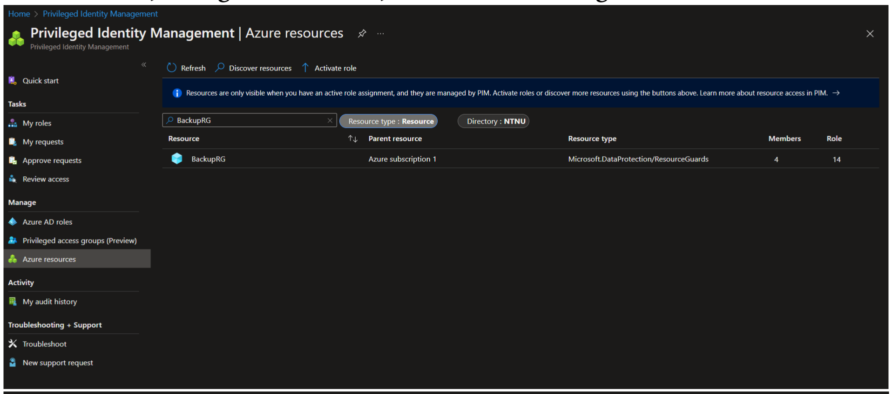
A.12.5 Permitting protected actions

In order to allow for protected actions to be performed on select occasions the reader role will not be sufficient. In order for a backup admin to perform protected actions within a resource guard's scope, they need the contributor-role on that resource guard. In order to to harness the security benefits of just-in-time access and multi user authorization, this can be configured with Azure Active Directory (Azure AD) Privileged Identity Management (PIM).

The end goal is for the backup admin to be able to raise a request for the contributor role on the resource guard, and temporarily be permitted to perform protected actions.

In PIM, the security admin must create an eligible assignment for the backup admin as contributor. In short, allowing for the backup admin to request the necessary access.

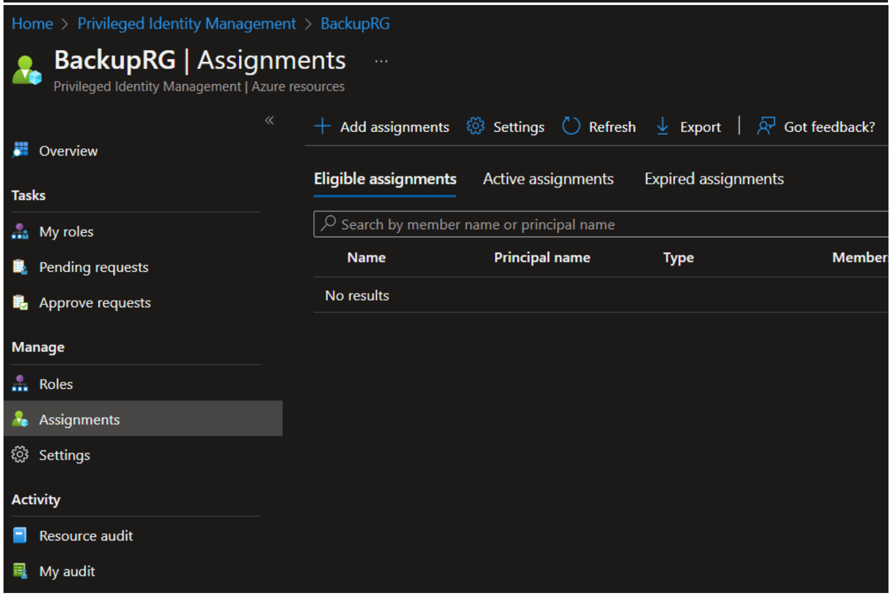
In Azure AD privileged identity management we navigate to the resource guard as shown, navigate within it, and click on assignments:



Resources are only visible when you have an active role assignment, and they are managed by PIM. Activate roles or discover more resources using the buttons above. Learn more about resource access in PIM. →

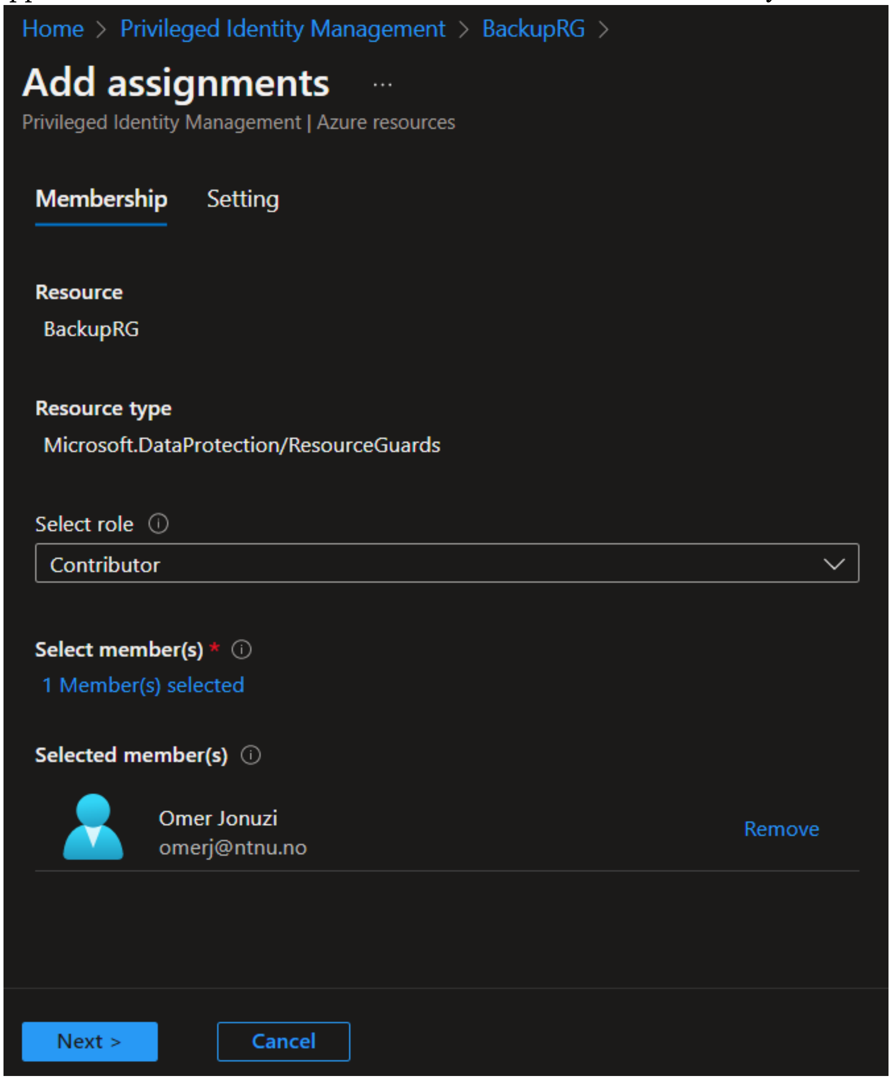
BackupRG | Resource type : Resource | Directory : NTNU

Resource	Parent resource	Resource type	Members	Role
BackupRG	Azure subscription 1	Microsoft.DataProtection/ResourceGuards	4	14



Name	Principal name	Type	Member
No results			

From here we add a new assignment, select the contributor role and our backup admin. THe security admin can choose to grant the assignent for a limitied amount of time, and from here set some settings for maximum duration, and the requirement of a justification before granting the assigned role. By selecting the assignemnt as "elligable", instead of "active", the backup admin must request to have the assigned role activated for them each time they need it. This must then be approved for the amount of time needed until it is automatically revoked.



Add assignments

Privileged Identity Management | Azure resources

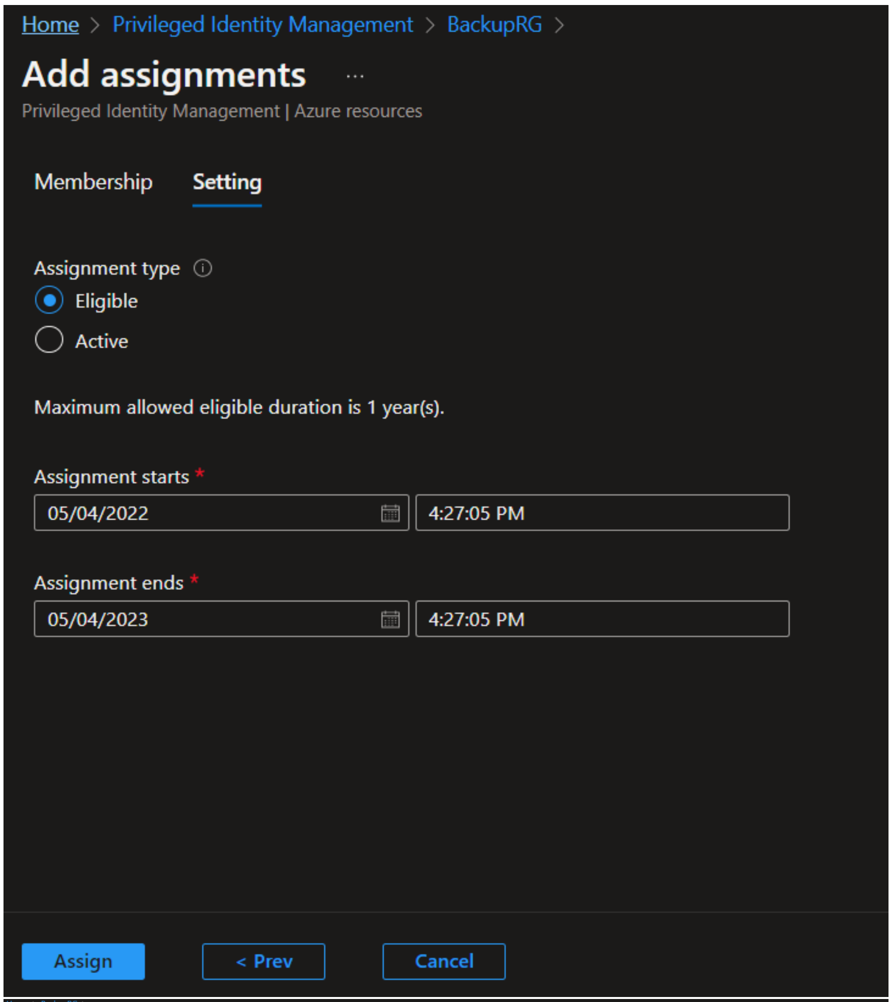
Resource
BackupRG

Resource type
Microsoft.DataProtection/ResourceGuards

Select role
Contributor

Select member(s) *
1 Member(s) selected

Selected member(s)
Omer Jonuzi
omerj@ntnu.no



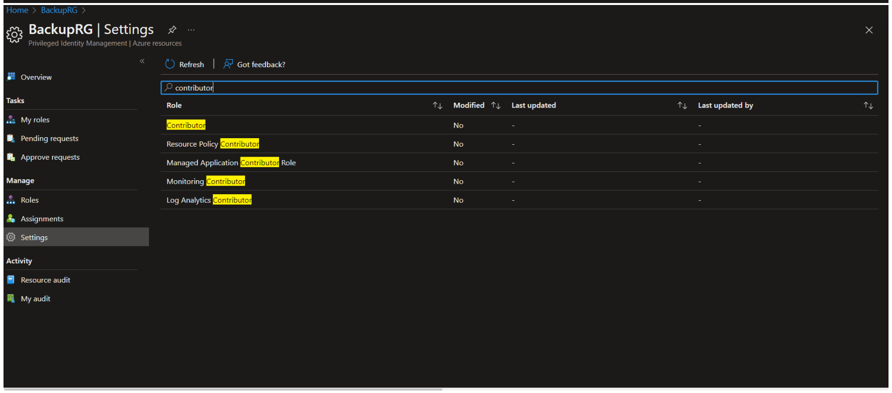
Home > Privileged Identity Management > BackupRG >
Add assignments ...
Privileged Identity Management | Azure resources
Membership
Setting

Assignment type
Eligible [✓]
Active

Maximum allowed eligible duration is 1 year(s).

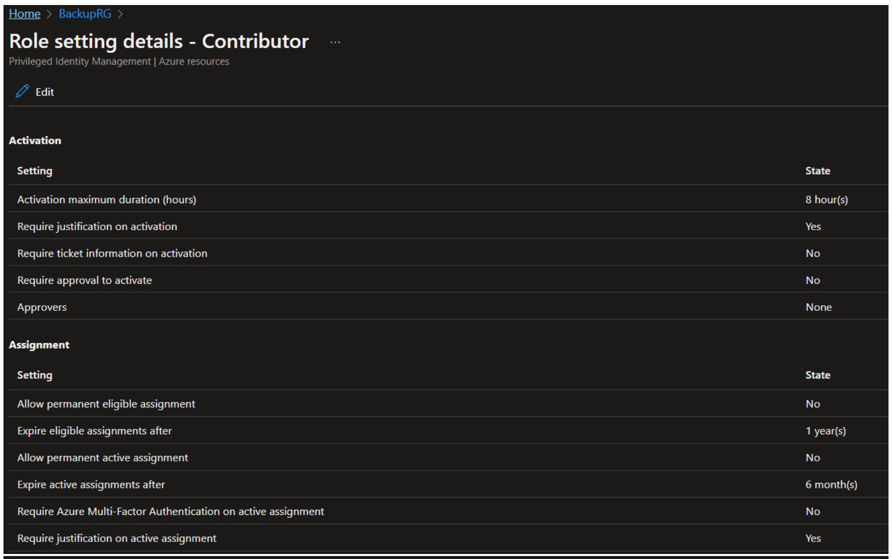
Assignment starts *
05/04/2022 4:27:05 PM

Assignment ends *
05/04/2023 4:27:05 PM



contributor

Role	Modified	Last updated	Last updated by
Contributor	No	-	-
Resource Policy Contributor	No	-	-
Managed Application Contributor Role	No	-	-
Monitoring Contributor	No	-	-
Log Analytics Contributor	No	-	-



Home > BackupRG >

Role setting details - Contributor ...

Privileged Identity Management | Azure resources

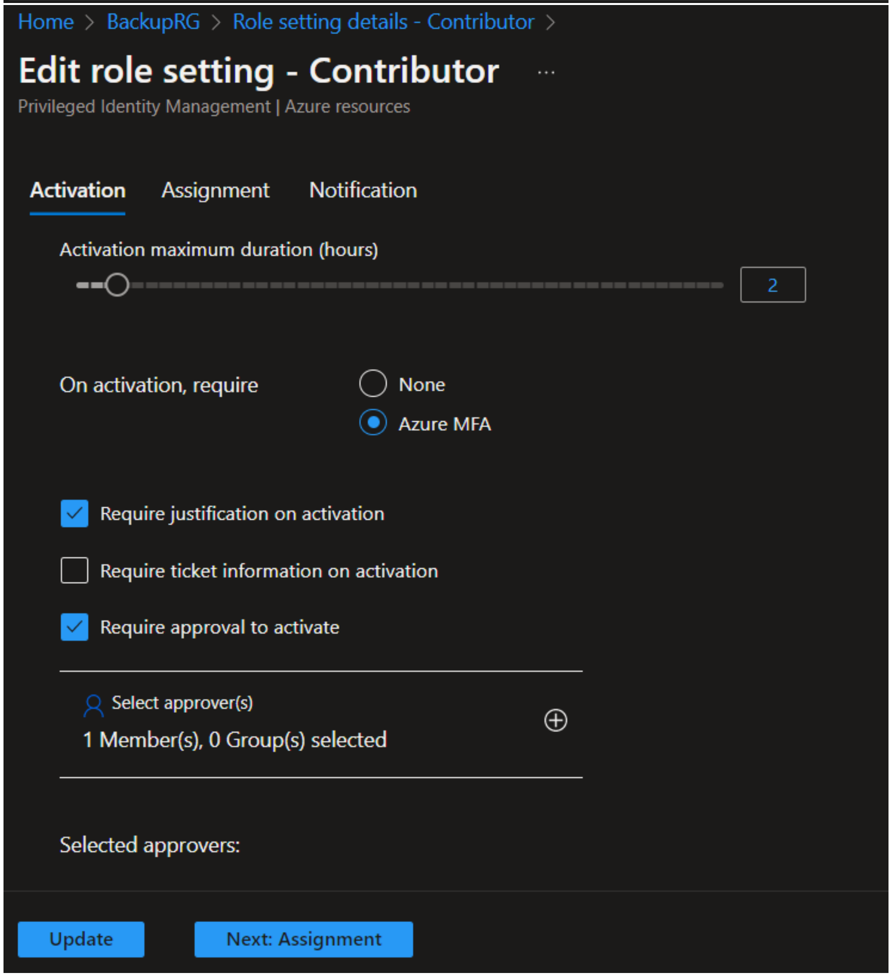
Edit

Activation

Setting	State
Activation maximum duration (hours)	8 hour(s)
Require justification on activation	Yes
Require ticket information on activation	No
Require approval to activate	No
Approvers	None

Assignment

Setting	State
Allow permanent eligible assignment	No
Expire eligible assignments after	1 year(s)
Allow permanent active assignment	No
Expire active assignments after	6 month(s)
Require Azure Multi-Factor Authentication on active assignment	No
Require justification on active assignment	Yes



Home > BackupRG > Role setting details - Contributor >

Edit role setting - Contributor ...

Privileged Identity Management | Azure resources

Activation | Assignment | Notification

Activation maximum duration (hours): 2

On activation, require: None [✓ Azure MFA]

- [x] Require justification on activation
- [ ] Require ticket information on activation
- [x] Require approval to activate

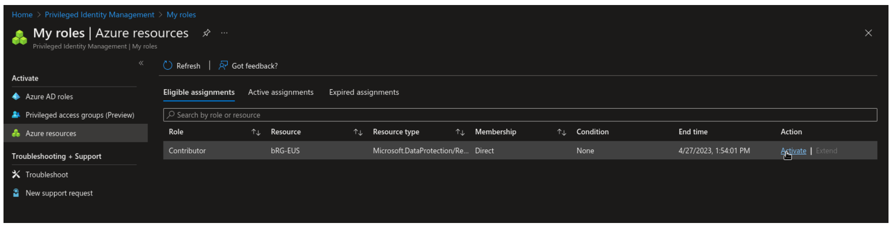
Select approver(s)

1 Member(s), 0 Group(s) selected

Selected approvers:

Update | Next: Assignment

Request access to contributor role Requesting access to the contributor role:

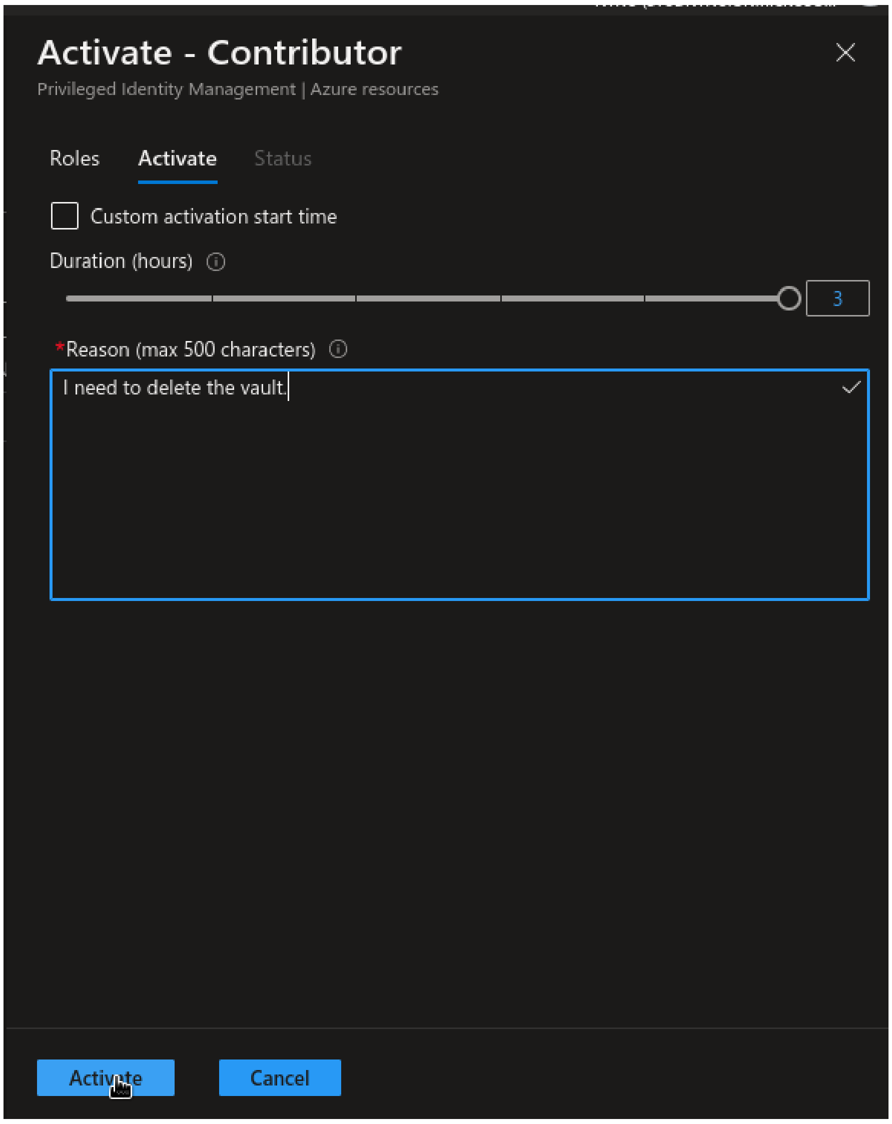

Home > Privileged Identity Management > My roles

My roles | Azure resources

Privileged Identity Management | My roles

Role	Resource	Resource type	Membership	Condition	End time	Action
Contributor	bRG-EUS	Microsoft.DataProtection/Re...	Direct	None	4/27/2023, 1:54:01 PM	Activate | Extend

Sending request:


Activate - Contributor

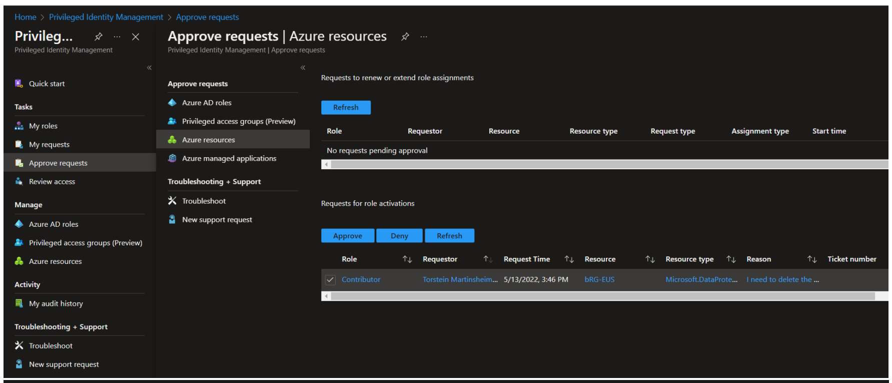
Privileged Identity Management | Azure resources

Custom activation start time: ☐

Duration (hours): 3

*Reason (max 500 characters): I need to delete the vault.

From the security administrators POV, A request has been recieived in the PIM-service:

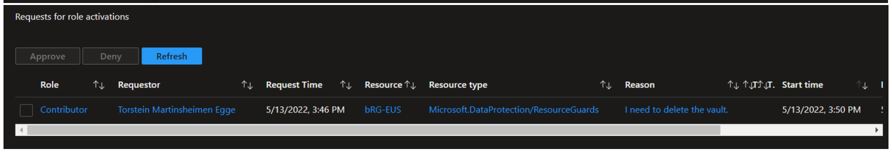

Home > Privileged Identity Management > Approve requests
Privileg...
Privileged Identity Management
Quick start
Tasks
My roles
My requests
Approve requests
Review access
Manage
Azure AD roles
Privileged access groups (Preview)
Azure resources
Activity
My audit history
Troubleshooting + Support
Troubleshoot
New support request
Approve requests | Azure resources
Privileged Identity Management | Approve requests
Approve requests
Azure AD roles
Privileged access groups (Preview)
Azure resources
Azure managed applications
Troubleshooting + Support
Troubleshoot
New support request
Approve
Deny
Refresh

Requests to renew or extend role assignments

Role	Requestor	Resource	Resource type	Request type	Assignment type	Start time
No requests pending approval						

Requests for role activations

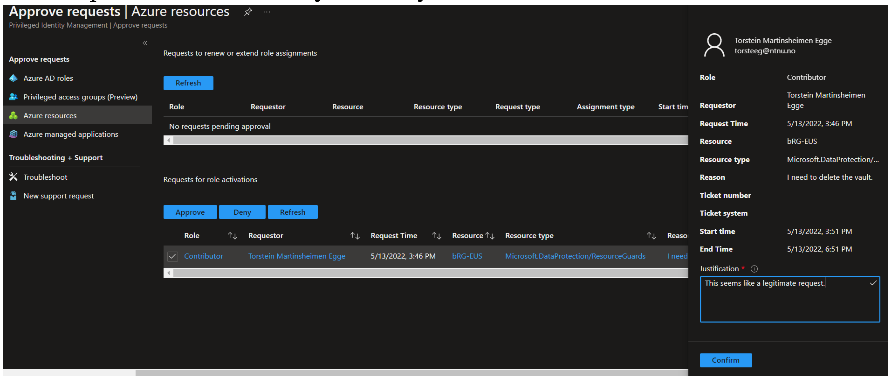
	Role	Requestor	Request Time	Resource	Resource type	Reason	Ticket number
✓	Contributor	Torstein Martinsheim...	5/13/2022, 3:46 PM	bRG-EUS	Microsoft.DataProte...	I need to delete the ...	



Requests for role activations

	Role	Requestor	Request Time	Resource	Resource type	Reason	Start time
	Contributor	Torstein Martinsheimen Egge	5/13/2022, 3:46 PM	bRG-EUS	Microsoft.DataProtection/ResourceGuards	I need to delete the vault.	5/13/2022, 3:50 PM

The request is activated by security admin as shown:


Approve requests | Azure resources
Privileged Identity Management | Approve requests
Approve requests
Azure AD roles
Privileged access groups (Preview)
Azure resources
Azure managed applications
Troubleshooting + Support
Troubleshoot
New support request
Refresh
Approve
Deny
Refresh
Confirm

Requests to renew or extend role assignments

Role	Requestor	Resource	Resource type	Request type	Assignment type	Start tim
No requests pending approval						

Requests for role activations

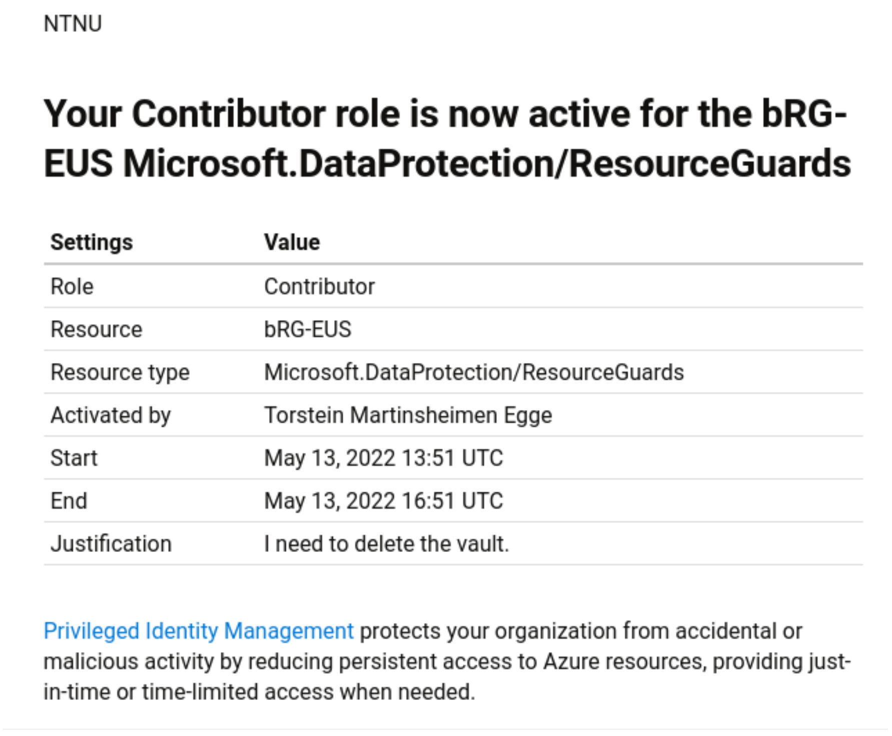
	Role	Requestor	Request Time	Resource	Resource type	Reaso
✓	Contributor	Torstein Martinsheimen Egge	5/13/2022, 3:46 PM	bRG-EUS	Microsoft.DataProtection/ResourceGuards	I need

Torstein Martinsheimen Egge
torsteeg@ntnu.no

Role	Contributor
Requestor	Torstein Martinsheimen Egge
Request Time	5/13/2022, 3:46 PM
Resource	bRG-EUS
Resource type	Microsoft.DataProtection/...
Reason	I need to delete the vault.
Ticket number	
Ticket system	
Start time	5/13/2022, 3:51 PM
End Time	5/13/2022, 6:51 PM

Justification *
This seems like a legitimate request.

An email was received by the backup admin when the request was accepted:



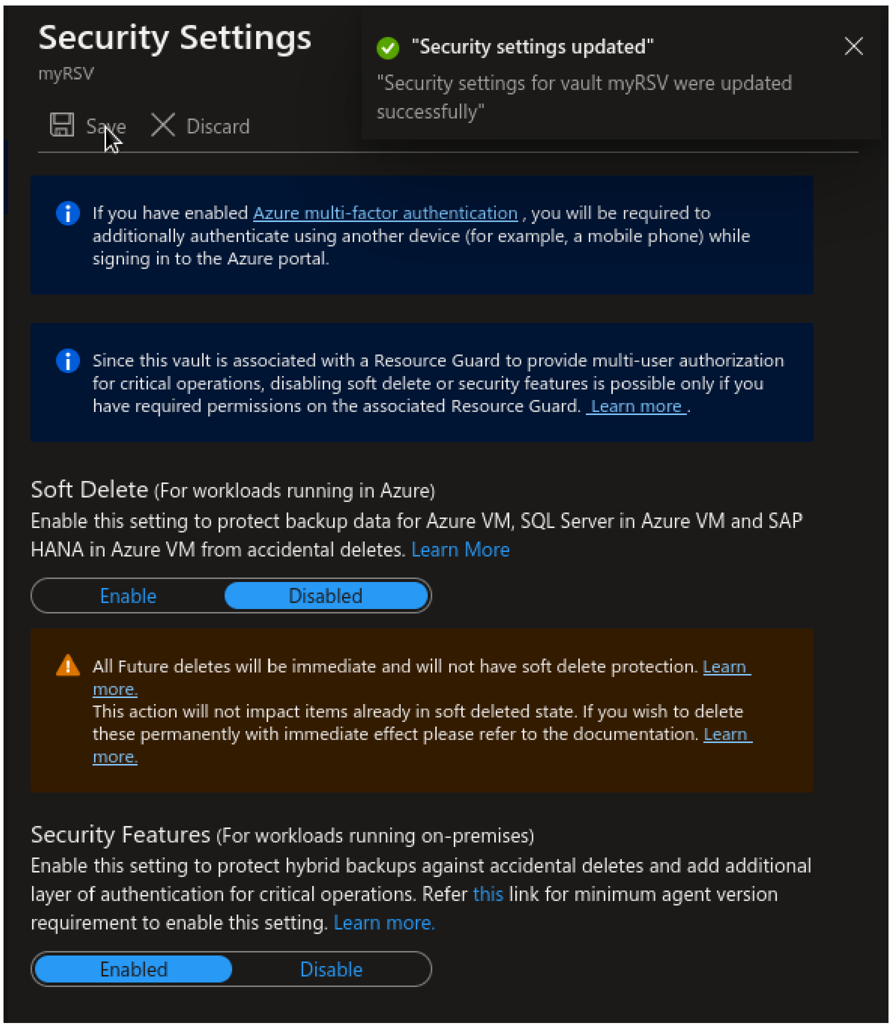
NTNU

Your Contributor role is now active for the bRG-EUS Microsoft.DataProtection/ResourceGuards

Settings	Value
Role	Contributor
Resource	bRG-EUS
Resource type	Microsoft.DataProtection/ResourceGuards
Activated by	Torstein Martinsheimen Egge
Start	May 13, 2022 13:51 UTC
End	May 13, 2022 16:51 UTC
Justification	I need to delete the vault.

Privileged Identity Management protects your organization from accidental or malicious activity by reducing persistent access to Azure resources, providing just-in-time or time-limited access when needed.

Try to disable soft delete Soft delete was successfully disabled for the vault:



Security Settings
myRSV

Save Discard

"Security settings updated"
"Security settings for vault myRSV were updated successfully"

If you have enabled Azure multi-factor authentication , you will be required to additionally authenticate using another device (for example, a mobile phone) while signing in to the Azure portal.

Since this vault is associated with a Resource Guard to provide multi-user authorization for critical operations, disabling soft delete or security features is possible only if you have required permissions on the associated Resource Guard. Learn more .

Soft Delete (For workloads running in Azure)
Enable this setting to protect backup data for Azure VM, SQL Server in Azure VM and SAP HANA in Azure VM from accidental deletes. Learn More

Enable [✓ Disabled]

All Future deletes will be immediate and will not have soft delete protection. Learn more.
This action will not impact items already in soft deleted state. If you wish to delete these permanently with immediate effect please refer to the documentation. Learn more.

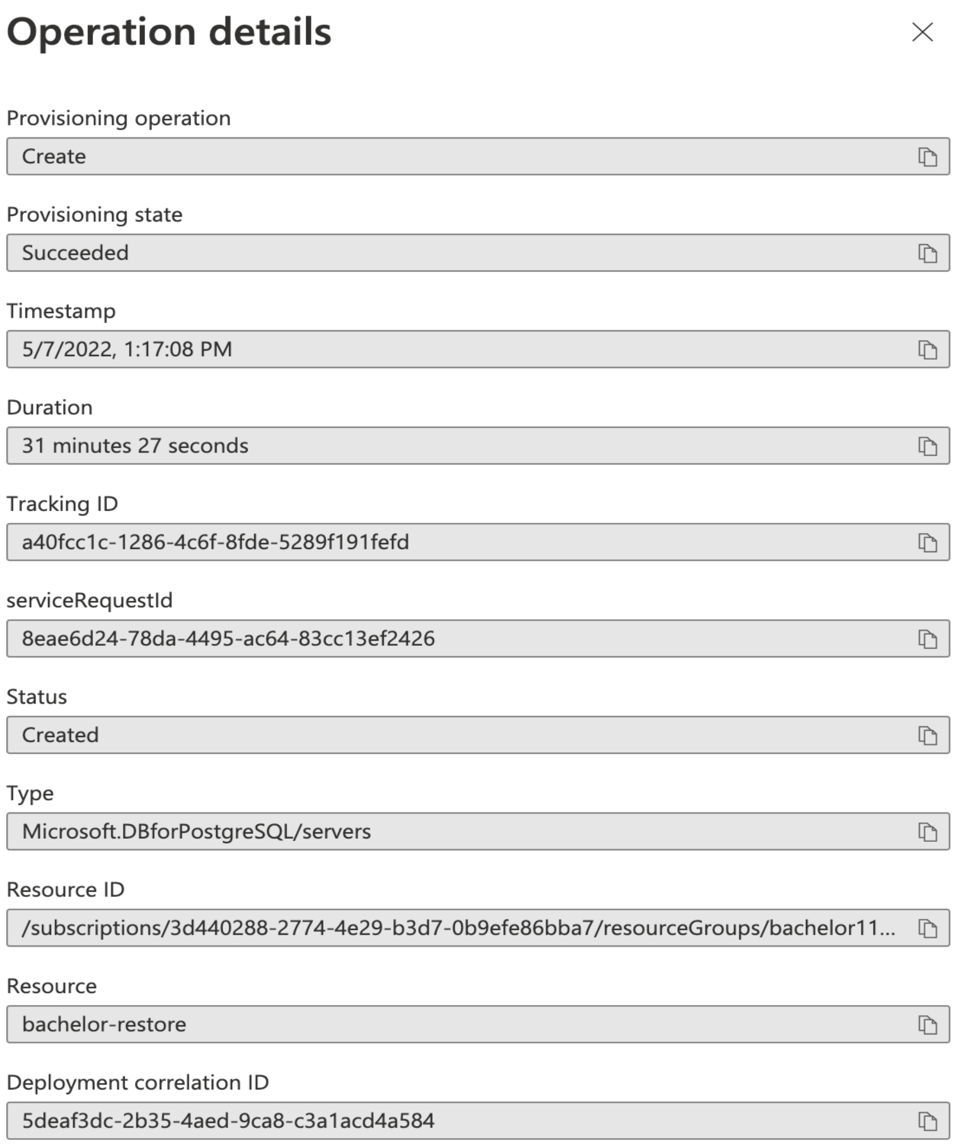
Security Features (For workloads running on-premises)
Enable this setting to protect hybrid backups against accidental deletes and add additional layer of authentication for critical operations. Refer this link for minimum agent version requirement to enable this setting. Learn more.

[✓ Enabled] Disable

A.13 Restore from a backup vault

```
{
   "authorization": {
       "action": "Microsoft.DataProtection/backupVaults/backupInsta
       ↪  nces/restore/action",
       "scope": "/subscriptions/3d440288-2774-4e29-b3d7-0b9efe86bba
       ↪  7/resourceGroups/bachelor119/providers/Microsoft.DataPro
       ↪  tection/backupVaults/bachelor119-performance-backup/back
       ↪  upInstances/bachelor119-performance-postgres-748e4aca-e5
       ↪  3b-4e15-9fe2-665e67d8c03e"
   },
   "caller": "omerj@ntnu.no",
   "channels": "Operation",
"claims": {
    "aud": "https://management.core.windows.net/",
    "iss": "https://sts.windows.net/09a10672-822f-4467-a5ba-5bb3⌋
    ↪  75967c05/",
    "iat": "1651736576",
    "nbf": "1651736576",
    "exp": "1651741076",
    "http://schemas.microsoft.com/claims/authnclassreference":
    ↪  "1",
    "aio": "AVQAq/8TAAAABL3Z7ULGAW1yexY77bzlzAyhUVgZSwf0iL4jELLf⌋
    ↪  gqEExxjWaqv/V018J4GhfqQcZV9PkxbGEDAvPyhTqxWWhMQVZu9S5OlQ⌋
    ↪  B4Ev2uHhnTE=",
    "http://schemas.microsoft.com/claims/authnmethodsreferences"⌋
    ↪  :
    ↪  "pwd,mfa",
    "appid": "c44b4083-3bb0-49c1-b47d-974e53cbdf3c",
    "appidacr": "2",
    "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surna⌋
    ↪  me":
    ↪  "Jonuzi",
    "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/given⌋
    ↪  name":
    ↪  "Omer",
"groups": "8d15c301-2022-4dc2-9550-f36dbdffee7c,61846c99-b2a⌟
 ↪  e-4898-a10c-2ddd7b4a013f,091bd972-9447-4208-b3cc-8a1959a⌟
 ↪  dd436,0107c21a-8a63-4b21-87a0-86a12547fd12,94fefe97-5b83-
 ↪  47a6-a829-01b4add29fe3,5f73363b-e2f9-4001-a71d-b6230f44a⌟
 ↪  293,3ef57554-d71c-41a1-a080-701c2320082c,e1adf917-cdb3-4⌟
 ↪  2cc-8425-79ffe844a86d,498ce0ed-e83d-413e-b529-bde0346ccd⌟
 ↪  f1,a7f6818e-774d-4e09-994f-77fbe6ff7925,aad00940-a490-41⌟
 ↪  c1-8a0c-9b0be5e7a390,be17024f-57a2-4729-a575-75d7d7b63e1⌟
 ↪  4,c497c5c3-4e26-4e2a-8a12-10e0d43288a1,0e414658-efef-480⌟
 ↪  4-a1c4-8a7f668a9e43,7f27023d-8163-41e9-9d11-1cab1ade4d6e⌟
 ↪  ,12e6cbdf-815f-478f-8f9d-3c87b6adc5a9,ab6b0bdb-d48d-4c80-
 ↪  bc61-7fca517eb7cf,37309702-9021-40ba-a2d8-3446dca747f2,e⌟
 ↪  1b09d79-d205-4bbe-9d5f-d79df23d5b56,027504c5-23ce-42ae-8⌟
 ↪  aaa-1c4c2df10ead,43a6e580-ca1f-4f8b-b63a-9b1a3e49b72b,46⌟
 ↪  bc9091-e862-4633-bc54-cbf1473fffe9,01fad63f-0f80-4da7-b9⌟
 ↪  2a-9bf43062d9c3,2433ef53-9c20-4a85-9c8d-136a4375435d,3a8⌟
 ↪  06c10-8ef6-4f00-a9ba-b78faee326a4,587174cf-9b08-44e9-8af⌟
 ↪  5-4d7ac74fa880,9e090282-49ce-4467-94f9-5a85e99c92cf,37db⌟
 ↪  b3c1-c5f7-4c80-a14a-d622c3775824,0ef51352-b5eb-4cac-a483-
 ↪  a9fa6e536129,8370354f-3451-40b2-9abf-a43e7e62d23d,8103e7⌟
 ↪  2f-5acd-4ce9-8c5e-a3af7ba98b36,e29fd27b-746d-4aab-b817-c⌟
 ↪  bc96fc07606,b9e3e345-c7a4-4b86-a1fc-e5b77a8ee6a1,a4295b3⌟
 ↪  f-f9c4-4253-a57a-f42e6ff1a163,9b319622-144f-4573-8a12-a7⌟
 ↪  4b6404ca15,b3df6df0-e4ac-44ca-b0c4-27219aaf7e33,6bac6c57-
 ↪  8c80-4639-9691-0ce756fda9ea,a9bafff9-b09e-4124-9d74-482f⌟
 ↪  4d653c4b,5fa55196-5b42-4d47-8599-a724a00435ab,fd780d1e-8⌟
 ↪  cb6-40b2-80cf-acb411f89833,b47da984-e1de-43cc-9149-8fcca⌟
 ↪  0b24340,42d87878-08fe-42f4-9eaa-cd82f1ffc733",
"ipaddr": "129.241.230.99",
"name": "Omer Jonuzi",
"http://schemas.microsoft.com/identity/claims/objectidentifi⌟
 ↪  er":
 ↪  "3f0b9f76-5dd0-4f4f-85d3-affbe3572a01",
"onprem_sid":
 ↪  "S-1-5-21-3959417778-1711865379-3952174976-358665",
"puid": "100320005758FB15",
"rh":
 ↪  "0.AQkAcgahCS-CZ0SluluzdZZ8BUZIf3kAutdPukPawfj2MBMJAAM.",
"http://schemas.microsoft.com/identity/claims/scope":
 ↪  "user_impersonation",
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/namei⌟
 ↪  dentifier":
 ↪  "i8uDlYJMGSSdVydTjF8rE7LP1tEpil787X1xYh8H-5I",
        "http://schemas.microsoft.com/identity/claims/tenantid":
         ↪  "09a10672-822f-4467-a5ba-5bb375967c05",
        "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name"⌋
         ↪  :
         ↪  "omerj@ntnu.no",
        "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn":
         ↪  "omerj@ntnu.no",
        "uti": "PTjw6g14Z06dvf2WnkwtAA",
        "ver": "1.0",
        "xms_tcdt": "1363351898"
    },
    "correlationId": "0c1c8dd6-246f-412d-9b2a-90baab4b1a1c",
    "description": "",
    "eventDataId": "f714efa3-701c-4673-bcb2-17bcca133cea",
    "eventName": {
        "value": "EndRequest",
        "localizedValue": "End request"
    },
    "category": {
        "value": "Administrative",
        "localizedValue": "Administrative"
    },
    "eventTimestamp": "2022-05-05T08:07:26.1105832Z",
    "id": "/subscriptions/3d440288-2774-4e29-b3d7-0b9efe86bba7/resou⌋
     ↪  rceGroups/bachelor119/providers/Microsoft.DataProtection/bac⌋
     ↪  kupVaults/bachelor119-performance-backup/backupInstances/bac⌋
     ↪  helor119-performance-postgres-748e4aca-e53b-4e15-9fe2-665e67⌋
     ↪  d8c03e/events/f714efa3-701c-4673-bcb2-17bcca133cea/ticks/637⌋
     ↪  873348461105832",
    "level": "Informational",
    "operationId": "c52e2132-8420-446d-b0b8-d7cd42eb2d2e",
    "operationName": {
        "value": "Microsoft.DataProtection/backupVaults/backupInstan⌋
         ↪  ces/restore/action",
        "localizedValue": "Restore Backup Instance"
    },
    "resourceGroupName": "bachelor119",
    "resourceProviderName": {
        "value": "Microsoft.DataProtection",
        "localizedValue": "Microsoft.DataProtection"
    },
    "resourceType": {
        "value":
         ↪  "Microsoft.DataProtection/backupVaults/backupInstances",
        "localizedValue":
         ↪  "Microsoft.DataProtection/backupVaults/backupInstances"
    },
    "resourceId": "/subscriptions/3d440288-2774-4e29-b3d7-0b9efe86bb⌋
     ↪  a7/resourceGroups/bachelor119/providers/Microsoft.DataProtec⌋
     ↪  tion/backupVaults/bachelor119-performance-backup/backupInsta⌋
     ↪  nces/bachelor119-performance-postgres-748e4aca-e53b-4e15-9fe⌋
     ↪  2-665e67d8c03e",
    "status": {
        "value": "Succeeded",
        "localizedValue": "Succeeded"
    },
    "subStatus": {
        "value": "",
        "localizedValue": ""
    },
    "submissionTimestamp": "2022-05-05T08:09:10.1063319Z",
    "subscriptionId": "3d440288-2774-4e29-b3d7-0b9efe86bba7",
    "tenantId": "09a10672-822f-4467-a5ba-5bb375967c05",
    "properties": {
        "eventCategory": "Administrative",
        "entity": "/subscriptions/3d440288-2774-4e29-b3d7-0b9efe86bb⌋
         ↪  a7/resourceGroups/bachelor119/providers/Microsoft.DataPr⌋
         ↪  otection/backupVaults/bachelor119-performance-backup/bac⌋
         ↪  kupInstances/bachelor119-performance-postgres-748e4aca-e⌋
         ↪  53b-4e15-9fe2-665e67d8c03e",
        "message": "Microsoft.DataProtection/backupVaults/backupInst⌋
         ↪  ances/restore/action",
        "hierarchy": "09a10672-822f-4467-a5ba-5bb375967c05/3d440288-⌋
         ↪  2774-4e29-b3d7-0b9efe86bba7"
    },
    "relatedEvents": []
}
```

A.14 Postgres: PITR with REST API performance test



Operation details

Provisioning operation
Create

Provisioning state
Succeeded

Timestamp
5/7/2022, 1:17:08 PM

Duration
31 minutes 27 seconds

Tracking ID
a40fcc1c-1286-4c6f-8fde-5289f191fefd

serviceRequestId
8eae6d24-78da-4495-ac64-83cc13ef2426

Status
Created

Type
Microsoft.DBforPostgreSQL/servers

Resource ID
/subscriptions/3d440288-2774-4e29-b3d7-0b9efe86bba7/resourceGroups/bachelor11...

Resource
bachelor-restore

Deployment correlation ID
5deaf3dc-2b35-4aed-9ca8-c3a1acd4a584

Figure A.1: PITR deploys the database in a newly created server

A.15 Setup of test environment for PostgreSQL experiments

A.15.1 Creating test environment for PostgreSQL

The PostgreSQL instance was

The Powershell script below is the template for our test environment on the

Azure Database for PostgreSQL Single Server.

```
    $Password = Read-Host -Prompt 'Please enter your password' -AsSecureString
    $RGName = "myresourcegroup"

    az group create --name $RGName --location northeurope

    New-AzPostgreSqlServer `
        -Name bachelorgroup119postgrestest1 `
        -ResourceGroupName $RGName `
        -Sku GP_Gen5_2 `
        -GeoRedundantBackup Enabled `
        -Location northeurope `
        -AdministratorUsername myadmin `
        -AdministratorLoginPassword $Password
```

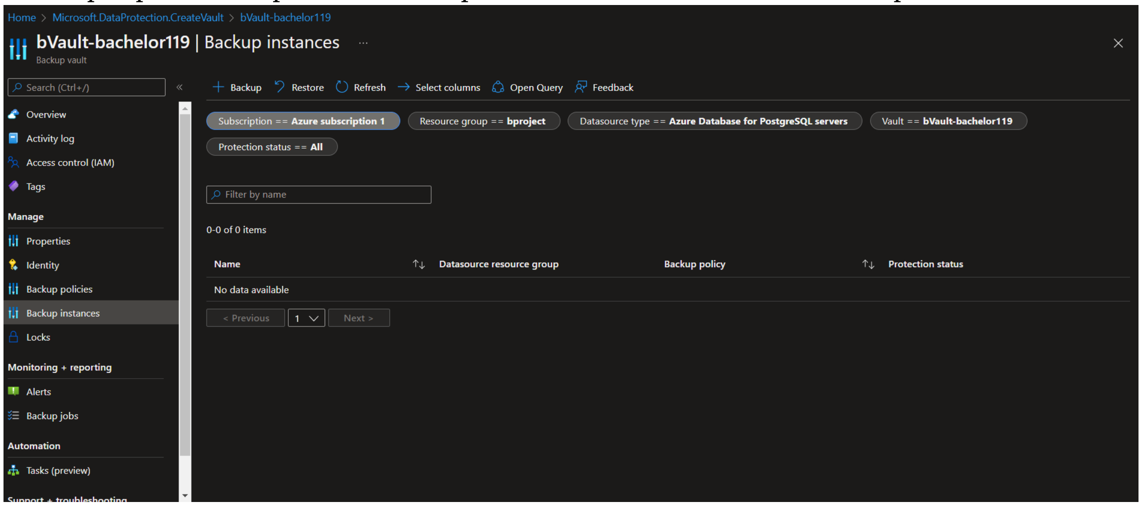
A.15.2 Data population

The script below was used to populate the database.

```
import psycopg2
# Connection string information
host = "bachelor119-performance.postgres.database.azure.com"
dbname = "postgres"
user = "bachelor119@bachelor119-performance"
password = "performance-test1"
sslmode = "require"
# Connection string constructed
conn_string = "host={0} user={1} dbname={2} password={3}
↪  sslmode={4}".format(host, user, dbname, password, sslmode)
conn = psycopg2.connect(conn_string)
cursor = conn.cursor()
# Check for duplicate tables
cursor.execute("DROP TABLE IF EXISTS inventory;")
# Create table
cursor.execute("CREATE TABLE inventory (id serial PRIMARY KEY, one
↪  BIGINT, two BIGINT, three BIGINT);")
# Insert some data into the table
cursor.execute("INSERT INTO inventory (one, two, three) SELECT
↪  generate_series(1, 90000000), generate_series(1, 90000000),
↪  generate_series(1, 90000000);")
# Resolve connection
conn.commit()
cursor.close()
conn.close()
```

A.15.3 Postgres: Azure backup setup

In order to back up a postgres server with Azure Backup the first step is to click the "Backup"-option on top of the "Backup instances"-blade inside the Backup Vault.



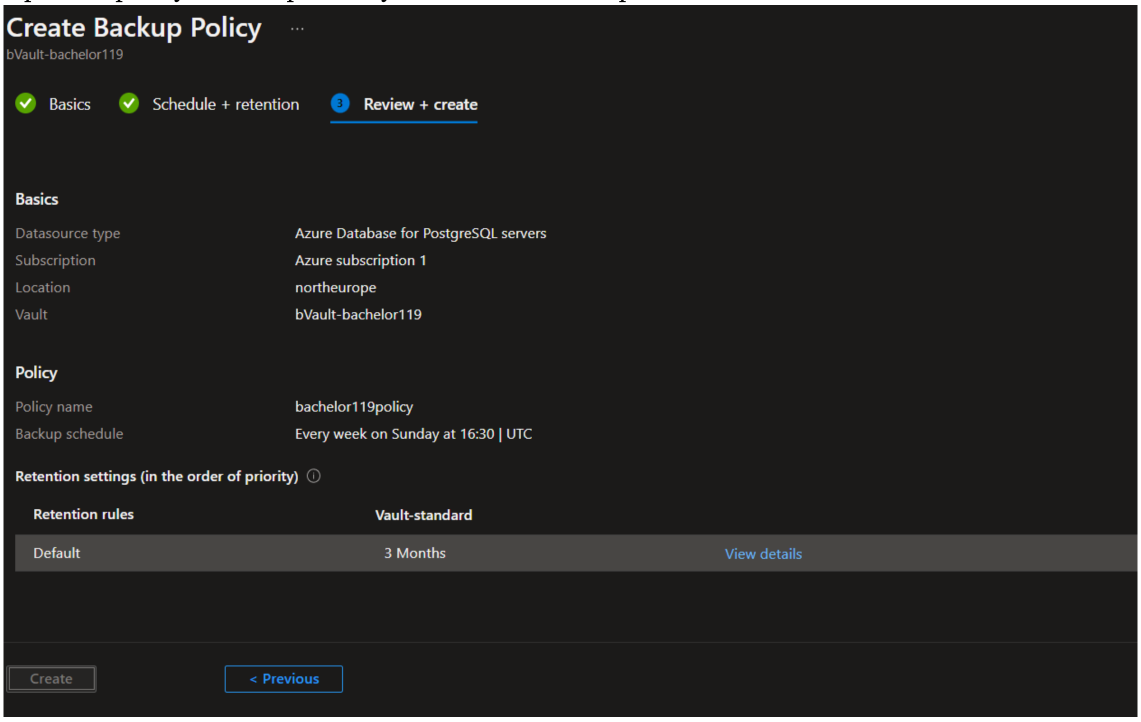
bVault-bachelor119 | Backup instances

Subscription == **Azure subscription 1** Resource group == **bproject** Datasource type == **Azure Database for PostgreSQL servers** Vault == **bVault-bachelor119**
Protection status == **All**

0-0 of 0 items

Name	Datasource resource group	Backup policy	Protection status
No data available			

In the following wizard each organisation must select the options that suit their needs, but importantly it requires the creation of a backup policy. In our example the policy backs up weekly and retains backups for three months.



Create Backup Policy
bVault-bachelor119

Basics

Datasource type	Azure Database for PostgreSQL servers
Subscription	Azure subscription 1
Location	northeurope
Vault	bVault-bachelor119

Policy

Policy name	bachelor119policy
Backup schedule	Every week on Sunday at 16:30 | UTC

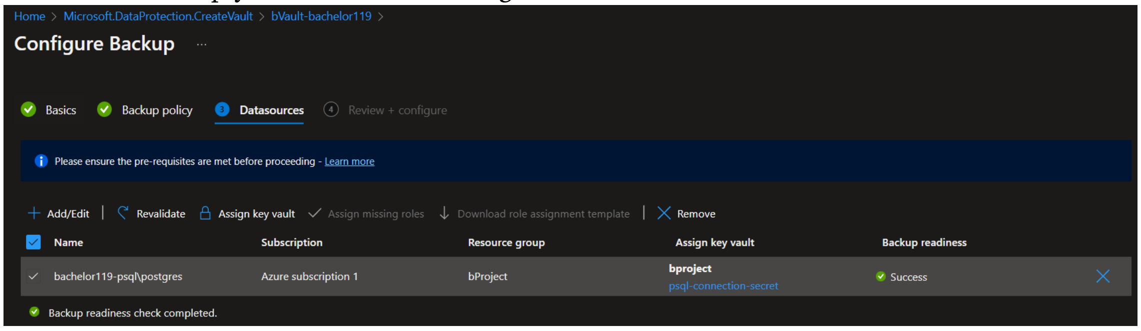
Retention settings (in the order of priority)

Retention rules	Vault-standard	
Default	3 Months	View details

Additionally when backing up managed postgres servers, we can select which databases to back up. Here we have chosen the standard postgres-database inside the instance, but had we had others we could have chosen them as well.

When selecting the database we must also give the secret that is required to connect to the database. This can be given as an object i a key vault, as we have done here:

This secret is simply the connection string from the database instance.



Home > Microsoft.DataProtection.CreateVault > bVault-bachelor119 >

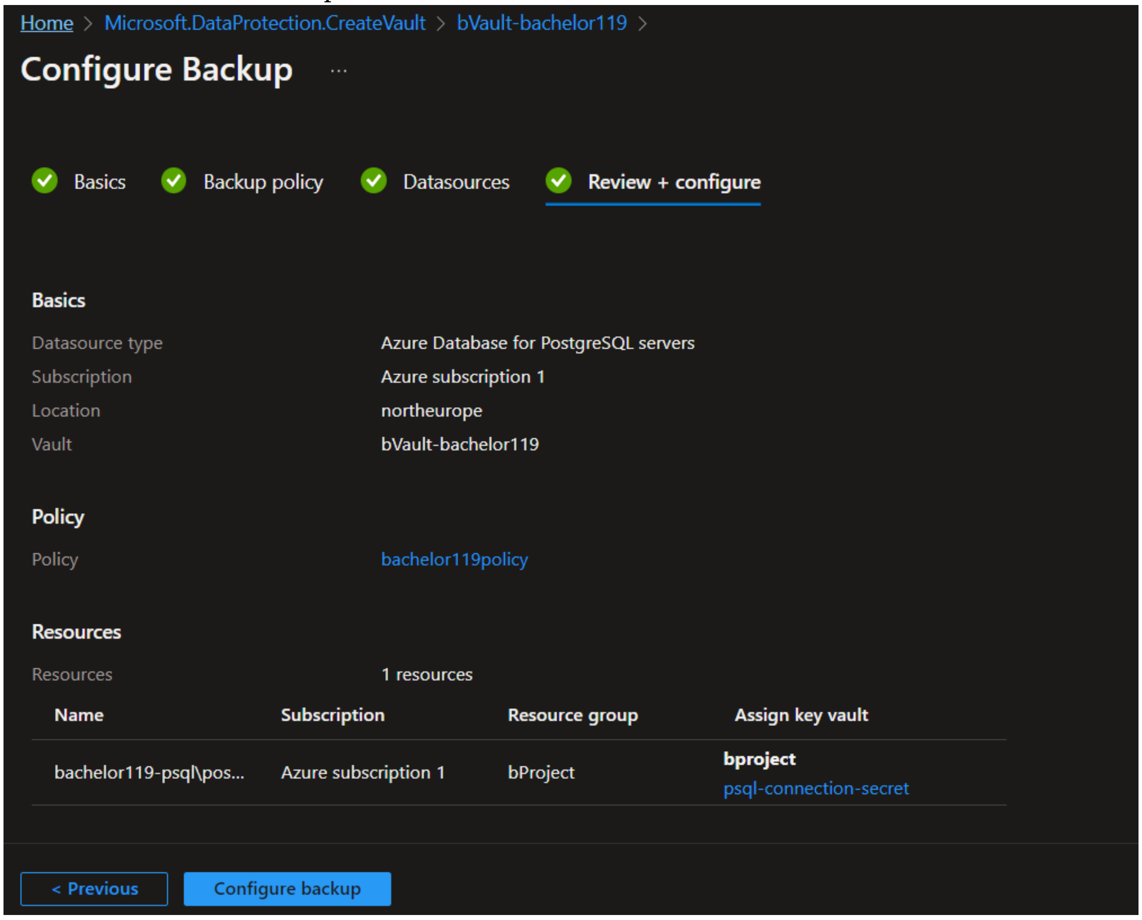
Configure Backup ...

Please ensure the pre-requisites are met before proceeding - Learn more

Name	Subscription	Resource group	Assign key vault	Backup readiness
bachelor119-psql\postgres	Azure subscription 1	bProject	bproject psql-connection-secret	Success

Backup readiness check completed.

Finally we have the option to review our options, and if we are happy with these we can create the backup instance.



Home > Microsoft.DataProtection.CreateVault > bVault-bachelor119 >

Configure Backup ...

Basics

Datasource type	Azure Database for PostgreSQL servers
Subscription	Azure subscription 1
Location	northeurope
Vault	bVault-bachelor119

Policy

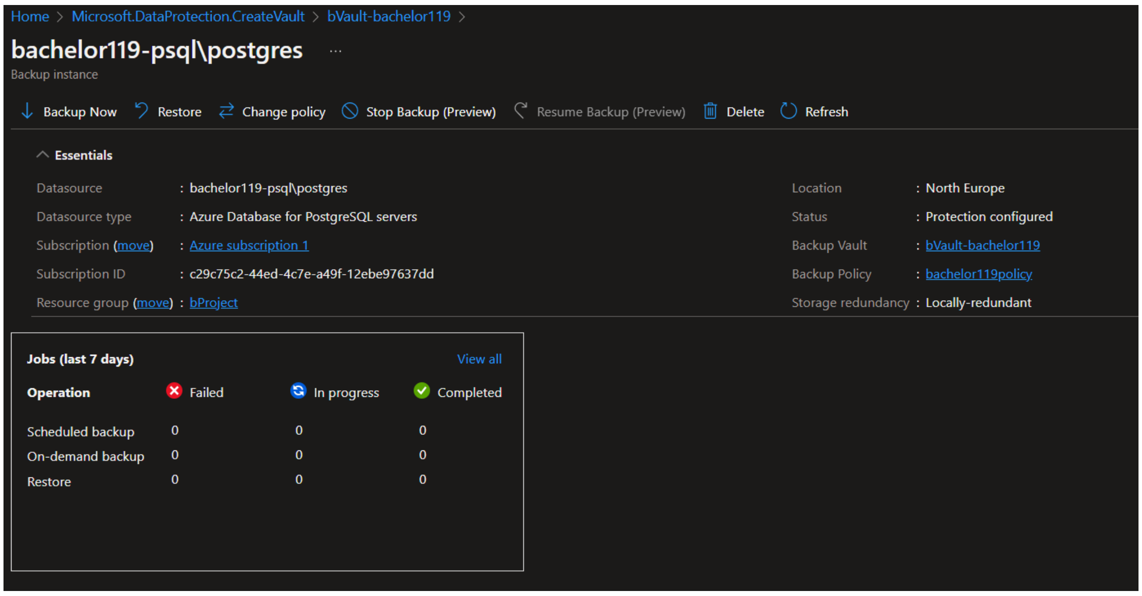
Policy	bachelor119policy

Resources

Resources	1 resources

Name	Subscription	Resource group	Assign key vault
bachelor119-psql\pos...	Azure subscription 1	bProject	bproject psql-connection-secret

Immediately after creation the backup instance have not started a backup, and it will not back up the database until we tell it to, or the policy determines that it shall. As shown no backups are completed:



Home > Microsoft.DataProtection.CreateVault > bVault-bachelor119 >

bachelor119-psql\postgres ...

Backup instance

Backup Now | Restore | Change policy | Stop Backup (Preview) | Resume Backup (Preview) | Delete | Refresh

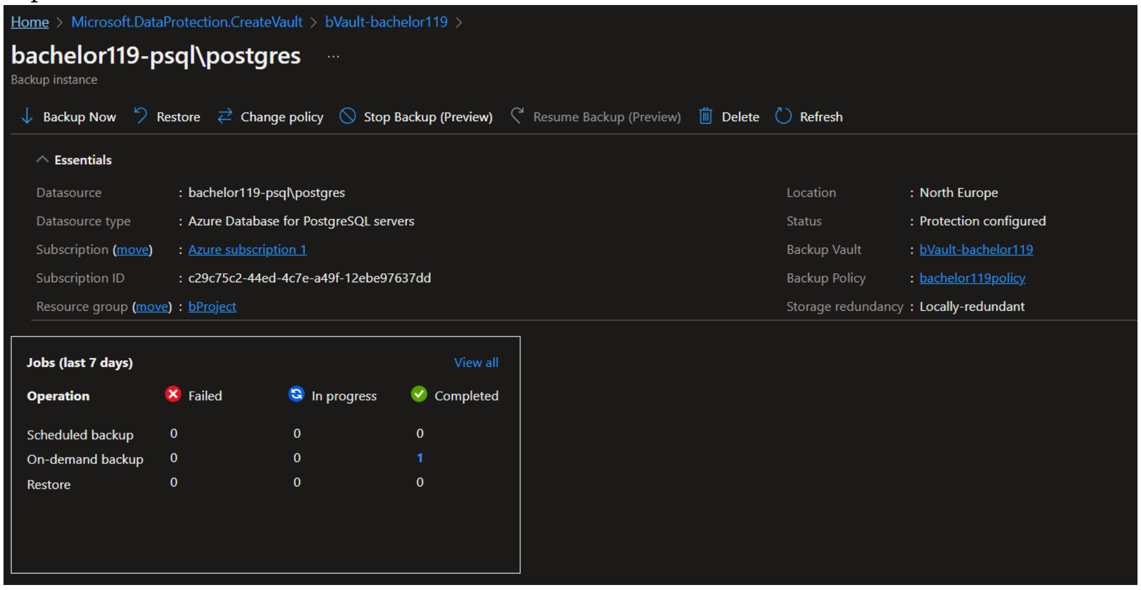
Essentials

Datasource	: bachelor119-psql\postgres	Location	: North Europe
Datasource type	: Azure Database for PostgreSQL servers	Status	: Protection configured
Subscription (move)	: Azure subscription 1	Backup Vault	: bVault-bachelor119
Subscription ID	: c29c75c2-44ed-4c7e-a49f-12ebe97637dd	Backup Policy	: bachelor119policy
Resource group (move)	: bProject	Storage redundancy	: Locally-redundant

Jobs (last 7 days) View all

Operation	Failed	In progress	Completed
Scheduled backup	0	0	0
On-demand backup	0	0	0
Restore	0	0	0

If we press "back up now" and let it run to completion, we have one completed backup however:



Home > Microsoft.DataProtection.CreateVault > bVault-bachelor119 >

bachelor119-psql\postgres ...

Backup instance

Backup Now | Restore | Change policy | Stop Backup (Preview) | Resume Backup (Preview) | Delete | Refresh

Essentials

Datasource	: bachelor119-psql\postgres	Location	: North Europe
Datasource type	: Azure Database for PostgreSQL servers	Status	: Protection configured
Subscription (move)	: Azure subscription 1	Backup Vault	: bVault-bachelor119
Subscription ID	: c29c75c2-44ed-4c7e-a49f-12ebe97637dd	Backup Policy	: bachelor119policy
Resource group (move)	: bProject	Storage redundancy	: Locally-redundant

Jobs (last 7 days) View all

Operation	Failed	In progress	Completed
Scheduled backup	0	0	0
On-demand backup	0	0	1
Restore	0	0	0

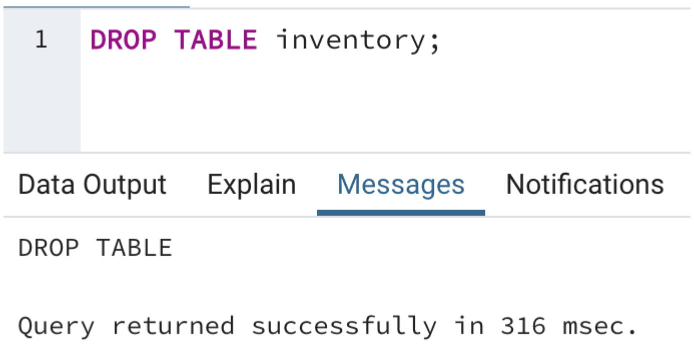
A.16 Postgres: Restoring from PITR

In this experiment we simluated data loss in a postgres database and restored the data with Point-in-time-Restore.

Given that an attacker encrypts data files in a database in a ransomware attack, we assume complete data loss in the database. This can be simulated with a database (consisting of a single table) being dropped. This is because the security measures designed to mitigate such an attack work the same regardless of the state of the database.

```
postgres=> SELECT pg_size_pretty( pg_database_size('postgres') );
 pg_size_pretty
----------------
 1264 MB
(1 row)
```

Figure A.2: The size of the inventory table



```
DROP TABLE inventory;
```

Data Output | Explain | Messages | Notifications

```
DROP TABLE

Query returned successfully in 316 msec.
```

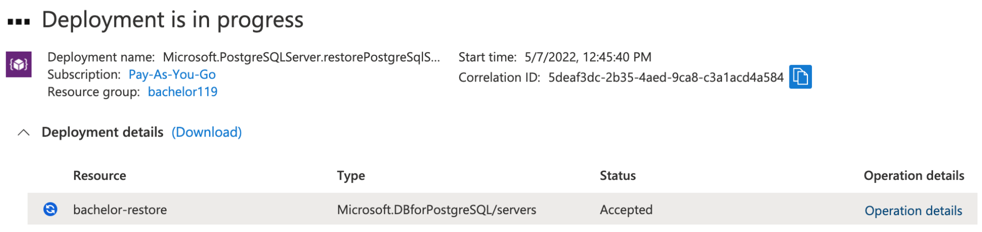
Figure A.3: Dropping the table

```
postgres=> SELECT pg_size_pretty( pg_database_size('postgres') );
 pg_size_pretty
----------------
 8173 kB
(1 row)
```

Figure A.4: The size of the inventory table after dropped table

The first security measure is also the simplest solution Azure provides for postgres databases: Point-in-time Restore (PITR). Though there are multiple ways to back up our database.

A point-in-time restore allows the database to be reverted back to a previous state where data has not been tampered with. The user specifies a point in time at which a copy of the database is made and used in the creation of a new managed database server. Such an operation is demonstrated in the following deployment of a restoration database.



Deployment is in progress

Deployment name: Microsoft.PostgreSQLServer.restorePostgreSqlS...
Subscription: Pay-As-You-Go
Resource group: bachelor119
Start time: 5/7/2022, 12:45:40 PM
Correlation ID: 5deaf3dc-2b35-4aed-9ca8-c3a1acd4a584

Deployment details (Download)

Resource	Type	Status	Operation details
bachelor-restore	Microsoft.DBforPostgreSQL/servers	Accepted	Operation details

Figure A.5: The restore operation took 31 minutes 27 seconds to complete

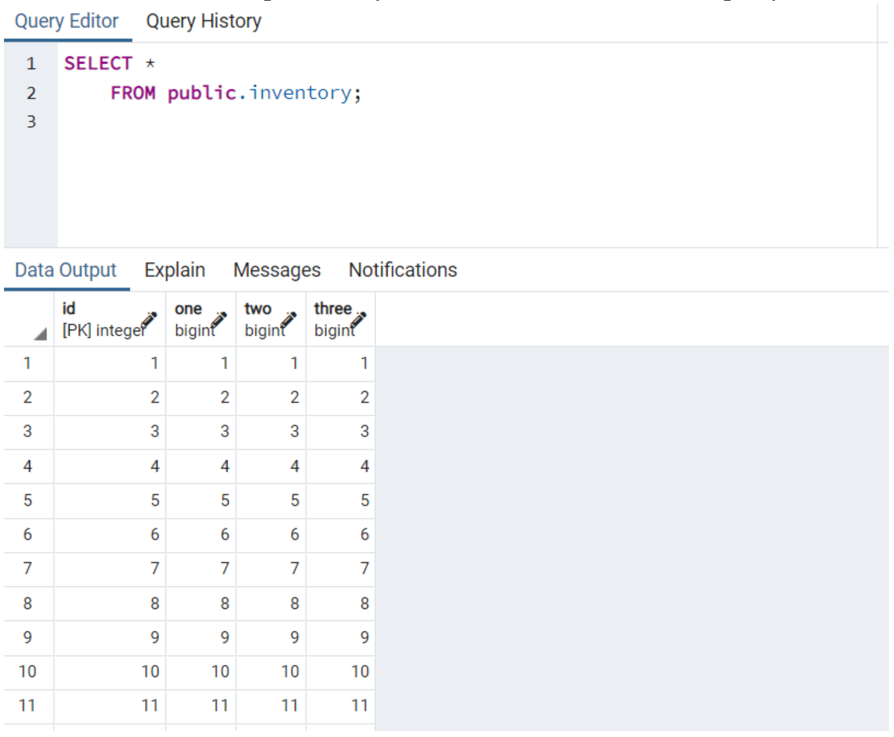
This way one can recover to the most updated version of the database possible while removing all remnants of malicious activity.

The policy for backup retention is dependent on the size of the server. Since we have a 7-day retention period set on the server and the server is no more than 4 TB the backup storage will retain 2 full backups, as well differential backups along with transaction logs from the point of the last full backup. Further information about how server size affects backup and restore policy can be found in the Azure docs for PostgreSQL single server.

A.17 Postgres: Protecting with Azure Backup

A.17.1 Restoring from Azure Backup

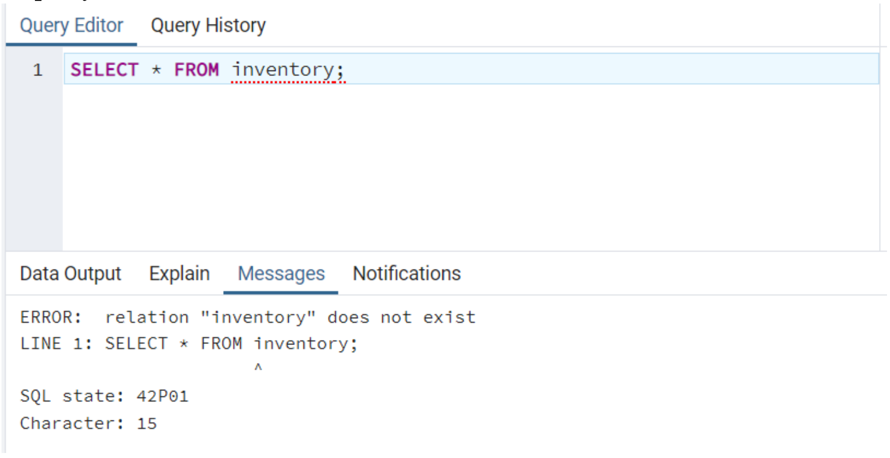
The data in our database when performing this test is simply 10000 lines of auto-generated code as shown previously. The result of a "SELECT *"-query is as follows:



```
SELECT *
    FROM public.inventory;
```

	id [PK] integer	one bigint	two bigint	three bigint
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	5	5	5	5
6	6	6	6	6
7	7	7	7	7
8	8	8	8	8
9	9	9	9	9
10	10	10	10	10
11	11	11	11	11

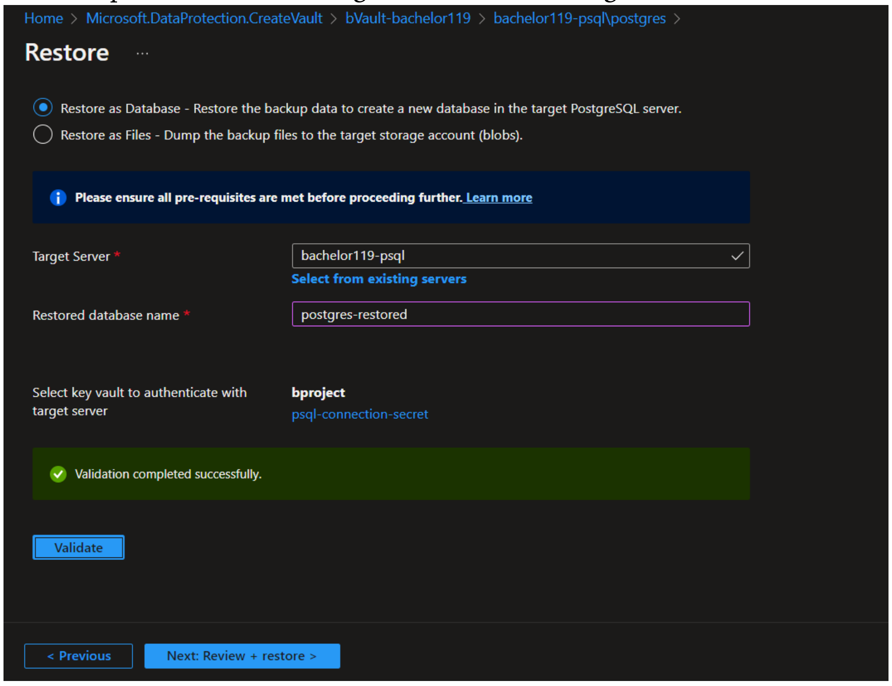
To simulate data loss, we lose some data by dropping table inventory. The same query now returns an error:



```
SELECT * FROM inventory;
```

```
ERROR:  relation "inventory" does not exist
LINE 1: SELECT * FROM inventory;
                      ^
SQL state: 42P01
Character: 15
```

In order to restore from our backup in Azure Backup, we press the "restore"-button on top of the blade. This gives us the following wizard:


Home > Microsoft.DataProtection.CreateVault > bVault-bachelor119 > bachelor119-psql\postgres >
Restore ...
Validate
< Previous
Next: Review + restore >

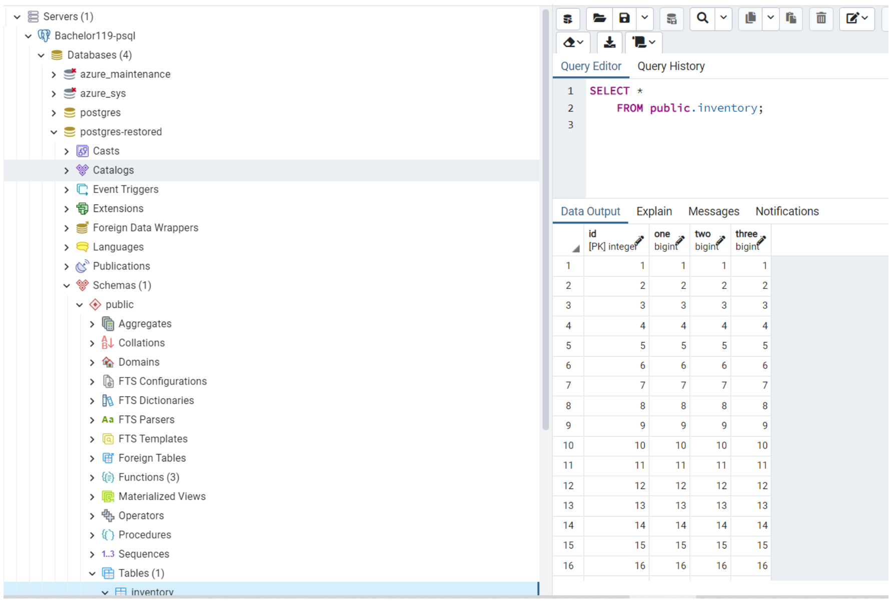
(●) Restore as Database - Restore the backup data to create a new database in the target PostgreSQL server.
( ) Restore as Files - Dump the backup files to the target storage account (blobs).

Please ensure all pre-requisites are met before proceeding further. Learn more

Target Server *	bachelor119-psql Select from existing servers
Restored database name *	postgres-restored
Select key vault to authenticate with target server	bproject psql-connection-secret

Validation completed successfully.

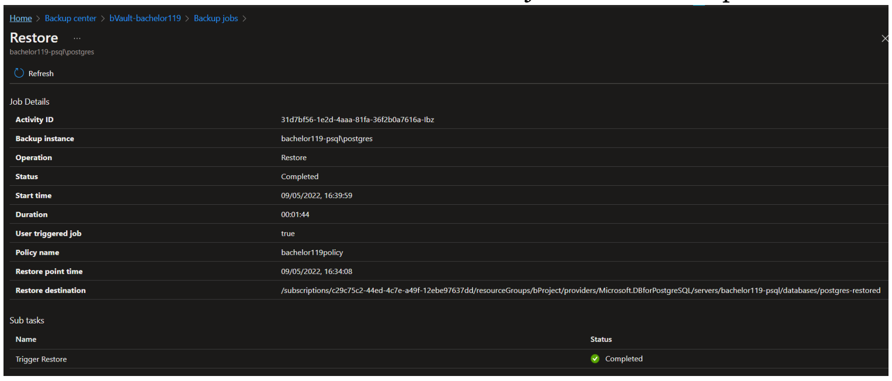
As shown, we are given the option between restoring files or as a database to a server. In this example we choose to restore as a new database on the same server as before. We name this new database "postgres-restored." The same query as before can now be shown on the restored database:



```
SELECT *
    FROM public.inventory;
```

	id [PK] integer	one bigint	two bigint	three bigint
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	5	5	5	5
6	6	6	6	6
7	7	7	7	7
8	8	8	8	8
9	9	9	9	9
10	10	10	10	10
11	11	11	11	11
12	12	12	12	12
13	13	13	13	13
14	14	14	14	14
15	15	15	15	15
16	16	16	16	16

After restoration we can see the successful job in the backup vault:



Restore

bachelor119-psql\postgres

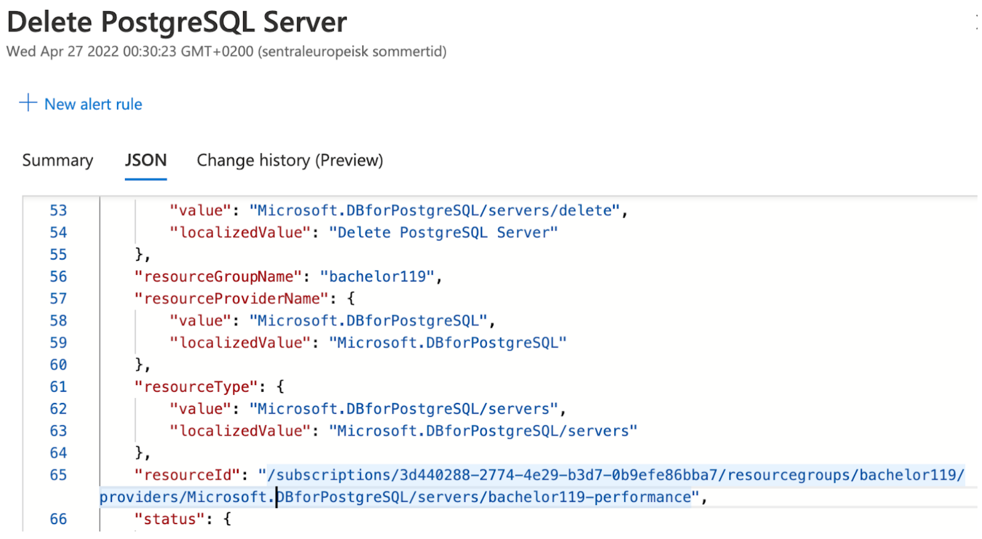
Job Details

Activity ID	31d7bf56-1e2d-4aaa-81fa-36f2b0a7616a-lbz
Backup instance	bachelor119-psql\postgres
Operation	Restore
Status	Completed
Start time	09/05/2022, 16:39:59
Duration	00:01:44
User triggered job	true
Policy name	bachelor119policy
Restore point time	09/05/2022, 16:34:08
Restore destination	/subscriptions/c29c75c2-44ed-4c7e-a49f-12ebe97637dd/resourceGroups/bProject/providers/Microsoft.DBforPostgreSQL/servers/bachelor119-psql/databases/postgres-restored

Sub tasks

Name	Status
Trigger Restore	✓ Completed

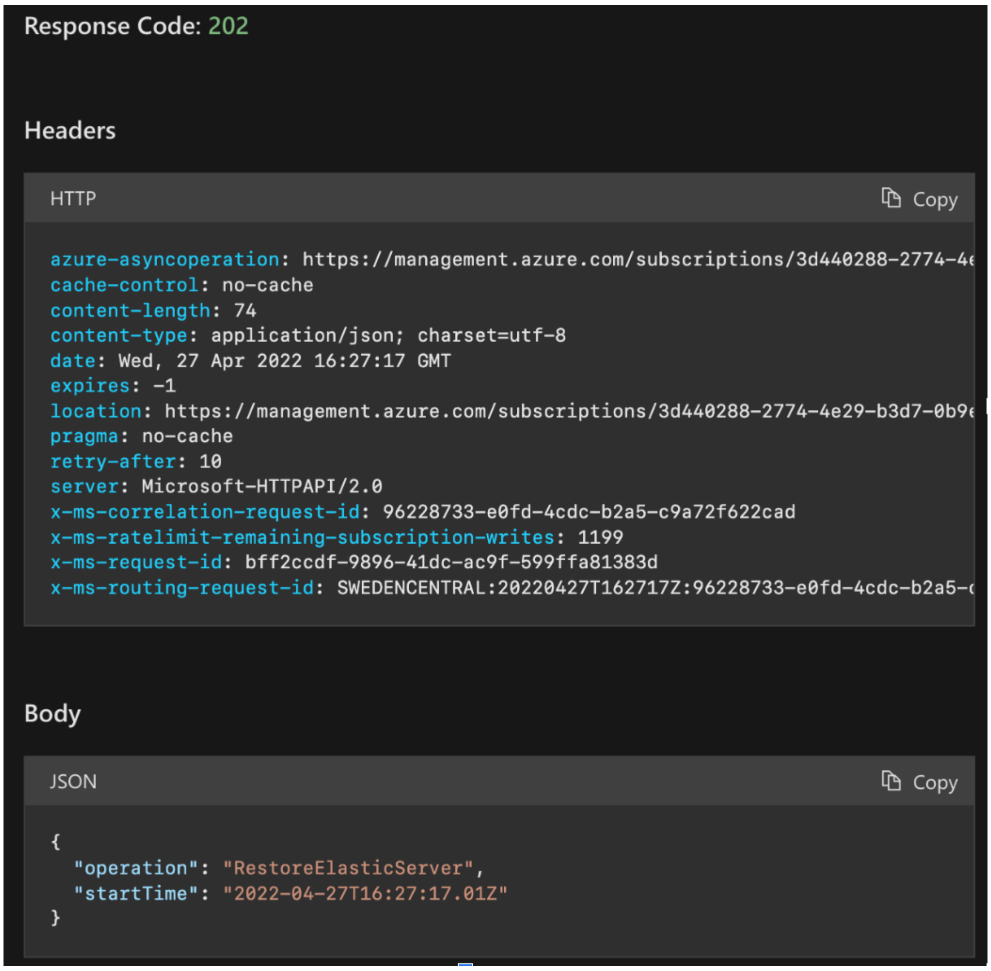
A.18 Postgres: Deleting database and restoring from PITR



```
        "value": "Microsoft.DBforPostgreSQL/servers/delete",
        "localizedValue": "Delete PostgreSQL Server"
    },
    "resourceGroupName": "bachelor119",
    "resourceProviderName": {
        "value": "Microsoft.DBforPostgreSQL",
        "localizedValue": "Microsoft.DBforPostgreSQL"
    },
    "resourceType": {
        "value": "Microsoft.DBforPostgreSQL/servers",
        "localizedValue": "Microsoft.DBforPostgreSQL/servers"
    },
    "resourceId": "/subscriptions/3d440288-2774-4e29-b3d7-0b9efe86bba7/resourcegroups/bachelor119/
providers/Microsoft.DBforPostgreSQL/servers/bachelor119-performance",
    "status": {
```

Figure A.6: JSON output of the delete operation

However, this is a form of soft delete since the Azure backup ensures that it is possible to redeploy the server after deletion within the retention period of said server. In addition, when creating the server the property of "createMode" can be set to "PointInTimeRestore". This can be done through the REST API endpoint for creating servers, where the resource ID of the deleted server has to be included in the request body. The resource ID of the deleted server can be found in the JSON file of the delete operation that should be stored in the Azure portal activity log. Below we demonstrate the recovery of the server. This way we establish that a server that was encrypted and then deleted still has the ability to fully recover to an operational pre-encryption state.


Response Code: 202

Headers

HTTP Copy

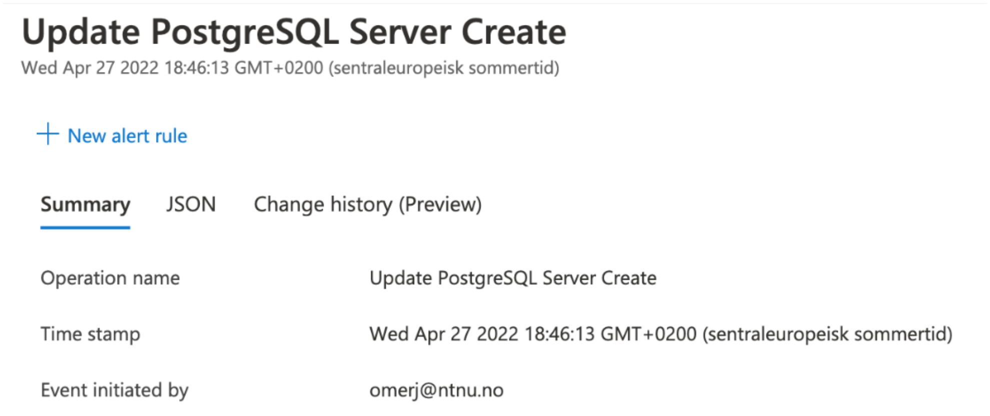
```
azure-asyncoperation: https://management.azure.com/subscriptions/3d440288-2774-4e
cache-control: no-cache
content-length: 74
content-type: application/json; charset=utf-8
date: Wed, 27 Apr 2022 16:27:17 GMT
expires: -1
location: https://management.azure.com/subscriptions/3d440288-2774-4e29-b3d7-0b9e
pragma: no-cache
retry-after: 10
server: Microsoft-HTTPAPI/2.0
x-ms-correlation-request-id: 96228733-e0fd-4cdc-b2a5-c9a72f622cad
x-ms-ratelimit-remaining-subscription-writes: 1199
x-ms-request-id: bff2ccdf-9896-41dc-ac9f-599ffa81383d
x-ms-routing-request-id: SWEDENCENTRAL:20220427T162717Z:96228733-e0fd-4cdc-b2a5-c
```

Body

JSON Copy

```
{
  "operation": "RestoreElasticServer",
  "startTime": "2022-04-27T16:27:17.01Z"
}
```

Figure A.7: Summary of the API call


Update PostgreSQL Server Create

Wed Apr 27 2022 18:46:13 GMT+0200 (sentraleuropeisk sommertid)

Operation name	Update PostgreSQL Server Create
Time stamp	Wed Apr 27 2022 18:46:13 GMT+0200 (sentraleuropeisk sommertid)
Event initiated by	omerj@ntnu.no

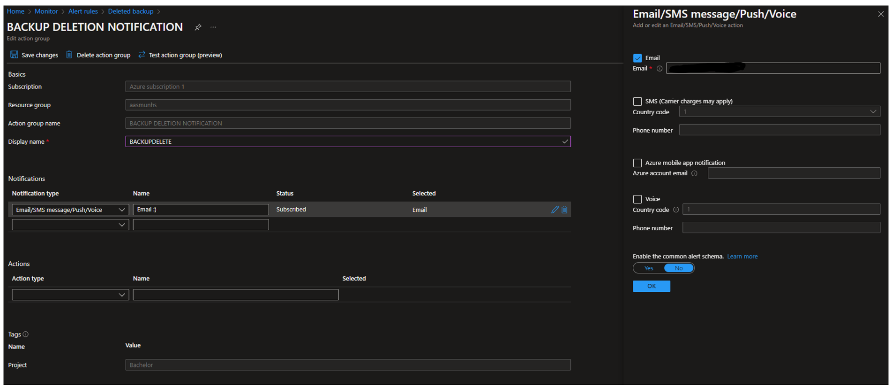
Figure A.8: Server created and database restored

A.19 Postgres: Enabling alerts for Backup vault instance deletion

The following template shows how an alert for deleted backup instances can be created. Worth noting that the only supported scope is for the whole subscription, and not a single resource group or Backup vault.

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04
     ↪  -01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "activityLogAlerts_Deleted_backup_name": {
            "defaultValue": "Deleted backup",
            "type": "String"
        },
        "actiongroups_backup_deletion_notification_externalid": {
            "defaultValue": "/subscriptions/c29c75c2-44ed-4c7e-a49f
             ↪  -12ebe97637dd/resourceGroups/aasmunhs/providers/mic
             ↪  rosoft.insights/actiongroups/backup deletion
             ↪  notification",
            "type": "String"
        }
    },
    "variables": {},
    "resources": [
        {
            "type": "microsoft.insights/activityLogAlerts",
            "apiVersion": "2020-10-01",
            "name": "[parameters('activityLogAlerts_Deleted_backup_
             ↪  name')]",
            "location": "Global",
            "properties": {
                "scopes": [
                    "/subscriptions/c29c75c2-44ed-4c7e-a49f-12ebe97
                     ↪  637dd"
                ],
                "condition": {
                    "allOf": [
                        {
                            "field": "category",
                            "equals": "Administrative"
                        },
                        {
                            "field": "operationName",
                            "equals": "Microsoft.DataProtection/bac⌋
                            ↪  kupVaults/backupInstances/delete"
                        },
                        {
                            "field": "level",
                            "containsAny": [
                                "informational"
                            ]
                        },
                        {
                            "field": "status",
                            "containsAny": [
                                "succeeded"
                            ]
                        }
                    ]
                },
                "actions": {
                    "actionGroups": [
                        {
                            "actionGroupId":
                            ↪  "[parameters('actiongroups_backup_d⌋
                            ↪  eletion_notification_externalid')]",
                            "webhookProperties": {}
                        }
                    ]
                },
                "enabled": true
            }
        }
    ]
}
```

As shown, notification type can be modified in the action group:


Home > Monitor > Alert rules > Deleted backup >

BACKUP DELETION NOTIFICATION

Edit action group

Save changes | Delete action group | Test action group (preview)

Basics

Subscription	Azure subscription 1
Resource group	aasmunhs
Action group name	BACKUP DELETION NOTIFICATION
Display name *	BACKUPDELETE

Notifications

Notification type	Name	Status	Selected
Email/SMS message/Push/Voice	Email :)	Subscribed	Email

Actions

Action type	Name	Selected

Tags

Name	Value
Project	Bachelor

Email/SMS message/Push/Voice

Add or edit an Email/SMS/Push/Voice action

☑ Email

Email * [redacted]

☐ SMS (Carrier charges may apply)

Country code: 1

Phone number:

☐ Azure mobile app notification

Azure account email:

☐ Voice

Country code: 1

Phone number:

Enable the common alert schema. Learn more

Yes [✓ No]

OK

When deleting a postgres-backup instance the following email was sent:

```
Azure Monitor alert 'Deleted backup' was activated for 'bachelor119⌋
 ↪  -psql-postgres-restored-748229ab-ebae-4836-baf5-87bb4150b08c'
 ↪  at May 9, 2022 22:36 UTC
You’re receiving this notification as a member of the BACKUP DELET
 ↪  action group because an Azure Monitor alert was
 ↪  activated.

Activity log alert        Deleted backup
Time          May 9, 2022 22:36 UTC
Category          Administrative
Operation name          Microsoft.DataProtection/backupVaults/backupI⌋
 ↪  nstances/delete
Correlation ID          74d31364-31cb-497a-8c8a-359a81d5b843
Level           Informational
Resource ID          /subscriptions/c29c75c2-44ed-4c7e-a49f-12ebe9763⌋
 ↪  7dd/resourceGroups/bProject/providers/Microsoft.DataProtection/⌋
 ↪  backupVaults/bVault-bachelor119/backupInstances/bachelor119-psq⌋
 ↪  l-postgres-restored-748229ab-ebae-4836-baf5-87bb4150b08c
Caller          aasmunhs@ntnu.no

Properties          {"eventCategory":"Administrative","entity":"/subs⌋
 ↪  criptions/c29c75c2-44ed-4c7e-a49f-12ebe97637dd/resourceGroups/b⌋
 ↪  Project/providers/Microsoft.DataProtection/backupVaults/bVault-⌋
 ↪  bachelor119/backupInstances/bachelor119-psql-postgres-restored-⌋
 ↪  748229ab-ebae-4836-baf5-87bb4150b08c","message":"Microsoft.Data⌋
 ↪  Protection/backupVaults/backupInstances/delete","hierarchy":"09⌋
 ↪  a10672-822f-4467-a5ba-5bb375967c05/c29c75c2-44ed-4c7e-a49f-12eb⌋
 ↪  e97637dd"}
```

NTNU

Kunnskap for en bedre verden