NTNU | Norwegian University of Science and Technology

DCSG2900 - Bachelor Thesis Bachelor of Science in Digital Infrastructure and Cyber Security 2021

Security within a multi-tenant Kubernetes cluster

Authors:
Simen Asplund Kjeserud
Vemund Rahm
Sonja Yoosuk Sanden
Eirik Tobiassen

19th May 2021

Sammendrag

Multi-tenancy er et konsept som begynner å bli mer populært, ettersom det er lettere å vedlikeholde og kan være kostnadsbesparende om det blir opprettet riktig. Geodata har utviklet et multi-tenant Kubernetes cluster som kjører i Amazon Web Services (AWS) som brukes til å drifte kundeløsningene deres. Denne rapporten forsøker å samle cloud-hosting industriens anbefalinger og "best practices" til hvordan sette opp og sikre et multi-tenant cluster for så å sammenligne det med arbeidet Geodata har gjort i sin plattform.

Abstract

Multi-tenancy is a concept that has become increasingly popular because it is easier to maintain and can be cost-effective if set up correctly. Geodata has developed a multi-tenant Kubernetes cluster that runs in Amazon Web Services (AWS) and is used to host their services. This report will attempt to collect the industries recommendations and "best practices" in regards to cloud-hosting and how to set up and secure a multi-tenant cluster, and then compare it to Geodata's platform.

Contents

Figures

Acronyms

Amazon EB Amazon EventBridge. 39

Amazon EBS Amazon Elastic Block Store. 40

Amazon EC2 Amazon Elastic Compute Cloud. 13

Amazon ECR Amazon Elastic Container Registry. 15, 39

Amazon IAM Amazon Identity and Access Management. 15, 36, 39

Amazon S3 Amazon Simple Storage Service. 15, 39, 40

Amazon VPC Amazon Virtual Private Cloud. 13

AWS Amazon Web Services. i, ii, v, 2, 6, 8, 13–15, 37, 57–59

CNCF Cloud Native Computing Foundation. 39

GDO Geodata Online. v, 1–3, 6, 9, 12, 35, 56, 59

NTNU Norwegian University of Science and Technology. 3, 4

PSP Pod Security Policy. 24, 57

RBAC Role-based access control. 20, 36

SaaS Software as a Service. 9, 33, 56

Glossary

data at rest Data stored in persistent storage (e.g. disk, tape, etc.).. 34

delegated creation The possibility to create and manipulate subnamespaces without cluster-level privileges.. 41

Deployment A Kubernetes object that allows defining a desired state for Pods. The Deployment will maintain the correct number of Pods, so long as it is possible.. vii, 38

DevOps A philosophy for combining software development and IT operations.. 25

kubeconfig kubeconfig is a file used by kubectl to configure access to a Kubernetes cluster. It can define such things as where to connect to the Kubernetes API and access authentication.. 41

kubectl A command line tool for interacting with Kubernetes clusters.. vii

mixed methods research design Mixed methods research design is a methodology design that combines quantitative and qualitative methods [1].. 4

policy inheritance In a hierarchical namespace structure, this is when subnamespaces are automatically given the policies of the parent namespace.. 41, 51

subnamespaces A namespace that is the child of another namespace.. vii, 41, 51

Chapter 1

Introduction

1.1 Background

Geodata is a company working with geographic data processing and custom map solutions for various customers in the Norwegian private and public sector.

One of their main products is their self developed platform, Geodata Online (GDO). It is used for hosting and operating custom solutions developed for their customers. GDO is built on the industry-standard application deployment platform Kubernetes.

All of Geodata's development teams host their solutions inside the same Kubernetes cluster. One of the main challenges with GDO and this setup is the separation and segregation of the different teams, or *tenants*, in Kubernetes.

1.2 Scope

1.2.1 Subject Areas

The main focus of this project is researching multi-tenant Kubernetes clusters and the security between tenants. The thesis will focus on these subject areas:

- Container orchestration
- Software and infrastructure security
- Infrastructure as code
- Infrastructure operation

1.2.2 Task Description

Look at general solutions for multi-tenant services, consider how security can be effectively maintained in such an environment, compare our findings with Geodata's experiences and make recommendations based on this.

Project focus areas:

1. Overview of the platform and its components
2. Security analysis of Geodata's platform
    a. How is the security and separation between projects (tenants) achieved and maintained
    b. List of existing security measures
3. Look at how Geodata's platform is set up on Amazon Web Services (AWS), and replicate this setup ourselves (on a smaller scale)
4. Research best security practices for a generalized multi-tenant Kubernetes environment
5. Based on our findings, make recommendations for Geodata to consider

1.3 Limitations

The project has been limited to focusing exclusively on security inside the Kubernetes cluster. As with all security, it is only as strong as its weakest link. This means that the rest of the architecture must be secured as well - something that falls outside the scope of this thesis. In particular the underlying infrastructure that the cluster runs on must be secured from outside access, and limited only to the persons who require access. This is true whether the cluster is hosted in the cloud and online access must be regulated or if the cluster is on bare-metal and proper, physical, access control must be implemented. In Geodata's case this means securing the AWS infrastructure they use to host their cluster. This will not be discussed in this thesis, because of the aforementioned reasons. In a similar manner, container security is important to maintain integrity within a Kubernetes cluster. Specific measures related to container security also fall out of scope.

The research environment is a close replication of the GDO platform, but it is missing some components (some visualizations and inconsequential configurations), and it is only used by us - not hundreds of people working on different projects. It is not as easy as it seems to implement large-scale changes in the real world when multiple factors have to be considered.

1.4 Purpose

The purpose of this thesis is to research how to create and secure multi-tenant Kubernetes clusters. It is of high importance that safety is maintained when separate groups of people work inside the same environment. Furthermore, this thesis will address how Geodata have chosen to create and secure their own multi-tenant infrastructure and cluster. The recommendations given to Geodata in this thesis, as detailed in chapter 6, may be used to improve Geodata's current infrastructure and to help visualize the foundations needed for future iterations of GDO.

1.5 Target Group

The primary target group of this project is Geodata, who are hosting a multi-tenant cluster, and wish to make sure it is adequately secured. They also wish to know which measures should be considered before developing a newer version of GDO. Assessment and analysis of their implementations can be found in chapter 4, 5 and 6. Other audiences who are looking into creating a multi-tenant cluster, may be interested in reading chapter 2 and 3, which detail what measures to consider before moving to a multi-tenant infrastructure.

1.6 The Groups Background

All members of the group study IT-Operations and Information Security at Norwegian University of Science and Technology (NTNU) Gjøvik. Throughout these studies the group has acquired experience within several fields that are relevant to this report. The group has had courses that relate to, amongst others:

- Infrastructure as Code
- Service Architecture Operations
- Containerization
- Linux
- Programming
- Software security

The course has not provided any significant experience with Kubernetes itself, but the group has used other container orchestration technologies. No group member has used or navigated a multi-tenant cluster before working on this thesis, and as such, this thesis will require familiarizing ourselves with the technologies and infrastructures in use by Geodata.

1.7 Roles

The three organizations involved in this project and their roles are:

- Geodata:
  - Joachim Eckbo Juell - Director for product development at Geodata and thesis contact person.
  - Pål Kristensen - Department manager System operations at Geodata and thesis liaison.
  - Anders Østhus - Senior developer at Geodata and thesis liaison.
- Students:
  - Simen Asplund Kjeserud - Substitute project leader
  - Vemund Rahm - Project leader
  - Sonja Yoosuk Sanden - Group member
  - Eirik Tobiassen - Group member
- NTNU:
  - Erik Hjelmås - Thesis supervisor.

1.8 Methodology

The research section of this project will be based on various online resources on multi-tenant clusters. Alongside this, data will be collected by analyzing a Kubernetes cluster functionally identical to the one Geodata runs in their own production and development environment. These two will then be compared.

1.8.1 Research Methods

This dissertation will use mixed methods research design, where both qualitative and quantitative methods are used to gather data. Some examples of data collection methods for this thesis include interviews, sandbox experimentation, and published articles.

1.8.2 Types of Data

- Geodata's own experiences and setup.
  - What experiences Geodata has had when dealing with multi-tenant clusters.
  - Experimenting with a sandbox setup based on their design.
- Websites, articles and research papers concerning best practices surrounding multi tenancy and its security.
- Websites and articles about experiences others have had with multi tenancy and its security.

- News articles about breaches concerning multi tenant setups and Kubernetes.

1.8.3 How the Data is Used

The collected data will be used for:

- Understanding the security challenges surrounding multi-tenancy hosting in Kubernetes
- Understanding the difficulties of operating a multi-tenant Kubernetes cluster
- Finding common challenges in the design, development, and deployment of a multi-tenant Kubernetes cluster.
- Identifying which measures are most effective for operating a healthy and secure multi-tenant infrastructure.
- Identifying common mistakes when operating a multi-tenant Kubernetes infrastructure.

1.9 Structure

This thesis will have some of the main content separated out into appendices due to confidentiality. These appendices will be referenced in the report and removed from the published version.

The report has the following structure:
Chapter 1: Introduction - An introduction to the project including a task description and methodology.
Chapter 2: Background Theory - A background on the technologies that are used in the project.
Chapter 3: Secure Multi-Tenancy - A list of best practices, including their advantages and disadvantages.
Chapter 4: Implementation Analysis - A list of Geodata's security measures.
Chapter 5: Implementation Assessment - Assessment of Geodata's security measures.
Chapter 6: Recommended Changes - Recommended changes and why they should be implemented.
Chapter 7: Conclusion - Discussion of results, further research and conclusion.

Chapter 2

Background Theory

This chapter will explain the basic theory behind the technologies used in GDO. It will also include explanations given by Geodata as to why they made the decisions they did while designing and developing the platform. The end of the chapter introduces the parts of Geodata's infrastructure that are relevant to the GDO platform. Describing the entire Amazon Web Services (AWS) infrastructure falls outside of the scope of this thesis.

2.1 Amazon Web Services

2.1.1 What is Amazon Web Services?

Amazon Web Services is a cloud computing platform that charges customers on a pay-as-you-go basis. Using a cloud platform is often preferable to purchasing and maintaining your own physical hardware due to high operational and maintenance cost, at least in the first few years of operation [2]. Using a cloud provider also allow for rapid scalability in situations where system load may vary. AWS also provides an interface to deploy and scale infrastructures.

2.1.2 Background

In 2008 when Geodata decided to move their services to the cloud, AWS was the natural choice as Google Cloud Platform had just been released [3] and Microsoft Azure did not exist at the time [4]. From 2008-2016 Geodata built their online platform directly on top of AWS, and in 2016 they decided to make their services less dependent on AWS by using Kubernetes, which allows for abstraction between the applications and the underlying hardware. This can make moving from one cloud provider to another easier. Running Kubernetes, Geodata was also able to adapt their infrastructure to enable use of the Spot [5] marketplace for unused capacity in AWS Cloud. Doing this resulted in total EC2 instance expenditures going down by 70%.

2.2 Kubernetes

2.2.1 What is Kubernetes?

Kubernetes can be used for many purposes, but at its essence it is a way to abstract away an entire underlying infrastructure of complicated inter-connecting components and their resources, and expose them as a single unit that can be utilized by running applications [6]. Because Kubernetes manages the underlying infrastructure, the job of developing and deploying an application or service is simplified.

Kubernetes uses containerization technologies to realize this task. This means each computer process related to the application runs in its own ecosystem, oblivious to everything else that is running on a particular physical machine. When an app is deployed to Kubernetes, the deployment team never decides which specific machine should run the process, only how many replications of the app is needed along with other specifying data, and Kubernetes manages the rest [7]. Adding machines to the Kubernetes system only represents additional computational resources, it does not, and should not, mean a developer needs to think about how to redistribute the application to utilize resources in the most efficient manner.

Throughout this thesis, the terms *resource* and *object* are used interchangeably. In the Kubernetes documentation these terms have specific meaning, but this falls outside the scope of this thesis. For this reason, any time these terms are in use, they should be interpreted according to their context. In this thesis, *resource* and *objects* are used any time some definition that can be posted to the Kubernetes API is being discussed, or when something is a part of the Kubernetes ecosystem in general.

2.2.2 Objects

Describing how Kubernetes works falls outside the scope of this thesis, however this section will describe some of the resources most commonly referred to in this thesis [8, p. 5-10].

Kubernetes control plane:
The Kubernetes control plane (sometimes called master) is the global decision maker in Kubernetes, consisting of several components, such as the scheduler and the API server. It is the part of the cluster that everything else communicates with, and is responsible for actions such as restarting Pods when they fail.

Pods:
A Pod is a collection of co-related containers that represent one "process" or "task". Often this can be just one container, but if a task is represented by more than one container, they are grouped together into a Pod. The Pod is the smallest deployable unit in Kubernetes, it is not possible to deploy a container directly.

Nodes:
A Node represents some amount of computational resources that can be consumed by other Kubernetes Resources. It can be either a virtual or physical machine, depending on the underlying infrastructure. A Node needs to be running certain components to be qualified as a functional Node, including a container runtime.

Service:
A service exposes a group of Pods that provide an application at one consistent, reachable point, even though Pods are, by nature, ephemeral. Usually this is used so that a group of Pods performing one task can reach another group of Pods providing some necessary functionality, without the need of a service discovery application. Services can also be used to expose an application to an external IP, i.e. the Internet.

Namespaces
Namespaces segregate a cluster into multiple smaller, virtual clusters. Most other resources (e.g. Pods and Services), created in Kubernetes live inside a namespace, though some are global (e.g. Nodes).

2.2.3 Why Kubernetes is Right for Geodata

Background

In 2016, Geodata realized the large amount of data and services used to host customer solutions in AWS required them to optimize their resource usage. They also had a need for ways to make small and quick changes to their products, especially the deployment of services. This led to Geodata researching whether Kubernetes would be a good solution for these rising issues.

Geodata started building the first version of the Geodata Online platform that ran on Kubernetes in 2017, and around the summer of 2018 the development of the second generation of the platform was started. This second iteration started seeing use in 2019. It is the basis for the system that is still in use today, and is being continuously developed. Geodata are looking at developing and moving to a third generation of their platform.

What Kubernetes Brings to GDO, According to Geodata:

- Optimization of server resource utilization
- Better self-healing and automated solutions (*Infrastructure as code*)
- Abstract away the underlying infrastructure
- Methods, tools and programs the development-teams create for the Geodata Online platform can be reused, with only small changes needed for customers who want the system to run on their own Kubernetes clusters
- The possibility of moving to, or expanding the platform to Microsoft Azure without making large changes to solutions/services.

2.2.4 Multi-tenancy

Tenants

When creating multiple workloads in Kubernetes, it can be difficult to separate out the work into different logical sections. This logical separation is commonly created in a way where each workload or team is called a *tenant*.

A tenant can be defined in multiple different ways, but several leading cloud computing providers separate tenants into one of two categories: Enterprise tenancy (soft multi-tenancy) or Software as a Service (SaaS) tenancy (hard multi-tenancy) [9, 10].

In enterprise tenancy the tenants are separate teams inside an organization. These tenants are assumed to be relatively non-hostile, and the main concerns with multi-tenancy therefore becomes fair resource sharing.

In a SaaS tenant situation, it can not be guaranteed that all end-users are non-adversarial. Some services may be facing the public Internet, or the end-users will be in organizations outside the cluster administrators control. In such a scenario, secure segregation of tenants must become a main focus, alongside resource sharing. It is necessary that all tenants are properly separated, so that one tenant can not interfere with another.

Geodata operate a cluster that lay somewhere in between these two definitions. Their development teams are the tenants in the cluster, but several of their developed solutions face the public Internet as well. In future iterations of the GDO platform, Geodata plan to move further toward a SaaS platform.

Managing tenants

Technically separating the tenants in a manner that is clear and secure is a large challenge when hosting a multi-tenant cluster. One natural solution is to host one cluster per tenant, but this can have unnecessary overhead, because even the smallest of projects would require at least one or two Nodes all to itself. It can also be difficult for the cluster administrators to manage, since there is no centralized place to view all the current running resources. Figure 2.1 shows how a cluster administrator has to manage multiple clusters.

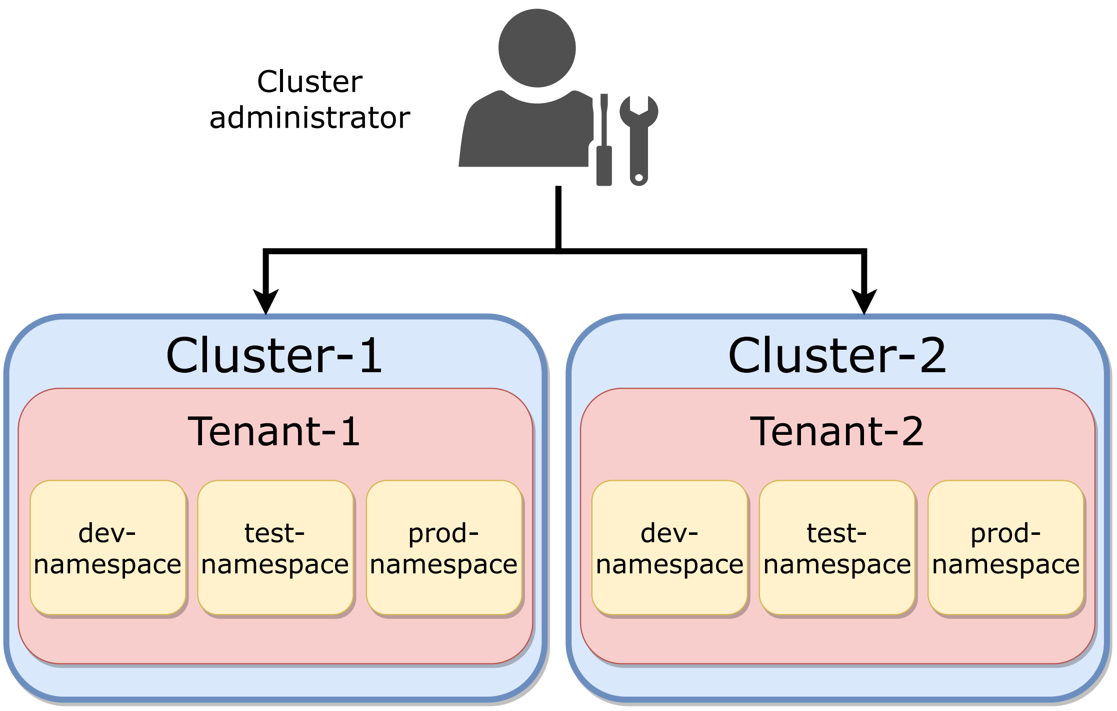


Figure 2.1: A visualization of one cluster per tenant. Here, separation of tenants is implied, since they have their own separate cluster.

Running a cluster in this manner can inhibit the resource inefficiencies Kubernetes and container technologies hope to rectify. This is why, in recent years, operating multi-tenant clusters has become significantly more popular. This is where several tenants all live inside the same Kubernetes cluster, running their containers on the same physical machines. Figure 2.2 shows this scenario. The proper separation of tenants living inside the same cluster in this manner is the main focus of the research part of this thesis.

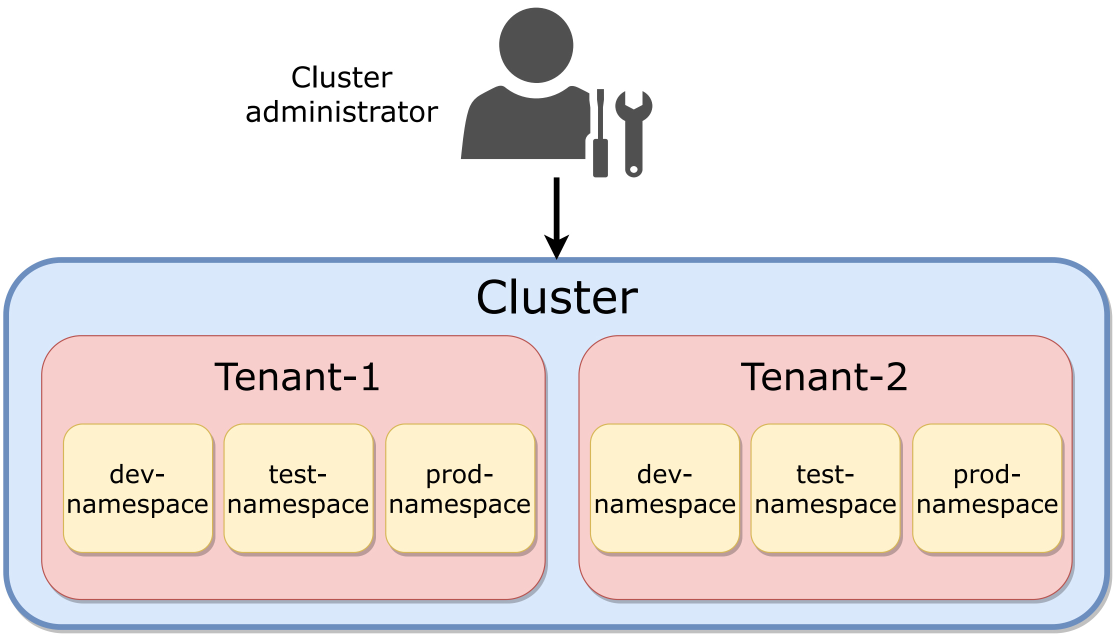


Figure 2.2: A multi-tenant Kubernetes cluster. In this scenario, measures must be taken to ensure that tenants are separated.

2.3 Architecture

2.3.1 Infrastructure

Geodata have a large, varied and modern infrastructure. A full overview of all their components can be seen in figure 2.3. This thesis will focus on the components in the *Orchestration* section and some parts of the *Storage* section. Any components higher up than the *Orchestration* level in the infrastructure fall outside the scope of this thesis.

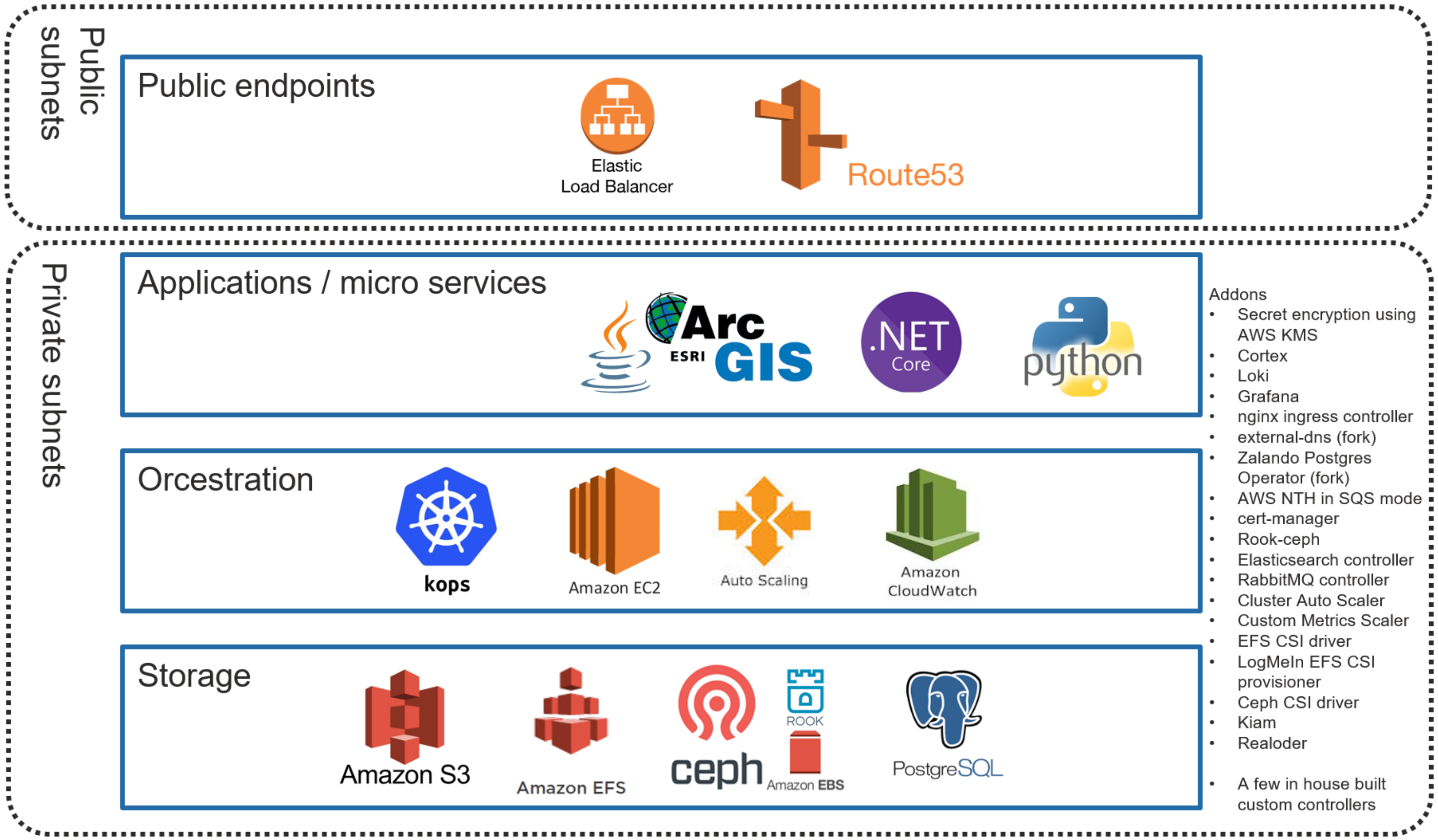


Figure 2.3: GDO Infrastructure Map.
Source: Provided by Geodata

A deeper view into how the Amazon VPC private and public subnets communicate can be seen in figure 2.4. This figure reflects how Geodata have set up their infrastructure. All of Geodata's Amazon EC2 machines run in the private subnets.

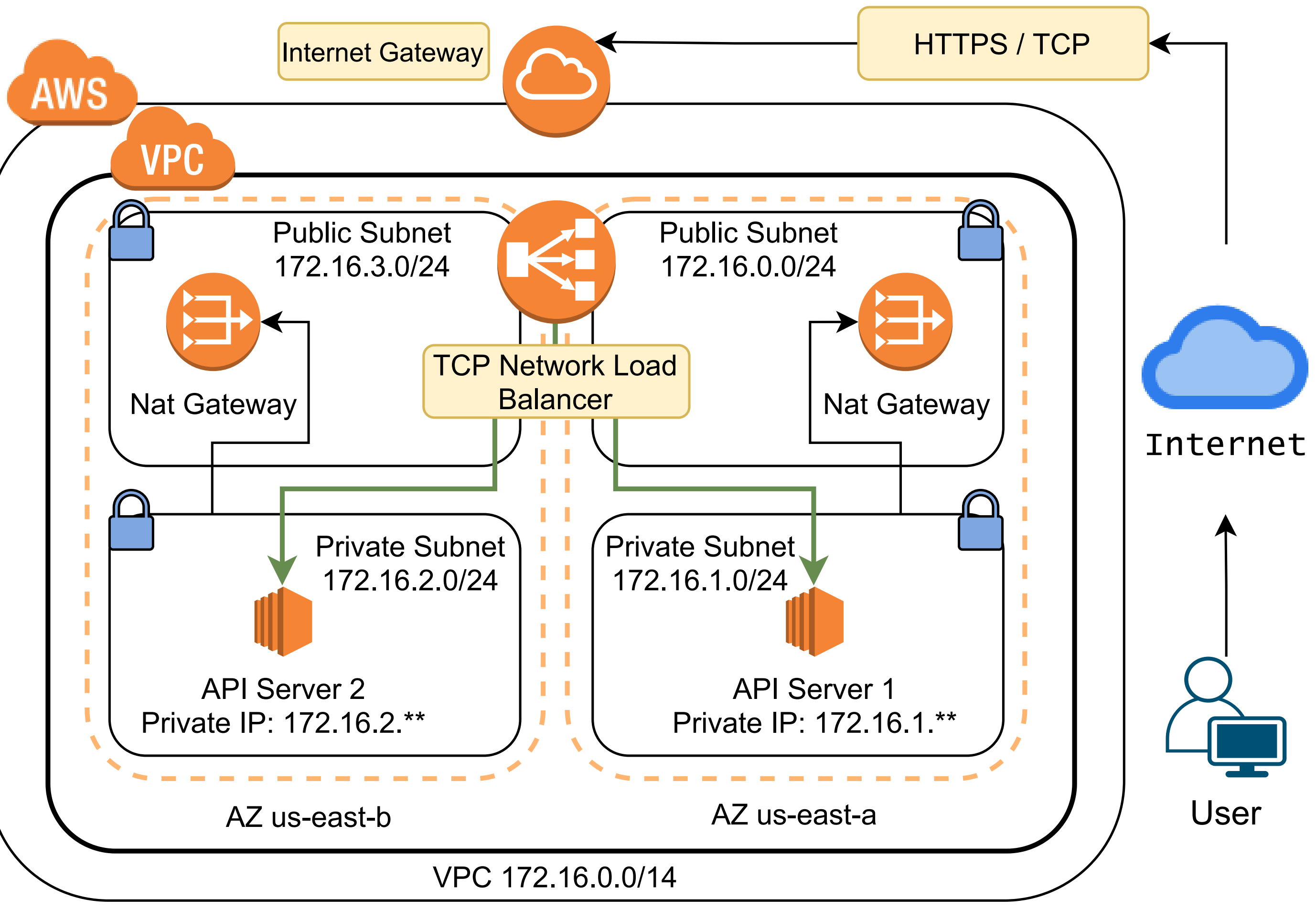


Figure 2.4: AWS Infrastructure Map.
Source: Based on a slide from a Geodata PowerPoint (Figure C.1)

2.3.2 Kubernetes Addons

Geodata use several addons on top of Kubernetes to make sure the platform has all the functionality required, some of which is not provided by Kubernetes natively. These addons are responsible for both simplifying deployment and for keeping the system healthy following the "architecture as code" principle. The addons are also responsible for making and showing analytics, helping with dynamic storage, keeping resources to a minimum, security, and more.

These are the thesis-relevant addons in use by Geodata (Other addons in use by Geodata can be found in Appendix B):

Grafana

Grafana is an open source visualization and analytics software. It is used for querying and visualizing metrics collected by other processes like Prometheus.

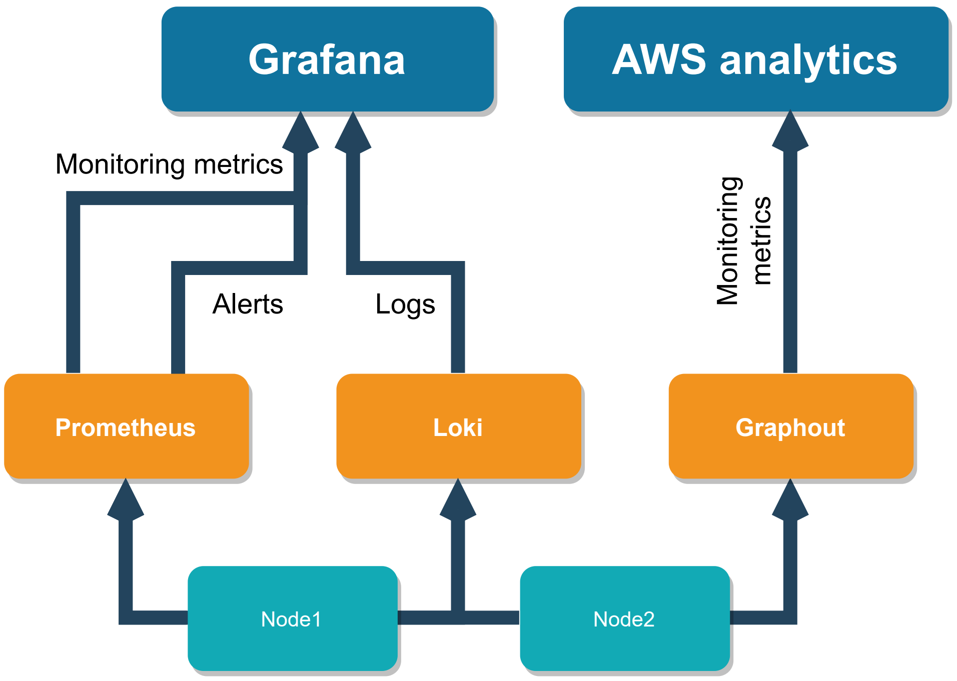


Figure 2.5: A visualization of the relationship between Grafana, Prometheus, Loki and Graphout.

Graphout

This is a component that can query metrics made by Prometheus and similar tools and forward it to external services such as Amazon analytics. The data can be used for auto scaling in AWS.

Kiam

Kiam is used for giving Amazon IAM roles to Pods. Amazon has their own official version, but Geodata use Kiam. Kiam runs as an agent on each Node in a Kubernetes cluster. Using kiam, one can access AWS services from Pods running in Kubernetes without putting AWS secret keys in the code [11]. This follows the security principle of least privilege.

Loki

Loki is a type of logging monitor which can be used with Grafana, to make it possible to show logs for a specific time period. This is very useful if, for example, some servers lose connection and you want to see the logs for the period of time the servers were down. This makes it possible to get a better view over an incident or if you want to see what caused a spike of resource usage.

Prometheus

Prometheus is a tool used for performing unit testing and collecting monitoring data about health, uptime, etc, from each Node in the infrastructure.

Prometheus has a ’single binary’ installed alongside with it, called Alertmanager. Alertmanager handles alerts sent by client applications such as the Prometheus server.

Rook.io (Ceph)

Rook is a storage solution which sets up a storage cluster using Ceph and provides redundant storage.

This means it can, for example, automatically repair itself if one of the engines is gone. In other words, it is a form of storage solution with automatic healing capabilities.

Reloader

The Reloader is a small controller that runs in the cluster and is responsible for restarting the Pods whenever a new update is made to the config map to make sure they are always using the latest configuration available.

Project-controller

An addon written by Geodata which controls the creation of Amazon S3 buckets, Amazon ECR images and, Amazon IAM users and roles. It can also discover these resources if they already exist. This addon is triggered whenever someone creates an object in a namespace.

Chapter 3

Secure Multi-Tenancy

This chapter will describe the details, best practices and recommendations that must be considered when hosting a multi-tenant cluster. The best practices are based on written guides, articles and previous attacks. Sources are linked in each best practice. Geodata's cluster has not been considered for this part of the thesis.

3.1 A Secure Cluster

If multiple tenants reside inside the same cluster without proper separation, a security breach stemming from one tenant can propagate to the entire cluster [12, 13]. In cases where tenants are not from the same organization, this can be especially dangerous.

By default, a Kubernetes cluster has essentially no separation measures for tenants, considering that any Pods or resources deployed:

- Are on a flat network, if following the Kubernetes networking model [14].
- As such, can hook onto the same services and see and communicate with each other.

To secure and effectivize a multi-tenant cluster, measures must be taken to control access to the cluster, and separate tenants from each other. Figure 3.1 shows the key parts of this process.

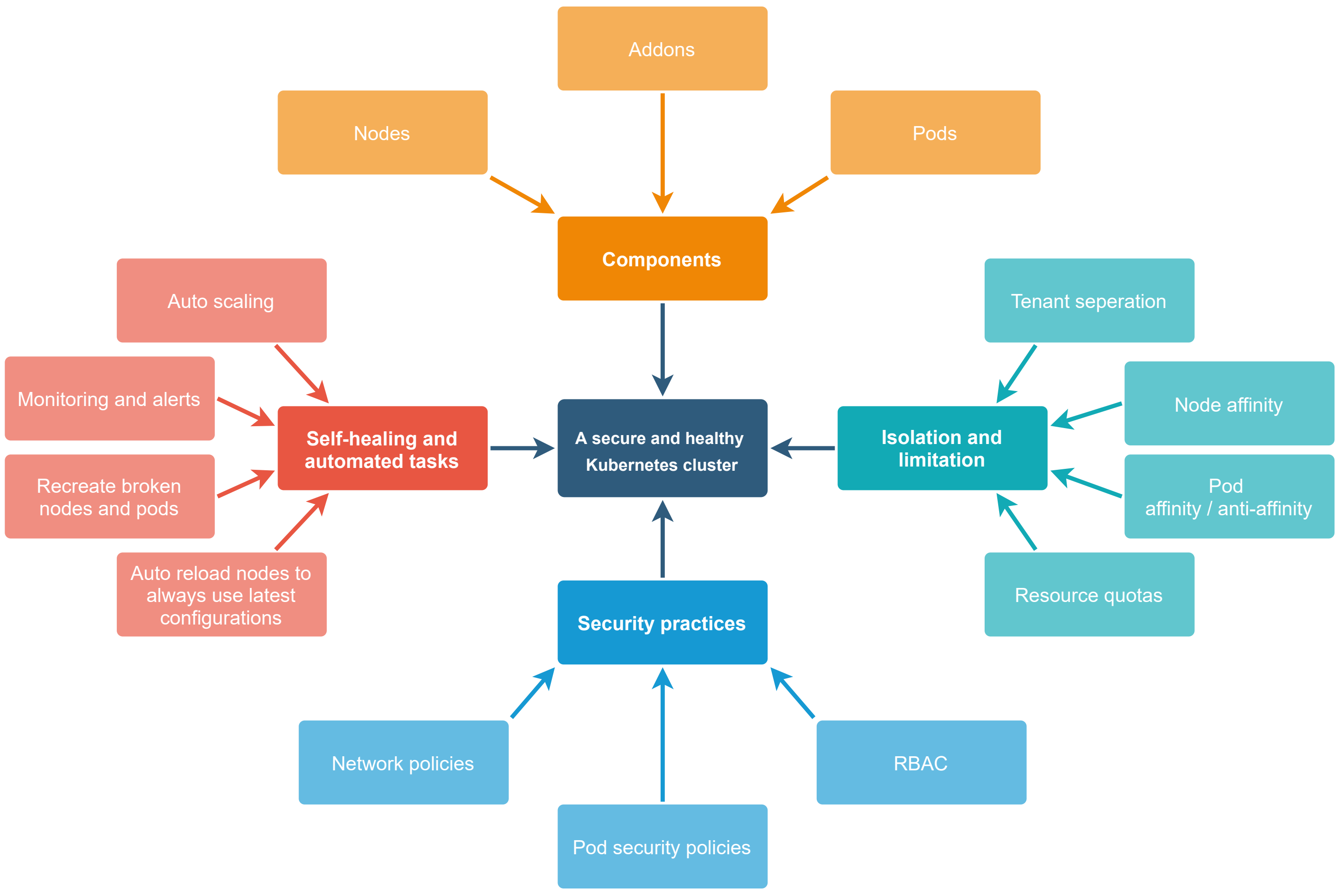


Figure 3.1: A secure and healthy Kubernetes cluster. This visualization is based on the best practices in chapter 3.2.

The core unit that enables most of the separation in Kubernetes is namespaces. A namespace is a logical virtual cluster inside the main cluster. Other measures such as policies are enabled on a per-namespace basis. Each tenant should have their own namespace, which should be isolated to the degree required by the multi-tenant implementation (hard or soft). If using a hard implementation, each tenant should, at the very least, be unable to see other tenants and have their own storage and network planes. Furthermore, no matter which implementation is used, access to the Kubernetes API should be on a "need-to-use" basis, in that each tenant must have access to read and modify resources only in their own namespace, and access to the cluster control plane must be given only to the cluster administrators, not the tenants.

3.2 Findings

3.2.1 List of Best Practices

T-1: Namespaces
T-2: Role-Based Access Control (RBAC)
T-3: Affinity
P-1: Resource Quotas
P-2: Network Policies
P-3: Pod Security Policies (PSP)
ML-1: Monitoring and alerts
ML-2: Per-tenant monitoring
DM-1: Automatically update Pods with the latest configurations
DM-2: Dependency tracker
DM-3: Set up consistent time windows for maintenance
DM-4: Continuous active vulnerability scanning
DM-5: Container security
DM-6: Only use trusted code in the infrastructure
S-1: Storage segregation
S-2: Data encryption

3.2.2 Tenant separation (T)

T-1: Namespaces

Description:
Every tenant should have a separate namespace, which gives them access to a logical partition inside the cluster. In addition, all namespaces should use a hierarchical structure [15].

Reasoning:
Namespaces or virtual clusters can be used to split a cluster into logical partitions. By giving each tenant their own namespace it is possible to isolate resources that are spread across teams and projects, isolate tenants from each other and set up different policies and constraints for each tenant; E.g.: Resource quotas, network policies, etc.

Advantages

- Logical separation of tenants.
- Each tenant have their own logs that can be used to troubleshoot.
- Resources created in one namespace are not visible from other namespaces

Disadvantages

- Requires consistent naming convention.
- Requires further separation to ensure security and mitigate risk of data leakage.

Source:

https://aws.github.io/aws-eks-best-practices/security/docs/multitenancy/#namespaces

https://kubernetes.io/blog/2020/08/14/introducing-hierarchical-namespaces/

T-2: Role-Based Access Control (RBAC)

Description:
Restrict system and resource access to only authorized users by setting up RBAC.

Reasoning:
By using RBAC one can give a user a specific set of permissions within a namespace. RBAC can be used not only within individual namespaces as an RBAC Role, but also across all namespaces in a cluster as an RBAC ClusterRole. Further one can use RoleBinding or ClusterRoleBinding to grant a role to users, where RoleBinding grants permissions within a namespace and ClusterRoleBinding grants cluster-wide permissions.

Advantages

- Enables user separation.
- Easier management of user access.

Disadvantages

- Easy to misuse if grants are given unnecessarily.
- Inactive users can be used as an attack vector.

Source: https://aws.github.io/aws-eks-best-practices/security/docs/multitenancy/#role-based-access-control-rbac

T-3: Affinity

Description:
Regulate Pods so they can only run on certain Nodes with Pod affinity and Pod anti-affinity.

Reasoning:
Affinity can be used to determine where the Kubernetes scheduler schedules Pods. In particular, Pod anti-affinity can be used to make sure two Pods from different namespaces never run on the same Node, using Pod labels.

Advantages

- Increases security between tenants.
- Greatly expands the types of constraints you can express. [16]

Disadvantages

- Can be abused by adding or changing a label.
- Anti-affinity alone can not enforce security policies.
- Can lead to resource-usage penalties.

Source: https://cloud.google.com/kubernetes-engine/docs/concepts/multitenancy-overview#pod_anti-affinity

3.2.3 Policies (P)

P-1: Resource Quotas

Description:
Set a resource quota to ensure that all tenants have access to a fair amount of resources and avoid disproportionate resource usage across tenants.

Reasoning:
In a multi-tenant environment where all tenants share resources, a resource quota can be used to avoid resource starvation, by defining the minimal and maximal resources each tenant can have access to. This can also increase security, because if an adversarial agent gets access to the API, they can not starve the other tenants of all resources. Specific quotas will depend on the use-case.

Advantages

- Prevents resource starvation.
- Better stability and performance.

Disadvantages

- Requires calculation of minimal needed resource for each tenant.

Source: https://aws.github.io/aws-eks-best-practices/security/docs/multitenancy/#quotas

P-2: Network Policies

Description:
Create policies for restricting Pod communication within a namespace.

Reasoning:
Network policies are used as a basis for restricting network communication between Pods in different namespaces. It is not enough to have network policies to isolate Pods and therefore tenants in a cluster, one would also need a policy engine to enforce network policies.

Advantages

- Control ingress/egress connectivity between Pods.
- Mitigates leakage of data between Pods.
- Ensures that only Pods who are allowed to communicate with each other are able to.

Disadvantages

- Requires a policy engine.
- Misconfiguration can hinder Pod communication.

Source: https://aws.github.io/aws-eks-best-practices/security/docs/multitenancy/#network-policies

P-3: Pod Security Policy (PSP)

Description:
PSP should be added to restrict actions and privileges of Pods.

Reasoning:
Pod Security Policy (PSP) gives the opportunity to define a set of conditions for creating and updating Pods. If a Pod does not run with the given conditions, it will not be accepted into the system.

Advantages

- Better control over Pod privileges.
- Stops misconfigured Pods from being accepted into the system.

Disadvantages

- Misconfiguration of the policies can prevent Pods that are correctly defined from being created.

Source: https://kubernetes.io/docs/concepts/policy/pod-security-policy /#what-is-a-pod-security-policy

3.2.4 Monitoring and Logging (ML)

ML-1: Monitoring and alerts

Description:
Set up monitoring of the cluster and make use of automatic alerts to notify the operational team of service irregularities in real time.

Reasoning:
Enabling monitoring and automatic alerts enables the DevOps team to stay ahead of potential operational incidents through early detection and mitigation. Having a dashboard to analyze metrics enables the team to identify symptoms of issues rather than simply monitoring failure states.

Advantages

- Quicker response to incidents.
- Less downtime.

Disadvantages

- False positives may lead to waste of resources.
- Can create a false sense of security. Could potentially lead to oversights if the monitoring is not well implemented or the alarm system is misconfigured.
- Requires monitoring services such as Grafana or Prometheus.

Source: https://sysdig.com/blog/alerting-kubernetes/

ML-2: Per-tenant monitoring

Description:
When monitoring systems and resources, have one separate section for each tenant.

Reasoning:
Monitoring on a per-tenant basis makes it clearer where certain issues may be arising, such as unfair distribution of resources. It also makes it significantly simpler for each development team to check their own statistics.

Advantages	Disadvantages
• Easier for tenants to see their own resource usage. • Simpler to troubleshoot a specific tenant.	• Cost of deploying multiple monitoring services.

Source: https://cloud.google.com/kubernetes-engine/docs/best-practices/enterprise-multitenancy#tenant-monitoring

3.2.5 Development and Maintenance (DM)

DM-1: Automatically update Pods with the latest configuration

Description: To ensure all Pods are always up to date and using the latest configuration, make sure Pods are automatically updated whenever changes to the configuration is made.

Reasoning: Making changes to the configuration files used by Kubernetes for each Pod will not automatically update the Pod. To make sure that Pods are always using the correct configuration, implement a system to automatically update running Pods. Pods using old configurations can cause instability in the infrastructure and security risks.

Advantages

- No need for manual Node updates when there is a configuration update.
- Improves security by always having the latest version.

Disadvantages

- Configuration might have issues that can cause Node failure.

Source:
https://medium.com/@damien.marshall/using-reloader-for-configuration-rollout-in-kubernetes-e0e988077dab

https://medium.com/stakater/stakater-reloader-4c985ac23e3b

DM-2: Dependency tracker

Description:
Set up a system that keeps track of all dependencies in the infrastructure.

Reasoning:
Keeping track of dependencies in the infrastructure can provide valuable insight into possible vulnerabilities and risks. The data from the dependency tracker can be used to make periodic re-evaluations of dependencies that the system is reliant upon. Dependencies that have known vulnerabilities, or which are no longer supported can be properly tracked and addressed.

Advantages

- Better control over which dependencies needs to be updated.
- Better control over which dependencies are used in the system.

Disadvantages

- If other systems that track vulnerabilities are in place, additional systems may be excessive, and may contribute to data overload

Source: https://www.bbva.com/en/vulnerability-management-in-dependencies-in-ci-cd-environments-with-open-source-tools/

DM-3: Set up consistent time windows for maintenance

Description: Decide on a time period where maintenance and updates to the infrastructure should be done.

Reasoning: By having a set time period for when maintenance and updates should be applied to the infrastructure, it is possible to keep the impact of a failed update relatively small. Having a consistent time window for updates can also help identify security breaches, e.g. if a significant change to the infrastructure is detected outside a maintenance time window.

Advantages

- Members of the organization can prepare for downtime.
- Updates can be scheduled when someone is available to respond, should an unforeseen incident occur.
- Easier to spot unusual changes to the infrastructure.

Disadvantages

- Small updates can not be sent out immediately when they are ready.
- Large updates can, in some cases, make it more difficult to discover which part of the update is at fault if something breaks.

Source: https://www.pktech.net/2019/01/picking-the-right-approach-to-it-maintenance-windows/

DM-4: Continuous active vulnerability scanning

Description: Monitor running services for open vulnerabilities by passively scanning the system.

Reasoning: In addition to keeping track of vulnerable dependencies and images, actively scanning Pods for known vulnerabilities is a good way to ensure patches are working as intended, policies are properly followed, and quickly identify any accidental configuration changes before potential vulnerabilities are exploited by third parties.

Advantages

- Potential discovery of vulnerabilities before they can be exploited.
- Patches can be applied before anyone can fully exploit the vulnerability.

Disadvantages

- False sense of security if the scanning does not reveal smaller vulnerabilities
- Running the software costs resources and may have additional licensing costs.

Source:
https://www.stackrox.com/post/2020/05/kubernetes-security-101/

DM-5: Container security

Description:
Follow best practices specific to container images.

Reasoning:
Maintain security at the container level to restrict access to the physical Nodes Pods are running on, thereby making sure no unauthorized API access can be attempted. It is recommended to follow best practices specifically made for container images. Which specific best practices to follow are outside the scope of this thesis.

Advantages	Disadvantages
• Prevent unauthorized access. • Secure containers and their content.	• Requires resources to both find best practices and cross reference with each container image • Hosting a Container Registry might be needed, which will cost resources.

Source: https://thenewstack.io/container-security-multitenant/

DM-6: Only use trusted code in the infrastructure

Description:
Before using a program, addons and other types of code in the infrastructure, evaluate whether the developers behind the code or the code itself can be trusted.

Reasoning:
Any code added to the infrastructure can directly impact its performance and security. For that reason it is imperative that anything that gets added does not pose any danger to the security and are not malicious. Malicious code may have threats such as backdoors, viruses, ransomware or simply not function as they should. The only way one can be somewhat sure code is safe to use is by looking at the source code. Evaluating if the developers behind the code itself are trustworthy is also important, since malicious code may be hidden or the developers may push an update with malicious code at a later date. An example of this was when the research team at Aqua Security found 23 container images with Potentially Unwanted Application (PUA) hidden either within their image layers or downloaded into their instantiated containers during runtime [17].

Advantages

- More control over what is running in the infrastructure.
- Less likelihood of having malicious code in the infrastructure.

Disadvantages

- Using any new third party code requires manpower to do the evaluation.
- Developers will not be able to use code made by unknown developers or companies unless the code is open source.

Source: https://www.whitesourcesoftware.com/resources/blog/kubernetes-security/

3.2.6 Storage (S)

S-1: Storage segregation

Description:
Ensure tenant storage spaces are properly segregated to avoid information leakage

Reasoning:
By properly segregating tenants so they can not access another tenants data, the security of a multi-tenant cluster is greatly increased. This is especially important when working with a SaaS infrastructure, where tenants should be completely isolated.

Advantages	**Disadvantages**
• Each tenant can only access their own data.	

Source: https://d0.awsstatic.com/whitepapers/Multi_Tenant_SaaS_Storage_Strategies.pdf

S-2: Data encryption

Description:
Encryption of customer data at rest.

Reasoning:
All customer data at rest should be encrypted. This provides increased data protection in case of security incidents, preventing data leakage from customers. Encrypting data at rest also provides a security mechanism to limit the exposure in a lateral attack including previously unknown threats, so long as the actor also has access to the key management system.

Advantages

- Ensures security of customer data in case of a data breach
- Provides an extra layer of privacy by protecting sensitive data

Disadvantages

- Requires resources to encrypt and decrypt data
- Requires an encryption key management system to manage the encryption keys
- If encryption keys are lost, the data may also be permanently lost unless there are backups of the keys.

Source:
https://aws.github.io/aws-eks-best-practices/security/docs/data/

Chapter 4

Implementation Analysis

This chapter will go through the actions Geodata has taken to securely and effectively enable multi-tenancy in their cluster. These actions are directly mapped to our findings in chapter 3. The data gathered is based on workshops and meetings with Geodata along with source code analysis from our simulated GDO cluster.

4.1 Geodata Security Measures

4.1.1 List of Security Measures

Geo-T-1: Namespaces
Geo-T-2: Role-Based Access Control (RBAC)
Geo-P-1: Resource Quotas
Geo-ML-1: Monitoring and alerts
Geo-ML-2: Per-tenant monitoring
Geo-DM-1: Automatically update Pods with the latest configuration
Geo-DM-2: Maintenance policy
Geo-DM-3: Image Vulnerability Scanning
Geo-DM-4: Only use trusted code
Geo-S-1: Storage segregation
Geo-S-2: Data encryption

4.1.2 Tenant Separation (Geo-T)

Geo-T-1: Namespaces

Description:

Geodata use namespaces to separate tenants and their resources.

How it is implemented:

Geodata have one separate namespace for each of their tenants (prefixed gdonline-{projectname}) and any addons that requires running Pods.

Geo-T-2: Role-based access control (RBAC)

Description:

Geodata uses role-based access control to regulate access to create and view resources.

How it is implemented:

Geodata have multiple roles in each of the namespaces, e.g.:

1. An operator role to manage resources
2. A role to manage addons and service accounts.

Geodata have implemented RBAC by using both Kubernetes RBAC and Amazon IAM, where Kiam is used to connect Amazon IAM roles with Kubernetes roles. RBAC users are made by the project-controller addon.

4.1.3 Policies (Geo-P)

Geo-P-1: Resource Quotas

Description:

Resource Quotas are used by Geodata to ensure that all tenants always have access to resources.

How it is implemented:

Geodata have specified how many requests on CPU, memory and local ephemeral storage Pods can make.

4.1.4 Monitoring and Logging (Geo-ML)

Geo-ML-1: Monitoring and alerts

Description:

Geodata uses a combination of addons to monitor the health and other metrics of their cluster.

How it is implemented:

To monitor their multi-tenant cluster, Geodata uses a collection of different addons.

Geodata use Prometheus to gather metrics, Grafana for visualization of the data, while Graphout is used to send the data to AWS. Loki is used for log aggregation, and Alertmanager is used to manage alerts.

Prometheus is self hosted by Geodata and the other addons are used on top or alongside Prometheus.

Geo-ML-2: Per-tenant monitoring

Description:

Geodata has monitoring on a per-tenant basis.

How it is implemented:

Geodata performs monitoring on a namespace basis, which can be viewed through Grafana.

4.1.5 Development and Maintenance (Geo-DM)

Geo-DM-1: Automatically update Pods with the latest configuration

Description:

Pods get updated whenever an update to the configurations are made.

How it is implemented:

Geodata use the reloader addon to automatically perform a manually specified update strategy for Pods (or more correctly, a Deployment) whenever an update is made to the configuration files, so all Pods are in accordance with the latest configuration changes. If no update strategy is specified, rolling update is used.

Geo-DM-2: Maintenance policy

Description:

A policy on how and when maintenance can be done.

How it is implemented:

Maintenance or updates are only done when there is a need for new versions or patches. Low risk updates are often done in work hours, while high risk updates are made during the evening, night, and weekends. Geodata require that no update should result in downtime of the system. If downtime is unavoidable, there will be a notice given internally at least a week before pushing the update.

Geo-DM-3: Image Vulnerability Scanning

Description:
Geodata have vulnerability scanning on their docker repository.

How it is implemented:
Every time Geodata push an image to their docker repository, Amazon Elastic Container Registry (Amazon ECR) will do a vulnerability scan of the image and send events to Amazon EventBridge [18].

Geo-DM-4: Only use trusted code

Description:
Geodata only use code from trusted sources and code they have reviewed.

How it is implemented:
Geodata have chosen to trust the Cloud Native Computing Foundation (CNCF) because they have strict policies when it comes to the conduct of their project maintainers [19]. Geodata also use code from other sources which they review and fork to their own project repository before using.

4.1.6 Storage (Geo-S)

Geo-S-1: Storage segregation

Description:

Geodata separate tenant storage to ensure security of information.

How it is implemented:

Amazon Simple Storage Service (Amazon S3) ensures by default that users only have access to resources they create. If a tenant has multiple users, access can be regulated using Amazon IAM. However, there are certain cases where tenant storage may not be completely segregated, see section 5.5.2.

Geo-S-2: Data encryption

Description:

Encryption of tenants data.

How it is implemented:

Geodata use Amazon S3 to store and encrypt their data. Amazon S3 store resources in buckets that get encrypted and Amazon Elastic Block Store (Amazon EBS) volumes makes sure that the Ceph storage is encrypted. Amazon S3 buckets are created by project-controller when a tenant creates a bucket object (CRD) in their namespace. The keys are renewed every 6 months.

Chapter 5

Implementation Assessment

This chapter is confidential, see Appendix_Bacheloroppgave_multi_tenancy.pdf in attachments.

Chapter 6

Recommended Changes

This chapter is confidential, see Appendix_Bacheloroppgave_multi_tenancy.pdf in attachments.

Chapter 7

Conclusion

7.1 Results

The thesis has broadly discussed what steps a cluster administrator should consider before allowing multiple tenants inside their Kubernetes cluster. Several different sources, mostly consisting of cloud providers, were used to determine a secure and effective way to host a multi-tenant Kubernetes cluster. Where applicable, the best practices are discussed in light of whether they are necessary when hosting an enterprise multi-tenant cluster, or a Software as a Service multi-tenant cluster. The results are listed in section 3.2, where each best practice is detailed with a reasoning, and advantages and disadvantages. The thesis subject area was inspired by Geodata's request to get an external opinion on how secure their multi-tenant cluster configuration is.

Geodata's request was also to consider what they could do to further improve security, especially for their next iteration of GDO. An assessment and analysis of their current infrastructure is located in chapter 4 and 5. The findings of this thesis have been that they have an infrastructure that is working, stable and secure given the context, but there are some improvements to the security measures that should be implemented before moving to a SaaS architecture. These recommendations are detailed in chapter 6, where each recommendation is discussed, with advantages and disadvantages of implementing the recommendation.

The findings and recommendations provided mainly consist of steps to further isolate tenants within the cluster, with the theoretical end-goal of completely separating tenants, to the degree possible with the technologies that currently exist. The thesis attempts to provide clear, concise steps for Geodata to consider for moving towards this goal. The recommendations are written with the intent to target the next version of GDO.

7.2 Further Research

Multi-tenancy in Kubernetes is a relatively new subject, but it is being adopted by companies all over the world who want to utilize the resource efficiencies of Kubernetes and the cloud, but who either have teams that work on fundamentally separate projects, or are providing a hosted Kubernetes cluster for others to use. Currently, scientific research into the security concerns that arise when hosting a multi-tenant setup are very limited. This was the inspiration for this thesis, however, this means there was very little previous research to base our recommendations on. Potential further research into the subject matter at hand could be:

- Consider all Kubernetes resources, such as secrets
- Perform a risk-to-return analysis of multi-tenancy.
- Further practical penetration testing, beyond the networking practical that was implemented in 5.2.1. E.g. attempting to abuse access to the Kubernetes API, or gain access to other tenants data.
- As mentioned in 6.3, Pod Security Policy is being deprecated. As of the publishing of this thesis, multiple solutions are being developed or are available, and these could be compared.

Further, creating an all-in-one multi-tenant addon to Kubernetes would be beneficial to all the future adaptors of multi-tenancy, but this would be an immense undertaking, and not all use-cases for multi-tenancy require the same guidelines and restrictions.

7.3 Project Review

During the project period, all meetings and project work has been done remotely because of Covid-19. The group cooperation has been adequate; We have had weekly meetings, following a scrum model, where workloads are divided and discussed based on what needs to be done before the next meeting. We had a weekly meeting with our supervisor, where we gave updates and received feedback on work that had been done the preceding week. This has allowed us to receive constructive feedback without bias. We also had bi-weekly meetings with Geodata to discuss progress and ask any questions that came up between meetings. During this project we got to receive hands on experience with a real-world infrastructure used in production. This has allowed us to learn a lot about multi-tenancy, Kubernetes, Amazon Web Services (AWS) and team work.

7.4 Review of Project Focus Areas

The goal of this project was to make improvement recommendations to Geodata based on general best-practices and Geodata's existing cluster. The project was divided into focus areas to break down the work that had to be done.

Project focus areas and how they were accomplished:

1. Overview of the platform and its components

    A brief overview of Geodata's platform and its components are described in chapter 2.3, where there is an infrastructure overview and description of addons. Geodata's platform also consist of many AWS components which are mentioned throughout this thesis. The platform components were found by researching Geodata's shared source code on GitHub and our workshops with Geodata.

2. Security analysis of Geodata's platform

    - How is the security and separation between projects (tenants) achieved and maintained
    - List of existing security measures

    Security measures by Geodata that correspond with best-practices are listed in chapter 4, and how Geodata secure and separate tenants is detailed in chapter 5, where we discuss measures that are implemented, but also measures that are missing in Geodata's infrastructure. Geodata's security measures were found by analysing the replicated cluster, source code on GitHub and meetings with Geodata. In the project plan there were also a focus area 2c, but this was removed as it was practically the same as focus area 5.

3. Look at how Geodata's platform is set up on AWS, and replicate this setup ourselves (on a smaller scale)

    To set up the replica of Geodata's cluster we had a couple of workshops with Geodata, where they showed us how things were set up and shared their source code with us through GitHub. These workshops were quite useful as Geodata not only showed us how they have set up their cluster, but they also described what each component was and did in their cluster.

4. Research best security practices for a generalized multi-tenant Kubernetes environment

   A list of all best-practices the group found is listed in chapter 3. All best-practices were found using reliable and trusted sources, such as AWS and Google Cloud. Many best-practices were found, of which Geodata has implemented most of them.

5. Based on our findings, make recommendations for Geodata to consider

   All recommendations that Geodata should consider implementing are mentioned in chapter 5 and further detailed in chapter 6, along with how they can implement some of the recommendations. All recommendations are based on which best-practices Geodata have not yet followed or implemented, with extra weighting being put on hard vs. soft multi-tenancy, where applicable.

All focus areas of this thesis are covered quite thoroughly. The group hope that the work we have done is useful to Geodata, and that they will consider implementing our recommendations in their next iterations of GDO.

Bibliography

[1] G. D. Caruth, 'Demystifying mixed methods research design: A review of the literature.,' *Online Submission*, vol. 3, no. 2, pp. 112–122, 2013.

[2] C. Fisher, 'Cloud versus on-premise computing,' *American Journal of Industrial and Busi-ness Management*, no. 8, pp. 1991–2006, 2018.

[3] *Google cloud computing*, [(Accessed 08/03/2021)]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Google_Cloud_Platform&oldid=1010715672.

[4] *Microsoft azure*, [(Accessed 08/03/2021)]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Microsoft_Azure&oldid=1009248064.

[5] Amazon, *Spot*, (Accessed 16/04/2021). [Online]. Available: https://aws.amazon.com/ec2/spot.

[6] Kubernetes team, *What is kubernetes?* [(Accessed 07/03/2021)]. [Online]. Available: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/.

[7] A. Verma, L. Pedrosa, M. R. Korupolu, D. Oppenheimer, E. Tune and J. Wilkes, 'Large-scale cluster management at Google with Borg,' in *Proceedings of the European Conference on Computer Systems (EuroSys)*, Bordeaux, France, 2015.

[8] G. Sayfan, *Mastering Kubernetes: Level up your container orchestration skills with Kubernetes to build, run, secure, and observe large-scale distributed apps, 3rd Edition*. Packt Publishing, 2020, ISBN: 9781839213083. [Online]. Available: https://books.google.no/books?id=ZK7uDwAAQBAJ.

[9] AliBaba cloud, *Practices of kubernetes multi-tenant clusters*, (Accessed 09/03/2021). [Online]. Available: https://www.alibabacloud.com/blog/practices-of-kubernetes-multi-tenant-clusters_596178.

[10] Google cloud, *Cluster multi-tenancy*, (Accessed 09/03/2021). [Online]. Available: https://cloud.google.com/kubernetes-engine/docs/concepts/multitenancy-overview.

[11] Alexander Brand, *Using kiam to access aws resources from kubernetes pods*, (Accessed 21/04/2021). [Online]. Available: https://alexbrand.dev/post/using-kiam-to-access-aws-resources-from-kubernetes-pods/.

[12] Z. Feng, B. Bai, B. Zhao and J. Su, 'Shrew attack in cloud data center networks,' in *2011 Seventh International Conference on Mobile Ad-hoc and Sensor Networks*, 2011, pp. 441–445. DOI: 10.1109/MSN.2011.71.

[13] T. Ristenpart, E. Tromer, H. Shacham and S. Savage, 'Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds,' in *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 199–212.

[14] Kubernetes team, *Cluster networking*, (Accessed 11/05/2021). [Online]. Available: https://kubernetes.io/docs/concepts/cluster-administration/networking/.

[15] Kubernetes team, *Introducing hierarchical namespaces*, (Accessed 25/04/2021). [Online]. Available: https://kubernetes.io/blog/2020/08/14/introducing-hierarchical-namespaces/.

[16] Kubernetes team, *Assigning pods to nodes*, (Accessed 19/04/2021). [Online]. Available: https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/.

[17] Assaf Morag, *Threat alert: An attack against a docker api leads to hidden cryptominers?* [Article (Accessed 09/03/2021)]. [Online]. Available: https://blog.aquasec.com/container-vulnerability-dzmlt-dynamic-container-analysis.

[18] Joel Brandenburg, *Amazon ecr image scanning*, (Accessed 03/05/2021). [Online]. Available: https://docs.aws.amazon.com/AmazonECR/latest/userguide/image-scanning.html.

[19] *Cncf code of conduct*, (Accessed 19/05/2021). [Online]. Available: https://github.com/cncf/foundation/blob/master/code-of-conduct.md.

[20] Kubernetes team, *Network policies*, (Accessed 18/05/2021). [Online]. Available: https://kubernetes.io/docs/concepts/services-networking/network-policies/.

[21] *Flannel github*, (Accessed 09/05/2021). [Online]. Available: https://github.com/flannel-io/flannel.

[22] X. Nguyen *et al.*, *Network isolation for kubernetes hard multi-tenancy*, 2020.

[23] L. Chen, R. DeJana and T. Nassar, *Sharing enterprise cloud securely at ibm*, 2021. DOI: 10.1109/MITP.2020.2977029.

[24] *Kops website*, (Accessed 16/05/2021). [Online]. Available: https://kops.sigs.k8s.io/.

[25] *Kops networking overview*, (Accessed 16/05/2021). [Online]. Available: https://kops.sigs.k8s.io/networking/.

[26] Kubernetes team, *Pod security policies*, (Accessed 18/05/2021). [Online]. Available: https://kubernetes.io/docs/concepts/policy/pod-security-policy/.

[27] Kubernetes team, *Podsecuritypolicy deprecation: Past, present, and future*, (Accessed 04/05/2021). [Online]. Available: https://kubernetes.io/blog/2021/04/06/podsecuritypolicy-deprecation-past-present-and-future/.

[28] *Dependencytrack github*, (Accessed 29/04/2021). [Online]. Available: https://github.com/DependencyTrack/dependency-track.

[29] Karen Bruner, *Container image security: Beyond vulnerability scanning*, (Accessed 18/04/2021). [Online]. Available: https://www.openshift.com/blog/container-image-security-beyond-vulnerability-scanning.

Appendix A

Project plan

Norwegian University of Science and Technology

DCSG2900 - Bachelor Thesis Bachelor of Science in Digital Infrastructure and Cyber Security 2021

Project Plan - Security within a multi-tenant kubernetes cluster

Authors:
Simen Asplund Kjeserud
Vemund Rahm
Sonja Yoosuk Sanden
Eirik Tobiassen

8th March 2021

Contents

Chapter 1

Goals and Limitations

1.1 Background

Geodata AS runs a kubernetes platform hosted in Amazon Web Services that is used by their development teams to deploy applications and services for multiple customers in a multi-tenancy configuration.
There are generally two different ways to run a kubernetes platform that hosts multiple applications like this. One way, and perhaps the most obvious, is to run multiple clusters in kubernetes, one for each tenant (different customers). Another way is to implement multi-tenancy. In such a configuration, multiple tenants all share resources deployed in a single cluster. Geodata AS runs a kubernetes platform for multiple tenants, using the latter approach. It is important that these tenants are separated properly, so that security and segregation between customers is well maintained.

1.2 Goals of the project

The goal of this project is to research the security in a multi-tenant kubernetes cluster. The project will focus on access control between tenants, especially how the information between tenants are secured.

1.3 Limitations

The project will have the following limitations:

- Research will be done on Amazon Web Services
- The research must have a reasonable spending
- The research environment will be a smaller scale version of Geodatas cluster
- Research has to be done on a multi-tenant kubernetes cluster

Chapter 2

Scope

2.1 Subject area

The main focus of this project is researching multi-tenant kubernetes clusters and the security between tenants. We will be dealing with subject areas such as:

- Infrastructure as code
- Infrastructure operation
- Software development
- Software and infrastructure security

2.2 Issue / Task description

Look at general solutions for multi-tenant services, consider how security can be effectively maintained in such an environment, compare our findings with Geodata's experiences and make recommendations and best practices based on this.

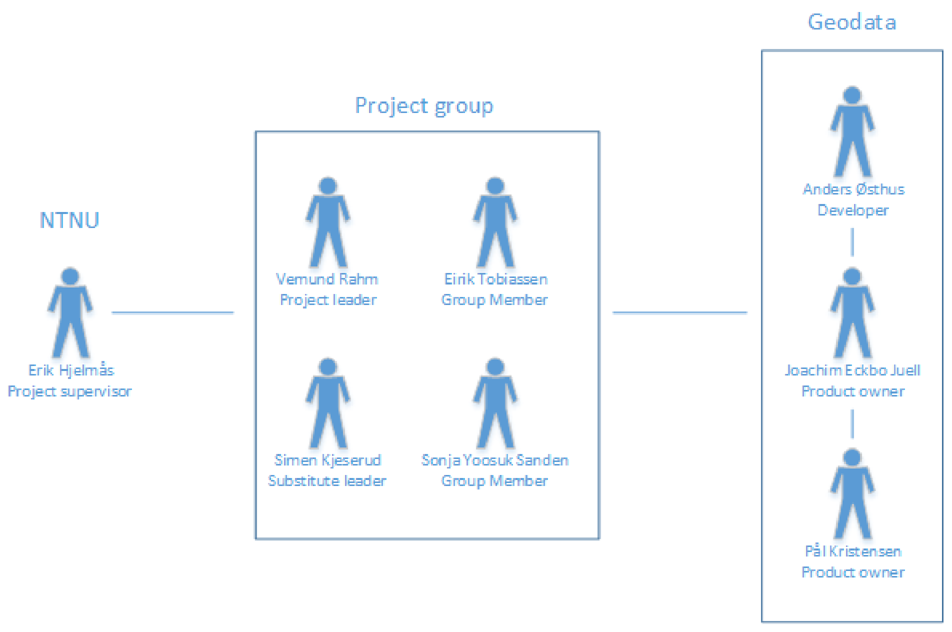
Project focus areas:

1. Overview of the platform and its components (e.g: pipeline, DNS, CI/CD, etc)
2. Security/project analysis of Geodata's platform
    a. How is the security and separation between projects (tenants) achieved and maintained
    b. List of existing security measures
    c. Security measures that should be implemented
3. Look at how Geodata's platform is set up on Amazon Web Services, and replicate this setup ourselves (on a smaller scale)
4. Research best security practices for a generalized multi-tenant kubernetes environment
5. Based on what we find, make recommendations for Geodata to consider

Chapter 3

Project Organization

3.1 Roles and responsibilities



3.2 Group rules

The group have outlined a rule structure as a separate document, that has been signed by all members. The full document can be found in Appendix A (in Norwegian), but a quick summary is provided here:

- Microsoft Teams will be used for communication, while overleaf will be used to write the report.

- The group will have one status meeting every week on Tuesdays, and a summary meeting every other Friday.
- All work should be done within the agreed upon time, if this is not met, the group leader should be informed.
- If a member cannot meet to an agreed time, a 12 hour notice must be given.
- If there are any academic disagreements, the groups supervisor should be contacted.

Chapter 4

Planning, Follow-up and Reporting

4.1 Progress plan

The project owner has expressed no specific needs in terms of the research, planning, testing or writing phases of the project. We will be supplied with the necessary resources to research and experiment with an infrastructure test environment, but the methodology for writing our paper is up to us. Considering the current COVID pandemic our workflow will have to be fully digitized to be able to quickly adapt to changes both internally in the group and external factors, such as governmental restrictions to reduce the spread of the virus.

4.2 System development model / Process model

The paper we are writing is dependent on our ability to do research effectively. For that reason, it is important that we don't waste time while we also keep track of what the other group members are doing to mitigate the chance of working on something that is already done, in addition to helping out if someone are stuck. It is also important for us to have a clear idea of what each of our tasks is, so we don't end up misunderstanding what we need to get done. As a consequence of our group consisting of four people with no/little previous history working together it is also important that we have a way to know if someone has taken on too much or too little responsibility.
Keeping all of this in mind we decided to settle on the agile process model called *Scrum*.

Following the *Scrum* approach allows us to easily organize our project into smaller and more manageable tasks, which allows us to prioritize tasks more effectively and have a detailed schedule. This will also help us notice at an earlier stage if we are behind the schedule and need to re-prioritize our tasks in the backlog. We are using the tasks tool in Microsoft Teams to keep track of our tasks during each sprint.

4.3 Plan for status meetings and decision points in the period

The group have fixed meetings every week, some of which are mentioned in the group rules as well. (There might be additional meetings based on the need):

- On every second Tuesday at 09:00 (bi-weekly) we have a meeting where we plan the sprint that will happen for the next to weeks. At these meetings we set up a backlog of tasks and prioritize them.
- At every Tuesday from 09:00 where there is not a sprint planning meeting, we have a smaller meeting (mid-sprint review) that will just focus on re-prioritizing tasks in the backlog if we notice that we are behind the schedule.
- There is meetings with our supervisor every Wednesday from 09:00 to 09:30
- There is meetings with our client Geodata every other Thursday from 13:00 to 13:30 (the day before sprint review)
- On every second Friday from 14:30 (about two weeks after the previous sprint planning meeting) we have a sprint review meeting where we summarize what we have done, and make sure that we finished everything we had planned for that sprint, in addition to review whether or not we should have done something different and this way improve our methods before the next sprint.
- We have frequent meetings / stand-up meetings through the week where we discuss:
  - What was my contribution yesterday?
  - What shall be my contribution today?
  - Are there any problems I am stuck with / Do I need help from the other group members?

  These are planned during each sprint planning meeting, because there are days where group members don't have time because of other classes and work.

Decisions will as mentioned in the group rules, be done by the whole group and if the group can not find common ground, we will ask our supervisor for input.

Chapter 5

Quality Assurance

5.1 Documentation and source code

All source code, scripts, and other items of such a nature, that do not contain infrastructure-revealing code, or any other code that presents a security issue, will be maintained in a version control system, so the group can more easily verify who has written what, and see all changes made. Any such code is required to be documented (i.e. commented) well.
The working and writing tools the group will use for the project are outlined in the group rules in Appendix A.

5.2 Risk analysis

The group sees that there is a risk that the project may be inhibited by several factors. Here we will list the ones we have identified, along with mitigating factors we will take, if any.

- Long-term disease, student drop out, etc.
  Mitigating factors:
  - If this occurs, assess whether the work can be turned over to the other members, or if the scope will have to be reduced.
- Geodata loses the ability to work with us, due to external factors, e.g. they cannot spare time due to lots of deadlines in their regular work.
  Mitigating Factors: None
- The work is more complicated than the group assumes
  Mitigating factors:
  - Make sure we properly understand the underlying systems before starting to write the core parts of the assignment.
  - Reduce the scope, if the issue arises.
- The group loses the overleaf document, containing the work

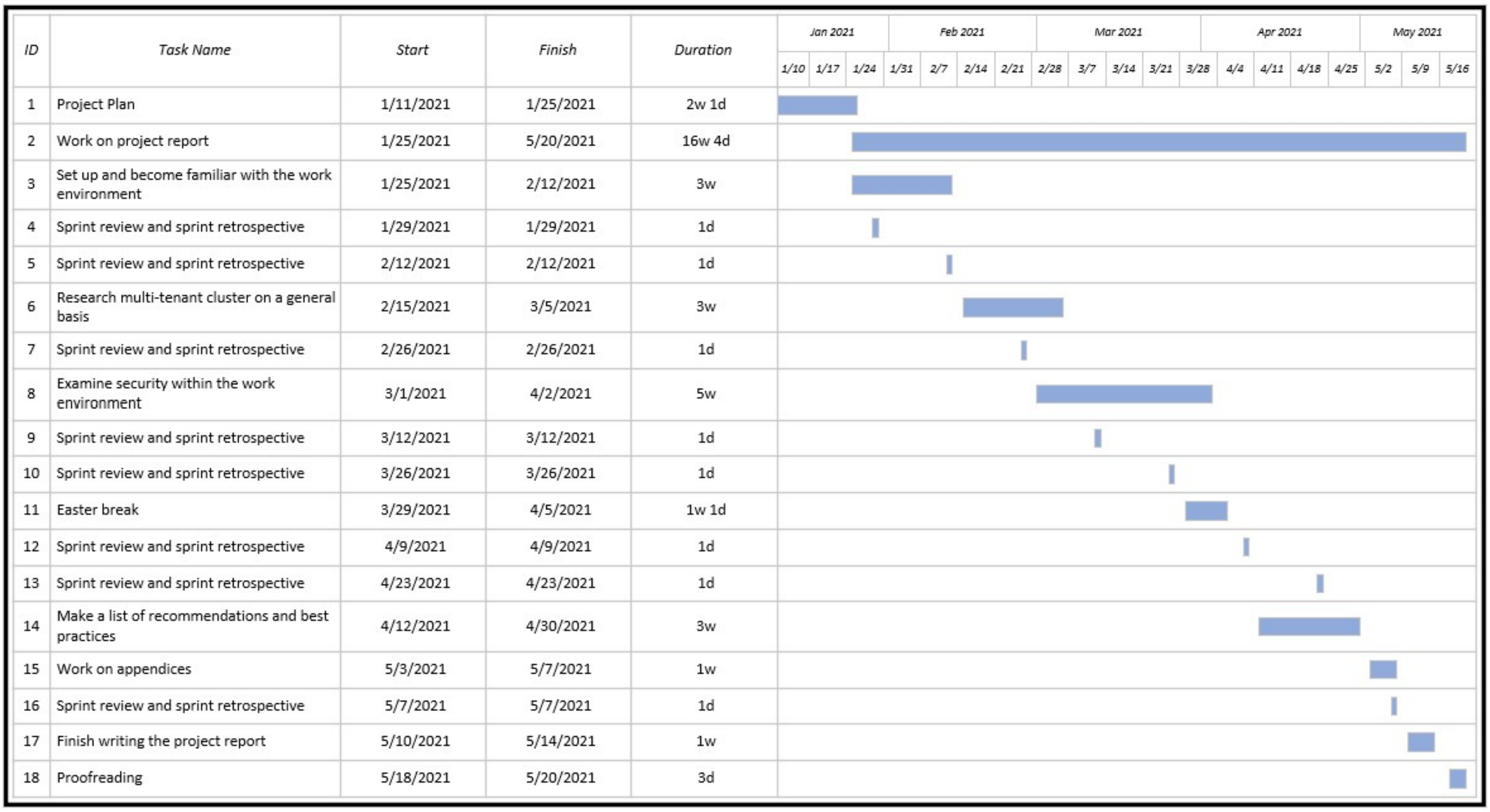
Mitigating factors:

- Overleaf stores all documents on their servers, and there is a history section to see previous changes. Should their infrastructure completely fail, we will use the regular updates that we have sent to our project supervisor and client as a backup.

Chapter 6

Executive plan

6.1 Gantt Chart



ID	Task Name	Start	Finish	Duration
1	Project Plan	1/11/2021	1/25/2021	2w 1d
2	Work on project report	1/25/2021	5/20/2021	16w 4d
3	Set up and become familiar with the work environment	1/25/2021	2/12/2021	3w
4	Sprint review and sprint retrospective	1/29/2021	1/29/2021	1d
5	Sprint review and sprint retrospective	2/12/2021	2/12/2021	1d
6	Research multi-tenant cluster on a general basis	2/15/2021	3/5/2021	3w
7	Sprint review and sprint retrospective	2/26/2021	2/26/2021	1d
8	Examine security within the work environment	3/1/2021	4/2/2021	5w
9	Sprint review and sprint retrospective	3/12/2021	3/12/2021	1d
10	Sprint review and sprint retrospective	3/26/2021	3/26/2021	1d
11	Easter break	3/29/2021	4/5/2021	1w 1d
12	Sprint review and sprint retrospective	4/9/2021	4/9/2021	1d
13	Sprint review and sprint retrospective	4/23/2021	4/23/2021	1d
14	Make a list of recommendations and best practices	4/12/2021	4/30/2021	3w
15	Work on appendices	5/3/2021	5/7/2021	1w
16	Sprint review and sprint retrospective	5/7/2021	5/7/2021	1d
17	Finish writing the project report	5/10/2021	5/14/2021	1w
18	Proofreading	5/18/2021	5/20/2021	3d

6.2 Work breakdown structure

- Introduction to existing platform
    - Gather information on existing security mechanisms
        - Platform infrastructure
        - Dedicated Intrusion Detection- and Intrusion Prevention Systems
        - Specific design considerations in their platform
    - Risk analysis
        - Establish acceptable risk levels
- Research existing multi-tenant configurations
- Finalize recommendations and best practices
    - Analyze suitability for different security measures

6.3 Milestones

- When we have made an overview over the platform, components and security measures
- When the analysis of Geodata's platform is done
- When the recommendations and best practices are completed
- When the project report is finished

6.4 Plan for time and resource spending

The group will initially spend their time according to the Gantt Chart outlined earlier in this section, however estimating exactly how long the tasks in this assignment will take is imprecise, at best. No one in the group has worked with multi-tenant clusters before, and we don't know exactly what we will encounter. We expect that we will find interesting parts of the platform to research, as we begin to learn how the system works. We simply plan to compare our findings through researching multi-tenant clusters, with what we see in Geodata's implementation, and see what issues or areas of research result from these observations.

The group expects a need to familiarize ourselves more with Kubernetes, Amazon Web Services (AWS), the cluster setup itself, and other technologies used by Geodata in the first three weeks of the project. Geodata has kindly offered to host a smaller scale version of their multi-tenant setup in AWS, which we will use to evaluate the platform.

Appendix A

Project contract

Gruppenummer: 120

DCSG2900 - Bacheloroppgave Bachelor i digital infrastruktur og cybersikkerhet

Eirik Tobiassen, Simen Kjeserud, Sonja Yoosuk Sanden, Vemun Rahm

January 12, 2021

Contents

1 Beskrivelse

Gruppen består av 4 medlemmer som har ulik erfaring innen IT-sikkerhet. Formålet med dette dokumentet er å gjøre rede for regler alle gruppemedlemmene skal følge, roller til de ulike medlemmer, gjøre rede for felles kommunikasjonsplatfrom og verdier, kriterier for møter, arbeidskrav, og konsekvenser ved regelbrudd eller avvik fra *kontrakten*.

2 Organisering

2.1 Roller

Leder *Vemund Rahm.*

Vara *Simen Kjeserud.*

Sekretær Går på rundgang

Medlem Gjøre ting.

2.2 Ansvar

Leder

- Delegere arbeidsoppgaver ved behov.
- Ta faglige avgjørelser ved konflikter.

Vara

- Hjelpe gruppelederen med sine ansvar ved behov.
- Ta over leder-rollen hvis lederen svikter.

Sekretær

- Ta notater av hver møte i stikkordsform.

Gruppemedlemmer

- Gjøre arbeidsoppgaver som delegert av gruppelederen.
- Møte opp på flest mulige gruppemøter.

3 Kommunikasjon- og arbeidsplatformer

Teams blir brukt til å delegere arbeidsoppgaver og til kommunikasjon innad i gruppen. Arbeidsplatformen gruppen bruker er Overleaf.

4 Møter

Organisasjon Minst én gang i uken skal det utføres et *statusmøte* internt i gruppen. Flere møter kan bli organisert etter behov, og kan bli holdt over nettet via forhåndsbestemt kommunikasjonskanal. Det skal utføres et *oppsummeringsmøte* minst én gang annenhver uke, hvor gruppen diskuterer og oppsummerer det som har blitt gjort.

Tidspunkt Statusmøter vil bli holdt hver tirsdag klokken 09:00. Oppsummeringsmøter vil bli holdt annenhver fredag klokken 14:30.

Struktur Statusmøtene vil starte med en situasjonsrapport med varighet på rundt fem minutter. Deretter skal gruppen diskutere rundt hva agendaen for neste møtet skal være. Ved slutten av statusmøtet skal det være bestemt hva som skal bli gjort til neste statusmøte, samt hva temaet skal være.

5 Regler

§1 Arbeidskrav

a. Arbeid som suppleres til alle gruppemedlemmer skal utføres innen avtalt tid.

b. Når og hvordan arbeid utføres er opp til den enkelte som har ansvar.

c. Hvis ikke arbeid utføres innen den bestemte tid skal det bli tatt opp med gruppeleder.

§2 Oppmøtekrav

a. Møter skjer ved avtalt tidspunkt.

b. Statusmøter tar sted hver tirsdag klokken 09:00, med mindre annet er avtalt.

c. Hvis et gruppemedlem ikke har mulighet til å møte opp skal det gis beskjed minst 12 timer i forvei til gruppen.

§3 Møtereferat

a. Kort referat skal bli skrevet under hvert møte av daværende sekretær.

§4 Opplysning om avvik i §1 Arbeidskrav

a. Gruppemedlemmer er pliktig til å gi beskjed tidlig om de ligger bak med tildelt arbeid.

b. Hvis gruppemedlemmer oppdager brudd på reglement skal dette bli tatt opp med gruppen før en eventuell beslutteIse blir vedtatt.

§5 Faglige uenigheter

a. Når gruppen skal ta stilling til et problem skal alle medlemmers mening bli tatt i betrakning.

b. Ved faglige uenigheter, hvor gruppen ikke klarer å komme til noen form for enighet skal veileder (Erik Hjelmås) involveres.

6 Sanksjoner og konsekvenser ved regelbrudd

a. Ved første brudd på reglementet vil det bli gitt muntlig advarsel fra gruppen, gjennom gruppeleder til vedkommende.

b. Ved andre brudd på reglementet vil det bli sendt en skriftlig klage fra gruppen, gjennom gruppeleder til vedkommende.

c. Ved tredje brudd på reglementet vil faglærer kontaktes angående problemstilling.

Medlemmers signatur

Navn: Simen Kjeserud

Signatur: Simen A. Kjeserud Dato: 12.01.2021

Navn: Vemund Rahm

Signatur: Vemund Rahm Dato: 12.01.2021

Navn: Sonja Yoosuk Sanden

Signatur: Sonja Y. Sanden Dato: 12.01.2021

Navn: Eirik Tobiassen

Signatur: Eirik Tobiassen Dato: 12.01.2021

Appendix B

Addons

Cert-manager

- All of the certificate handling is integrated.
- Uses Let's Encrypt, which saves money.
- Let's Encrypt is a free, automated, and open certificate authority (CA), run for the public's benefit.
- Digital certificates are needed in order to enable HTTPS (SSL/TLS) for websites.

Cluster-AutoScaler

- Scaling and adjust the amount of nodes based on needs
- This helps the system run optimally and makes sure the system has enough resources

Descheduler

- This addon is used for balancing clusters so they are more evenly distributed among the node group.
- This is important for larger clusters.

efs-csi (AWS EFS file systems)

- Some pods are dependant on using the same storage as other pods; For example if multiple pods needs to read the same file that is uniquely generated by another pod.
- Geodata could also have used Rook for this, seeing that Rook also has support for a filesystem, but they decided that using efs-csi was a better solution and simpler to use.

External-dns

- Creates DNS entries. Has a namespace filter and can control what domains each project has access to.

Extras

- PodMonitor

nginx-ingress-controller

- Responsible for the Ingress in the Kubernetes cluster.
- Traffic management solution.

node-termination-handler (AWS)

- Responsible for making sure that nodes are terminated gracefully.

Postgres-operator

- Manages PostgreSQL clusters on Kubernetes.

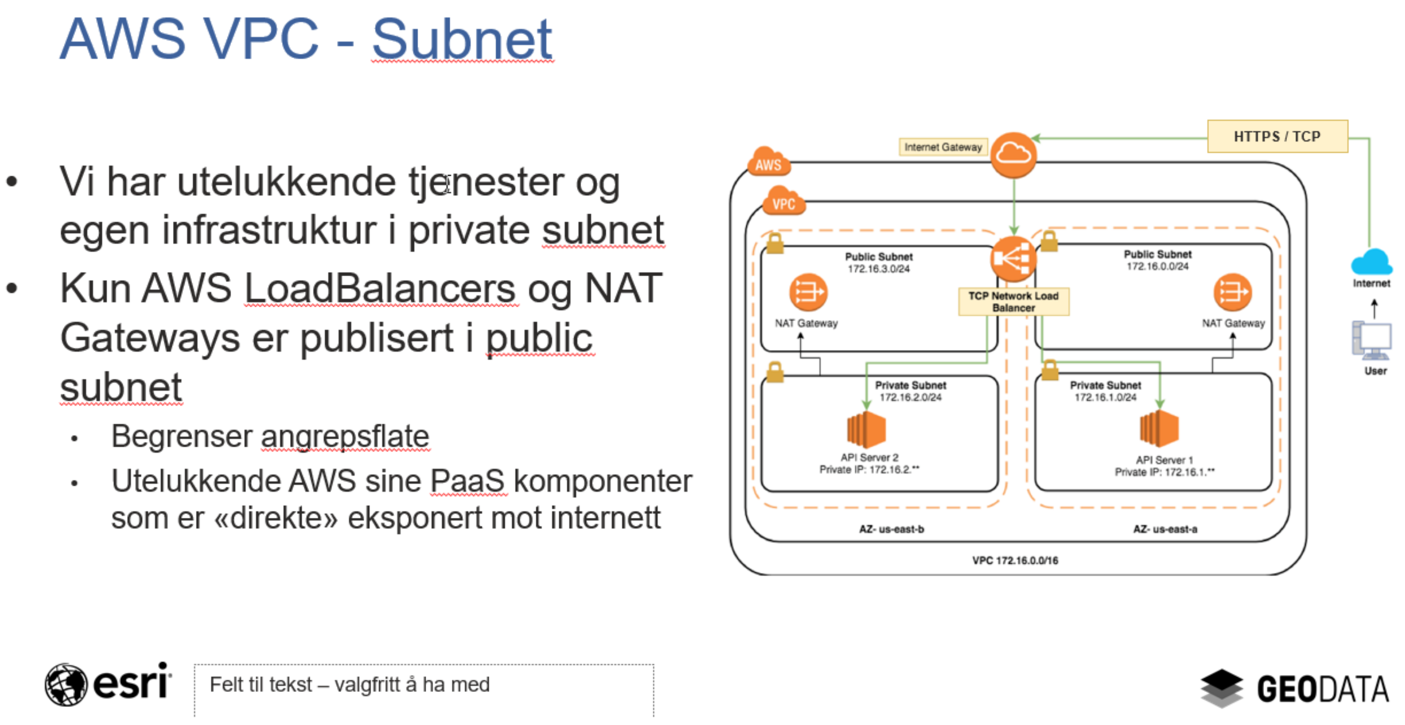
RabbitMQ-operator

CloudFormation

- Amazon technology.
- Templating system.
- Can define resources, which Amazon will define and create for you.
- Very similar to yaml templates in Kubernetes.
- Used for building VPC (Amazon Virtual Private Cloud); Don't have to use GUI, etc.

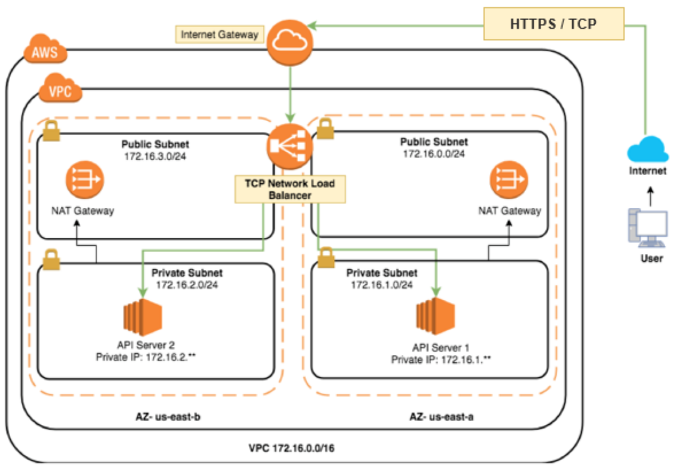
Appendix C

Geodata PowerPoint Slide



AWS VPC - Subnet

- Vi har utelukkende tjenester og egen infrastruktur i private subnet
- Kun AWS LoadBalancers og NAT Gateways er publisert i public subnet
  - Begrenser angrepsflate
  - Utelukkende AWS sine PaaS komponenter som er «direkte» eksponert mot internett



esri | Felt til tekst – valgfritt å ha med | GEODATA

Figure C.1: Slide from a Geodata PowerPoint