

Olav Valle
Simen Nesse Wiik

3D Printer Farm Management System

Bachelor's thesis in Computer Science Engineering
Supervisor: Kjell Inge Tomren
May 2022

Olav Valle
Simen Nesse Wiik

3D Printer Farm Management System

Bachelor's thesis in Computer Science Engineering
Supervisor: Kjell Inge Tomren
May 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

”The greatest
teacher, failure is.
- *Yoda*

I Acknowledgements

We would like to dedicate this text to Paul Steffen Kleppe for his guidance, inspiration and the unwavering support and trust he has placed in us throughout the project. Paul Steffen has had an enthusiasm and drive for the project that often outshone that of our own. Without his management of the logistics tied to the systems that our project depended on at MANULAB, we would not have gotten anywhere. We consider ourselves incredibly fortunate to have had him as both product owner and mentor.

A special "thank you" also goes out to the faculty and students at MANULAB, for providing excellent support, insight and companionship throughout the project.

We would also like to thank all the teachers and lecturers that have strived to instil their knowledge and wisdom into us during our years at NTNU Ålesund.

Our supervisor, Kjell Inge Tomren, also deserves recognition for his patience and guidance throughout the project.

II Abstract

In the summer of 2021, students at NTNU Ålesund developed a prototype for a management system for the printer farm at MANULAB Ålesund. In fall of that year, the faculty of the Department of Ocean Operations and Civil Engineering (DOOCE) at NTNU Ålesund recruited student assistants from the computer science program to continue development of this project. The intention was to expand the functionality of the prototype, and re-implement it as a web application.

Examination by the students found that the existing prototype, while a good proof-of-concept, would require an extensive redesign to fulfill the requirements laid out by the DOOCE. As the work involved in this endeavour would be more than the students had capacity for alongside their regular studies, one of the students involved suggested that they could take on the project as their bachelors thesis in the spring semester of 2022. The department agreed to this suggestion, and recommended that the student continue working on the project through the fall semester of 2021, as a form of planning and design phase for the project. The student recruited a colleague from their study program, and submitted an application to have the project approved by their supervisors.

Work on this project started in full in January of 2022. At the agreement of both the department faculty and the student team, an additional requirement goal was added to the project; the design and implementation of a server solution that could be used for both the farm management system being developed, as well as being a foundation for future software and Internet of Things (IoT) projects at MANULAB Ålesund.

The result of this project is the design and prototype of a server infrastructure consisting of clustered ARM architecture single-board computers, which provides both horizontal scalability and high-availability of the containerized services it hosts. A prototype of the 3D printer farm administration system was hosted on this server as a containerized web app front end and back end system, both implemented in TypeScript. The back end connects 3D printers to the web app front end, allowing for uploading of 3D models for printing and supervision of the status of the printers. The front end and back end systems communicate using REST API's and uses the WebSocket protocol to transmit real time data from the printers.

III Sammendrag

Sommeren 2021 utviklet studenter ved NTNU Ålesund en prototype for et styringssystem for printerfarmen ved MANULAB Ålesund. Høsten samme år rekrutterte fakultetet fra Institutt for Havromsoperasjoner og Byggeteknikk (IHB) ved NTNU Ålesund studentassistenter fra informatikkprogrammet for å fortsette utviklingen av dette prosjektet. Intensjonen var å utvide funksjonaliteten til prototypen, og re-implementere den som en webapplikasjon.

Undersøkelse utført av studentene fant at den eksisterende prototypen, selv om den var en god proof-of-concept, ville kreve omfattende redesign for å oppfylle kravene fastsatt av IHB. Siden dette arbeidet var mer enn studentene hadde kapasitet til ved siden av sine vanlige studier, foreslo en av de involverte studentene at de kunne ta på seg prosjektet som bacheloroppgave i vårsemesteret 2022. Instituttet sluttet seg til dette forslaget, og oppfordret studenten å jobbe videre med prosjektet gjennom høstsemesteret 2021, som en form for planleggings- og designfase for prosjektet. Studenten rekrutterte en kollega fra studiet sitt, og sendte inn søknad om å få prosjektet godkjent av sine veiledere.

Arbeidet med dette prosjektet startet for fullt i januar 2022. Etter enighet mellom instituttet og studentteamet ble det lagt inn et ekstra kravmål til prosjektet; design og implementering av en serverløsning som kunne brukes til både printerfarmstyringssystemet som skulle utvikles, samt være et fundament for fremtidige programvare- og Internet of Things (IoT)-prosjekter ved MANULAB Ålesund.

Resultatet av dette prosjektet er designet for og prototypen av en serverinfrastruktur som består av en klynge ARM-arkitektur enkeltbordsdatamaskiner, som muliggjør både horisontal skalerbarhet og høy tilgjengelighet for de containeriserte tjenestene den er vert for. En prototype av styringssystemet som ble utviklet for 3D-printerfarmen kjører på denne serveren som et containerisert nettapp-frontend og backend-system, begge implementert i TypeScript. Bakenden kobler 3D-printere til nettappens frontend, noe som muliggjør opplasting av 3D-modeller for utskrift og overvåking av statusen til skrifterne. Frontend og backend kommuniserer gjennom REST API og bruker WebSocket-protokoll for overføring av sanntidsdata fra printerene.

IV Foreword

This document is the final report of the bachelors thesis project of Olav Valle and Simen Nesse Wiik. The project was completed in the spring semester of 2022, as part of the Computer Science Engineer program at NTNU Ålesund. It was undertaken on behalf of the Department of Ocean Operations and Civil Engineering at NTNU Ålesund (DOOCE), at the request of Paul Steffen Kleppe, Irina-Emily Hansen and Ola Jon Mork.

The aim of this project was to develop the server infrastructure and web application for a management system for the 3D printer farm at MANULAB Ålesund. The requirements and goals of the project were planned out in cooperation between the two students and the aforementioned faculty members of DOOCE. The project builds upon previous work done by students at MANULAB in the summer of 2021, and the results from this work served as a prototype and proof-of-concept for the products developed in this project.

The students chose this project because of their interest in 3D printing, system administration and server infrastructure design. The opportunity to work with software and hardware in a cross-disciplinary environment, in close cooperation with students and teachers at the automation and mechanical engineering programs, also interested the team.

V Problem Formulation

This project aims to design and implement the software and hardware infrastructure for a 3D printer farm management system. A web application user interface for interacting with the farm from a user and administrative perspective is to be designed and implemented. The server infrastructure being designed and developed will host the web application services, and as serve as the foundation for future projects at MANULAB Ålesund.

Contents

- I Acknowledgements** **2**

- II Abstract** **3**

- III Sammendrag** **4**

- IV Foreword** **5**

- V Problem Formulation** **6**

- Contents** **7**

- List of Figures** **13**

- Glossary** **15**

- Acronyms** **16**

- 1 Introduction** **18**
 - 1.1 Background 18
 - 1.2 Existing Solutions 18
 - 1.3 Aim 19
 - 1.4 Structure 19

- 2 Theory and Materials** **20**
 - 2.1 Frameworks and technologies 20
 - 2.1.1 TypeScript and JavaScript 20
 - 2.1.2 Bash 21

2.1.3	YAML	21
2.1.4	Linux	21
2.1.5	HTML	21
2.1.6	CSS	21
2.1.7	3rd party libraries	23
2.1.8	Version control	23
2.1.9	Development Environment	23
2.1.10	Express	23
2.2	Industry 4.0	23
2.2.1	Industry 4.0	23
2.2.2	3D Printers	23
2.2.3	IoT	24
2.3	Client-server communication	24
2.3.1	HTTP	24
2.3.2	MQTT	24
2.3.3	Websocket	24
2.4	Server system infrastructure	25
2.4.1	Single board computers	25
2.4.2	ARM	25
2.4.3	Clustering	25
2.4.4	Scalability	25
2.4.5	System administration	26
2.4.6	Netboot	27
2.4.7	ZFS	28
2.5	Network infrastructure	29
2.5.1	Ethernet	29
2.5.2	Wi-Fi	29
2.5.3	DHCP	30
2.5.4	TFTP	30
2.6	Project management methods	30

2.6.1	Scrum	30
2.6.2	Use case	32
2.6.3	Persona	32
2.6.4	User story	32
3	Method	33
3.1	Scientific method	33
3.2	Hardware	35
3.2.1	Prusa 3D Printers	35
3.2.2	Raspberry Pi	35
3.3	Frameworks and libraries	37
3.3.1	Containers	37
3.3.2	Kafka	37
3.3.3	High availability	37
3.3.4	MQTT	38
3.3.5	Nest	38
3.3.6	Turborepo	38
3.4	Project organization	39
3.4.1	Discord	39
3.4.2	Google Drive	39
3.4.3	Overleaf	39
3.4.4	Scrum	40
3.4.5	Distribution of work	40
3.5	Software and applications	40
3.5.1	OctoPi	40
4	Results	41
4.1	Scientific results	41
4.1.1	Relevance	41
4.1.2	Knowledge base building	41
4.1.3	User testing	42

4.2	Technical results	42
4.2.1	Raspberry Pi management	42
4.2.2	Infrastructure scalability	42
4.2.3	Frontend	43
4.2.4	Backend	43
4.2.5	Storage	43
4.2.6	MQTT	43
4.2.7	Printers, OctoPi and OctoPrint	44
4.3	Administrative results	44
4.3.1	Scrum	44
4.3.2	Distribution of work	44
5	Discussion	45
5.0.1	Printers	45
5.0.2	Raspberry Pi	45
5.0.3	Kubernetes	45
5.0.4	Frontend	46
5.0.5	Turborepo	46
5.0.6	Kafka	46
5.1	Results	46
6	Conclusion and recommendations further work	48
6.1	Conclusion	48
6.1.1	Kubernetes	49
6.1.2	Raspberry Pi	49
6.1.3	Further work	49
7	Effects on society	50
	Bibliography	51
A	Pre-project plan	54

B Requirement specification	69
B.1 Contents	69
B.2 Introduction	69
B.3 Use Case diagram	70
B.4 User Stories	70
B.5 Domain model	71
B.6 Wireframes	72
C Research issues	75
C.1 User Stories	85
D Decision Reports	96
E System documentation	105
E.1 Introduction	105
E.2 Hardware Architecture	105
A Networking	105
B Raspberry Pi	106
E.3 Software architecture	106
A Frontend	106
B backend	106
E.4 Code repository	106
E.5 Containers	106
E.6 Kubernetes	106
A Using Kubectl to administer the cluster	107
B Adding nodes to the cluster	107
E.7 Security	107
E.8 Installation	107
E.9 Source code documentation	107
F Source Code and Bash Scripts	108
F.1 Kubernetes YAML definitions	108

A	IngressRoute	108
B	Deployment	109
C	Service	110
G	Work diaries	111

List of Figures

- 2.1 TypeScript enables code autocompletion 21
- 2.2 Horizontal scaling. Workload is distributed among the PCs 26
- 2.3 Vertical scaling. A single PC is upgraded with more powerful hardware. 26
- 2.4 Use case diagrams consist of actors and use cases. Image source: <https://www.visual-paradigm.com/guide/agile-software-development/user-story-vs-use-case/> 32

- 3.1 Raspberry Pi Imager for easily flashing SD cards 36

- B.1 Use case diagram 70
- B.2 Domain model diagram. 71
- B.3 Wireframe of card module for printer grid. 72
- B.4 Wireframe of the detail printer view. 73
- B.5 Wireframe of the printer grid overview page layout. 74

- E.1 Hardware system diagram. 105

List of Code Examples

1	Example of dynamic typing in JavaScript code.	20
2	Example of static typing error in TypeScript code.	20
3	Example of a YAML file.	22
4	Example of a typical "Hello World" HTML document.	22
5	Example of CSS code.	22

Glossary

bit rot Bit rot (aka. data decay, data degradation) is the gradual degradation of data caused by the accumulation of hardware failures in a storage device. These failures can be caused by the constituent bits in magnetic media losing or altering their magnetic charge. In solid state devices, the decay may be caused by the device losing its electrical charge. In both these cases, the result of the degradation is that one or more bits have their values flipped (bit flipping), e.g. from a 1 to a 0. 28

container Isolated execution environment only containing the required libraries to run a program. 37

data streaming The speed at which sequential bits of data can be read from or written to a storage device. 28

monorepo A code repository with multiple projects, and tools to facilitate sharing code, running tests and caching build artifacts. 38

OS kernel The core of an operating system's software. Facilitates allocation of hardware resources to software processes on the system. 27

root file system The top node of a file system. Contains the files most critical for an operating system's basic functionality. 27

space efficiency The ratio of available storage space in a RAID storage configuration, in relation to the raw storage space of the disks in the configuration. 28

telemetry The collection and transmission of data (e.g. sensor readings or system status reports) from remote devices to a server. 24

Acronyms

API application programming interface. 37

BIOS basic input/output system. 27

CSS cascading style sheets. 21

DHCP dynamic host configuration protocol. 27, 30

DNS domain name system. 30

DSR design science research. 34, 41

HTML hypertext markup language. 21

HTTP hypertext transfer protocol. 24

IOPS I/O operations per second. 28

IoT internet of things. 24, 29, 35, 36, 38

IP Internet Protocol. 27

JBOD Just a Bunch of Disks. 29

MQTT Message Queuing Telemetry Transport. 24

NFS Network File System. 27

OS operating system. 27

PXE Preboot Execution Environment. 27

RAID Redundant Array of Inexpensive Disks, or Redundant Array of Independent Disks. 28

SBC single board computer. 25, 35

TFTP trivial file transfer protocol. 27, 30

UEFI unified extensible firmware interface. 27

VCS version control system. 23

YAML YAML Ain't Markup Language. 21

ZFS Previously the Zettabyte File System. Now named ZFS, or Z File System ("zed file system"). 28

1 Introduction

1.1 Background

This project was undertaken on assignment from MANULAB Ålesund, on behalf of the Department of Ocean Operations and Civil Engineering at NTNU Ålesund. MANULAB is an infrastructure partnership project between the three NTNU campuses (Gjøvik, Trondheim and Ålesund) [1] [2], and is funded by the Norwegian research council.

Manulab (formally IDEA-lab) can be found in the Lanterna building of NTNU Ålesund. It houses machines for production automation, including robot arms, conveyor belts and self-driving robots. It also houses 3D printers, which can be freely used by students and faculty.

1.2 Existing Solutions

Several companies provide 3D printer farm administration solutions that meet many of the specifications that Manulab requires.

The Prusa Automated Farm System[3], while still in closed beta, is targeted towards businesses and large scale industrial printer farms. Astroprint promises a "simple and powerful way to manage 3D printers." RaiseCloud, a 3D printing web platform, boasts features such as statistical insights into the status of the farm, and integrates OctoPrint support in its product.

Common to most systems out there, however, is the fact that they are tied to business models that require payment for any large scale deployments. These commercial products are also usually closed source, which limits the flexibility and customization options available to tailor the solution for specific use cases. Some of these systems are also locked to specific brands of 3D printers, which again limits their relevance and applicability to existing printer farms.

While many of these solutions could provide most of the features and functionality required by MANULAB, a custom solution clearly provides the best possibilities for tailoring and integrating the farm into other systems and projects at MANULAB.

1.3 Aim

The aim of this project is twofold. One part aims to develop a hardware and software system intended to become the foundation for future projects at Manulab, and encompasses the design and implementation of a clustered server architecture for containerised services built from Raspberry Pi hardware. This server solution will host the software developed as the second part of the project; A web application that will act as a frontend for the 3D printer farm at Manulab. This application will provide a user interface for uploading 3D printing files to printers in the farm, and administration of printing jobs in the farm.

1.4 Structure

In the second chapter, required theory to understand the method and result is described. The third chapter describes the methods employed by the team to produce the work. The fourth chapter presents the resulting work. Chapter five discusses why the results came to be. The sixth chapter concludes the work. In the seventh chapter, social, economic and other impacts are described.

2 Theory and Materials

2.1 Frameworks and technologies

2.1.1 TypeScript and JavaScript

JavaScript is a dynamically typed language. It was originally only used for creating interactive web applications, but expanded to desktop application development with the introduction of Node.js. Its core feature is dynamic typing. This means a variable may hold different types of values throughout their lifetime.

```
let favoriteNumber = 103
favoriteNumber = "#103" // Previously held number, now holds string
```

Code 1: Example of dynamic typing in JavaScript code.

The advantage of dynamic typing is concise programs. Because a variable can be expressed without a type, an algorithmic solution in JavaScript is expressed in fewer characters. Another advantage is easy conversion between types.

Dynamic typing is also a disadvantage. Errors that occur as the program is running, also called runtime errors, may happen as the developer was not warned about faulty conversions. The IDE also cannot help with code autocompletion as there is no type information to infer from any variables.

TypeScript

TypeScript enhances JavaScript with type safety. Type safety means the compiler can catch mismatching types at compile time, and eliminates errors at runtime.

```
let favoriteNumber: number = 103
favoriteNumber = "#103" // Error: Type 'string' is not assignable to type 'number'.
```

Code 2: Example of static typing error in TypeScript code.

TypeScript's typing is favorable when programming with an IDE. The TypeScript language server enriches the IDE with autocompletion, shown in figure 2.1.


```
3
4  async function bootstrap() {
5      const app = await NestFactory.create(AppModule)
6      app.
7      app. close (method) INestApplication.close(): Promis...
8      await connectMicroservice
9      enableCors
10     bootstrap enableShutdownHooks
11     enableVersioning
        flushLogs
        get
        getHttpAdapter
        getHttpServer
        getMicroservices
        getUrl
        init
```

Figure 2.1: TypeScript enables code autocompletion

2.1.2 Bash

Bash is a shell scripting language used in Unix-based operating systems.

2.1.3 YAML

YAML Ain't Markup Language (YAML) is a data storage format. It uses spacing and hyphens to denote collections and nesting. Code 3 is an example YAML file.

2.1.4 Linux

Linux is a widely used operating system kernel. A kernel is the software that controls memory, I/O ports, threads and other operating system resources.

2.1.5 HTML

Hypertext markup language (HTML) is a markup language for defining the structure of a document. HTML documents are made up of pairs of tags [4]. Code 4 shows a simple HTML document that displays a page with the text "Hello World!".

2.1.6 CSS

Cascading style sheets (CSS) is a language for defining the style of an XML or HTML document. It consists of selectors, @-rules, properties and values. Code 5 shows an example using both CSS classes and media queries.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: web
spec:
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
      - image: revosw/printman-web
        name: printman-web
```

Code 3: Example of a YAML file.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <p>Hello World!</p>
</body>
</html>
```

Code 4: Example of a typical "Hello World" HTML document.

```
.card {
  width: 100px;
}

@media screen and (max-width: 768px) {
  .card {
    width: 100%;
  }
}
```

Code 5: Example of CSS code.

2.1.7 3rd party libraries

A library is code compiled to a file which other programs may use the functionality of.

2.1.8 Version control

Version control system (VCS) is a tool which keeps track of a history of changes. Although VCS is widely used in programming for versioning code, VCS applies to documents as well. The most common VCS in use today is git. Git features a form of temporary versions through branches. Branching encourages separating features in different branches, enabling context switching between the features [5]. experimenting with new code, and easily disposing the code if the branch is no longer needed.

2.1.9 Development Environment

When developing a project, it is preferable to separate the publicly deployed version from a sandbox version for testing. A development environment resembles the production setup, from databases to API gateways to frontend.

2.1.10 Express

Express is a framework for developing a RESTful API. It abstracts the details of setting correct HTTP headers and request handling by defining abstract concepts such as routers, request/response objects and middleware.

2.2 Industry 4.0

2.2.1 Industry 4.0

Industry 1.0, 2.0 and 3.0 encompass the use of steam, electrical energy, and transistors and ICs (Integrated Circuits) in production machinery respectively. Industry 4.0 emerged as the internet became ubiquitous. With internet, devices can communicate without geographical boundaries, and entire production lines can be monitored and controlled remotely (Howard, 2018).

2.2.2 3D Printers

3D printing is a form of additive manufacturing. A 3D printer is a machine that constructs an object by depositing material in layers until the desired object has been formed. The materials used include metals, concrete, foodstuffs, resin, rubber or plastics. The most common form of 3D printing uses plastic or rubber materials, which is deposited by extrusion through a nozzle. The material is first melted at temperatures between 150-300 degrees Celsius, dictated by the chemical composition of the material. The raw material takes the form of granules or filament. Granules are small pellets of plastic,

similar to those used in injection molding processes. Filament is made from granules that have been melted and extruded into long strands with specific diameters, and is wound on spools akin to fishing line or sowing thread.

A 3D printer creates objects from a pre-processed 3D model. The model can be created using CAD (Computer Aided Design) or artistic modeling software. A model created in this fashion must be *sliced* before it can be printed. Slicing a model involves analysing the structure of the model and the orientation in which it is to be printed. The model is split into a large number layers, each representing a two dimensional horizontal cross-section of the model. These cross-sections are translated to a set of coordinates in a plane (both Cartesian and polar coordinates can be used, depending on the printer design), and describe the path that the extrusion nozzle should follow.

2.2.3 IoT

Internet of things (IoT) is a term used for all small computers connected to a network. The computers may be used for reading measurements such as humidity and temperature, and controlling electronic hardware such as garage doors, door locks, and lights.

2.3 Client-server communication

2.3.1 HTTP

Hypertext transfer protocol (HTTP) is a standardized request-response model for communication. It is widely used for internet traffic involving websites and APIs. It is an application-layer protocol, implemented with TCP as the transport protocol.

2.3.2 MQTT

MQTT (previously an acronym of Message Queuing Telemetry Transport) is a publish-subscribe messaging protocol designed for use by remote IoT devices for transmitting telemetry data in scenarios with unreliable network connections and limited bandwidth.

2.3.3 WebSocket

While the HTTP protocol is request-response oriented, the websocket protocol allows for persistent bidirectional communication. This enables real-time services such as chat, online games and digital twins.

2.4 Server system infrastructure

2.4.1 Single board computers

A single board computer (SBC) is a computer with all essential hardware and ports soldered on the same circuit board. This is in contrast with regular PC motherboards with pluggable hardware. The most popular SBC is the Raspberry Pi.

2.4.2 ARM

ARM is a company specialized in designing CPUs and licensing the designs to other companies. Their flagship CPU architecture, also named ARM, is a leading CPU architecture for embedded and low power devices, as well as being used in laptop computers and mobile devices.

2.4.3 Clustering

A cluster is a group of machines, which can be seen as a single logical unit. In a cluster, all machines cooperate to serve incoming requests. One advantage of a cluster is fault tolerance. A request can be served even when a machine unexpectedly becomes unavailable. Another advantage is the ability to dynamically tune prioritization of workloads. In the event that a service experiences a sudden increase in demand, the cluster can respond by devoting more resources for the particular service.

2.4.4 Scalability

A computer system is installed to solve a problem. The problem may however change over time. The scalability of a system determines how quickly the system can accommodate to new or redefined problems and demands. Scalability of a system is measured according to how much load it can sustain. Storage, network, cpu and memory are the four limiting factors of a computer system.

The two forms for scaling are horizontal scaling and vertical scaling. Horizontal scaling for hardware means utilizing multiple machines with the same hardware. The workload is then distributed among all the machines.

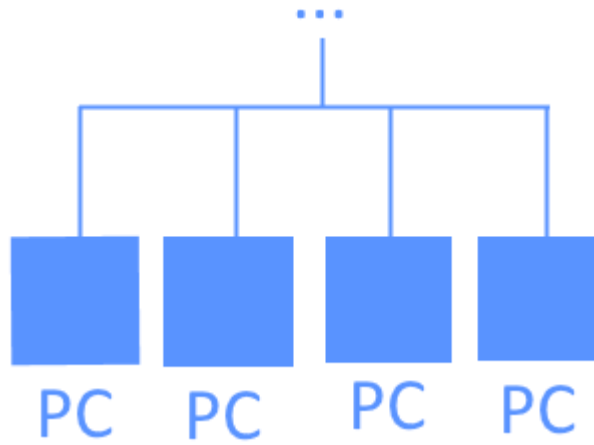


Figure 2.2: Horizontal scaling. Workload is distributed among the PCs

Vertical scaling for hardware means upgrading a single machine with more powerful hardware. The advantage of vertical scaling is better space efficiency. A single machine occupies less space than multiple machines. The disadvantage is that the price of hardware in terms of performance is non-linear. A 32 TB SSD for example is usually more than 32 times expensive than a 1TB SSD.



Figure 2.3: Vertical scaling. A single PC is upgraded with more powerful hardware.

2.4.5 System administration

System administration encompasses the tasks and tools related to configuring and maintaining a software and computer hardware system. Among these are configuring operating systems, file systems and software and hardware security, installing computer and network hardware, as well as performing maintenance on these.

2.4.6 Netboot

Netbooting, or netboot, is a method for providing a computer hardware system with an operating system over a network connection. This differs from booting an operating system in the normal way, where OS kernel files, drivers and software are stored on disks that are local to the computer. In a netbooting scenario, these files are provided to the computer via a network connection. The operating system (OS) files are stored on a server, which provides the files upon request from the computer's bootstrap firmware. On modern hardware, this request and boot process is handled by the Preboot Execution Environment (PXE) of the system BIOS/UEFI firmware. On Internet Protocol (IP) networks, the transfer protocol in question is typically the trivial file transfer protocol (TFTP). Computers configured to request netbooting services locate the server through a dynamic host configuration protocol (DHCP) service on the network. The DHCP server on the network is configured to announce the presence of the netboot server to all clients on the network.

TFTP

The Trivial File Transfer Protocol is a light-weight and simple protocol for allowing clients to get and put files on a host. Due to its simple implementation and small memory footprint, the protocol can be implemented as part of a system's boot firmware. TFTP is widely used in netbooting scenarios to retrieve OS kernel files required by the bootloader firmware. See also: 2.5.4 TFTP

NFS

The Network File System (NFS) protocol was developed by Sun Microsystems in 1984. It allows clients to access and mount remote, distributed, network connected storage devices. The mounted file system can be integrated into the local file system, and used by the OS as if it were local storage. In netbooting scenarios, NFS can be used by the OS kernel (which is loaded using TFTP) of the netbooting client to mount the root file system.

DHCP

The DHCP is a network protocol used on IP networks to automatically assign IP addresses to network clients. In a netbooting scenario, the DHCP server announces the presence of a TFTP server providing netboot services, to the netboot client firmware. See also: 2.5.3 DHCP

PXE

The PXE specification defines the interaction between a netboot enabled client and a server that provides netbooting services. On the client side, PXE is integrated into the network interface firmware. PXE utilises DHCP and TFTP protocols for communication and file transfer between client and host systems. PXE is part of the UEFI standard, and is one of the most commonly used systems for netbooting modern computer systems.

BIOS/UEFI

2.4.7 ZFS

ZFS is a file system and volume manager developed by Sun Microsystems in 2001. Unlike traditional RAID storage systems, where device management, volume management and file system data management are handled by separate hardware or software, ZFS combines these things into one software system. This unification allows ZFS to have greater control of the file system and its constituent storage devices, giving it deeper insight into the integrity of the data stored within. ZFS employs techniques such as data checksumming, which, along with also monitoring device status, is used to detect data corruption caused by bit rot or software errors.

VDEV

The ZFS volume and device manager abstracts physical storage devices into VDEVs (Virtual Devices). A VDEV can consist of one or more devices, which can be assembled in three main topology formations for data storage. These present different benefits and drawbacks in the metrics of IO performance (data streaming and IOPS), fault tolerance and space efficiency.

Striped: A striped VDEV is akin to a RAID 0 configuration. This configuration splits data into blocks that are spread equally across all drives in the VDEV. Striping provides no fault tolerance, and a single disk failure will render the data chunks stored on other disks in the VDEV useless. However, it does provide a benefit in read and write speeds, as the speed of the VDEV is equal to $N * \text{single disk speed}$ for both IOPS and streaming. Striping also allows for 100% storage space efficiency.

Mirrored: A mirrored VDEV stores identical copies of all data on each disk. This results in greater fault tolerance, as all but one of the disks in the mirror can fail, while still retaining full data integrity. Mirroring provides read IOPS and streaming operation speeds equal to $N * \text{single disk speed}$, as the different parts of the data can be read from different disks in parallel. However, speeds for write IOPS and streaming follow the speed of a single drive, as data must be written to each of the disks in the VDEV. Mirroring also provides poor storage space availability, following $\frac{N-1}{N}\%$ for N disks.

RAIDz: RAIDz comes in three varieties; RAIDz1, 2 and 3. Comparable to traditional RAID5 or RAID6, RAIDz provides fault tolerance through data parity information. The number indicates how many copies of the parity information is stored, and hence the number of drives in the VDEV that can fail before data integrity is compromised. All data in the VDEV is striped across the disks, as is the parity information; there are no dedicated parity disks in a RAIDz VDEV. IO performance for RAIDz is equal to single drive performance for read and write IOPS, but streaming follows $(N - P) * \text{single drive speed}$, for N data disk and P parity blocks. Storage space efficiency in RAIDz scales with $\frac{N-P}{N}$, with $P \in \{1, 2, 3\}$. The recommended minimum number of disks N in a RAIDz configuration is $P + 2$.

ZPool

ZFS groups one or more VDEVs into a storage pool. The VDEVs in a pool typically have the same configuration (all mirrored, all RAIDz2, etc), but this is not required. Fault tolerance is nonexistent at the Zpool level in ZFS. All redundancy in a pool is provided by the VDEVs, and ZFS treats the VDEVs in a pool akin to a JBOD storage configuration. The data stored in a Zpool is split into chunks and spread across the storage VDEVs in the pool. Data is spread across VDEVs according to the free storage space of each individual VDEV, and the distribution may also be affected by the IO traffic of a VDEV at the time data is written to the pool (e.g. a busy VDEV may be passed over). If any of the VDEVs in a pool fails completely, the entire pool is compromised and all data is lost.

Data Checksum

ZFS checksums all data stored in the file system. These checksums are not stored in the block of the data itself. Instead, it is stored as part of the file system pointer to the block. The pointers themselves are also checksummed, and these are again stored in the parent block pointer. This practice is employed all the way up the file system hierarchy to the root node. By separating the data from its checksums, ZFS retains control of data integrity in cases where a data block is corrupted due to the failure of physical sectors on the disk (where also the checksum could be corrupted if stored in the same block).

2.5 Network infrastructure

2.5.1 Ethernet

Ethernet is a hardware communication protocol. It is implemented as Ethernet cables, with RJ-45 plugs on both ends. It is used globally to connect network peripherals such as computers, routers, switches and IoT devices.

Routers and switches act as data mediators. By use of the MAC protocol, devices can send packets of data orderly from the source to the destination.

2.5.2 Wi-Fi

Wi-Fi is a technology for wireless communication between devices. Instead of a plug and socket, wireless communication is performed with antennas. Its ease of use has led to adoption in all public and private spaces. Wi-Fi is an implementation of the IEEE 802.11 standard.

Wireless communication uses radio waves to transport data bidirectionally between devices. The most common form of wireless communication happens in the 2.4GHz and 5.0GHz frequency bands [6]. The 2.4GHz band is segmented into 14 channels, each channel being 22MHz wide. For a wireless connection between a computer and a router to be made, the computer occupies one or more channels. The multi-input multi-output (MIMO) rating of a network interface specifies how many channels can be used for simultaneous data transmission. In general, the number of antennas correspond to the MIMO

rating. A 4x4 MIMO network interface therefore often means a network interface with four antennas. Occupying multiple channels means bandwidth increases. Communicating over a single channel means the maximum bandwidth is 22MHz.

This form of communication is prone to interference. Because all devices send and receive data in the same frequency band, the signals may combine into due to the potentially many sources of wireless signals, and physical blockage such as walls. Loss of signal is measured in -dBm, and dense materials such as brick and concrete attenuate the signal stronger than light materials such as plywood and plaster. The choice between wired and wireless communication is therefore a tradeoff between reliability and convenience.

2.5.3 DHCP

For computers in a network to be able to send and receive packets of data, each computer must obtain an IP address. A DHCP server is responsible for assigning each computer a unique address within its configured network. The DHCP server can also be configured to provide other configuration parameters, such as specifying the IP address of a server providing domain name system (DNS) services.

2.5.4 TFTP

TFTP is a simplified standard of FTP which allows for file transfer between two devices. TFTP is often used for deploying updates to simple hardware such as routers, and provisioning operating system installations to remote hardware. TFTP does not employ security features such as authentication. This makes TFTP unfit for sharing sensitive data.

2.6 Project management methods

2.6.1 Scrum

Scrum is an agile development method for projects experiencing frequent changes in requirements. In a scrum managed project, tasks are periodically defined and put in the backlog.

Backlog

The backlog contains all unfinished tasks currently defined for the project. A product backlog keeps track of all the tasks to be done for the product to be considered complete. A sprint backlog is a subset of the product backlog which the team should focus on in the current sprint.

Sprint

A sprint is a short period devoted to complete a discussed set of tasks. The sprint follows a life cycle.

Sprint planning

Sprint planning is the first activity in a sprint. New tasks are brainstormed and put in the product backlog. Tasks are chosen from the product backlog and put in the sprint backlog. After performing sprint poker on the new tasks and agreeing on which tasks to do in the sprint, the sprint commences.

Sprint poker

After creating tasks in the sprint planning phase, the new tasks must be assigned points based on their difficulty. All participants think about which score the task should be assigned, without revealing the score. When everyone is ready, everyone reveals which score they chose. If there is a deviation from the average score, it opens up for discussion on why the individual thinks the score is higher or lower. This way, new insights about the task may be revealed.

Sprint review

The sprint review is an assessment of the product after a period of work. At the end of a sprint, the team shows the work done to stakeholders and discusses product goals [7]. New opportunities and changes in the environment are also discussed and acted on, by reviewing the product backlog.

Sprint retrospective

The sprint retrospective is about assessing the people and processes instead of the product itself. If a task was not completed due to circumstances, it may be discussed and acted on in the retrospective. Through iterations, the people and processes become more efficient.

Roles

There are three roles in a Scrum team.

The product owner is responsible for the overall vision of the project. The product owner keeps track of the product backlog, which is a collection of all tasks to be done for the product to be considered complete.

The Scrum master is responsible for keeping everyone informed about Scrum, and help the team members better practice Scrum. The Scrum master is also In scrum, all members of a team are assigned one of three roles; product owner, scrum master, and stakeholder.

A stakeholder is a person or entity with an interest in the product's success. Examples of stakeholders are the users of a product and the client ordering the product.

2.6.2 Use case

A use case represent a detailed requirement of the product. For a product to have value, it must fulfill a user's needs. Thus, the use case documents how the user interacts with the product to produce value [8]. A use case diagram consist of actors and use cases, as shown in figure B.1.

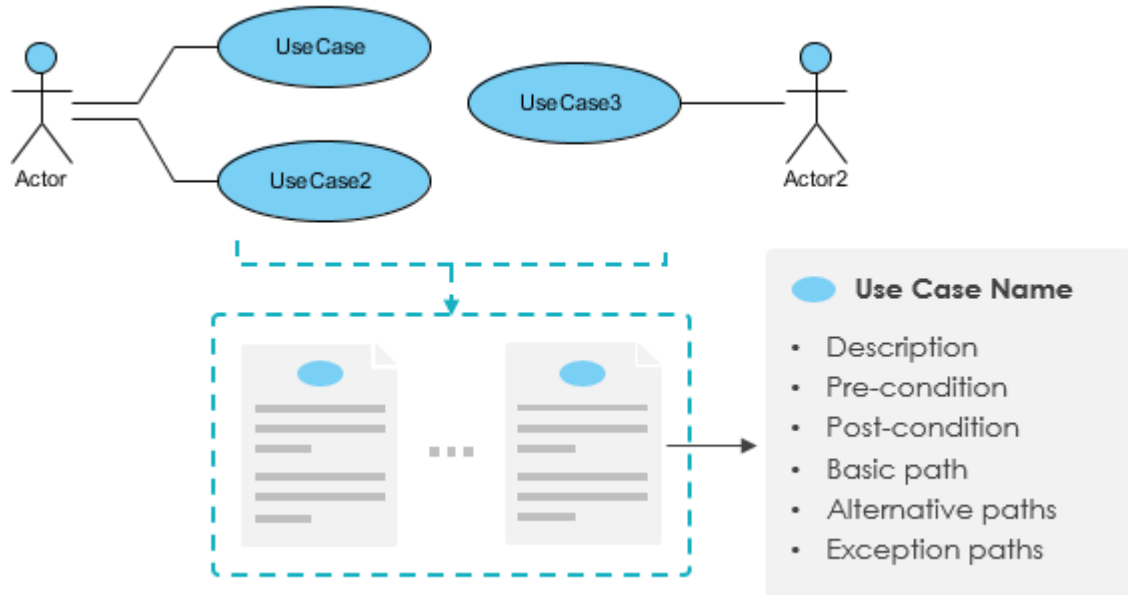


Figure 2.4: Use case diagrams consist of actors and use cases. Image source: <https://www.visual-paradigm.com/guide/agile-software-development/user-story-vs-use-case/>

2.6.3 Persona

In agile terminology, a persona is a fictional character with specific traits such as age, gender, prior knowledge, interests and hobbies, and abilities and disabilities. The goal of a persona is to represent a specific demography and psychography. Personas form the foundation of how features of a hardware or software solution are implemented. Therefore, a mismatch between the personas and the real user leads to a lessened user experience.

2.6.4 User story

A user story is a story-like description of a persona and a goal. An example of a user story is "As a hotel clerk, I want to manually reserve rooms to arriving guests". User stories is a tool to effectively describe a requirement of a product that everyone can understand. This is in contrast to a use case, which aims to describe a requirement in details for implementors.

3 Method

3.1 Scientific method

In the fields of natural science, the tenets of the scientific method are vital to ensure both rigor in the methods employed in the work performed, and to account for possible error or bias in the findings resulting from experimentation.

Due to the practical and problem solving oriented nature of engineering work, the traditional scientific methods of an inductive or deductive approach are not always easily applied. A an engineer does not, as a matter of course, set out to disprove a hypothesis developed through observation, via the means of empirical experimentation. Rather, a problem has been identified, and the task is to solve it.

Hevner et. al [9] and livari[10] both posit that a vital part of the engineering process is the understanding of the problem domain and its solution through the design, application and evaluation of the product of proposed solutions¹.

Here, the scientific methods are vitally applied to the problem solving processes that so much of engineering work consists of. The methodologies and practices prescribed by the various scientific paradigms are applied to the processes of domain modeling, requirement development, knowledge building, and the design and implementation of the product.

These practices help ensure that the results of a development project are both reproducible and of high quality, as well as being relevant to the problem that should be solved. Reproducibility is important to ensure the repeatability of the work, such that the designed product can be taken from a small-scale design and development stage, to large-scale production. Ensuring the relevance of the developed solution, i.e. that it solves the problem it is meant to, is also paramount, as a product without purpose has little to no value.

Furthermore, a rigorous, logical and scientific development process is important in order to ensure that the results and findings produced throughout the project can be documented in such a way that it aides both engineers and researchers in future endeavors. Hevener[11] writes that "understanding and communicating the design science research process is essential ... to establish the credibility of IS design science research among the larger body of researchers [in other fields] ...", emphasizing the importance of scientific rigor in engineering as a means to ensure that the work can be communicated to those in other disciplines.

¹Hevner refers to such products as "*artifacts*"[9]

As many of the concepts and theories the project is based on were largely unknown to the team, extensive work had to be done to build up the knowledge base of the team. This included both knowledge regarding the problem domain (3D printers, Industry 4.0), and subjects related to the engineering work (kubernetes, linux sysadmin, server system provisioning).

Additionally, a lot of knowledge needed to be acquired about the expected use cases of the products being developed, in order to be able to develop the design requirements of the product.

While this project was not intended to be a purely design science research (DSR) project, some of the DSR methodologies and guidelines described by Hevener and livari have been applied to certain parts of the project.

Relevance for the designed systems was sought through qualitative methods, including conversations with and observation of students at MANULAB in order to identify their needs. The team was also provided with several user reports from an assignment that students at MANULAB had been given, where they were asked to test, evaluate and provide feedback on the first prototype of the printer farm system. The feedback in these reports helped inform the requirements that were laid out during the design process.

Plans for further user tests of the system (for qualitative evaluation of the products relevance), in addition to multiple-choice questionnaires (for quantitative evaluation), were planned for when the product reached a usable state. However, these tests were never designed or preformed.

In the work to build the required knowledge and skills to develop the required systems, the team read extensively on the subjects. The sources for this literary study were mainly found online, in the form of the system documentation provided by the developers of the systems and frameworks being employed. Heavy use was made of the documentation of Kubernetes, K3S, the Raspberry Pi SBC, the Linux kernel, the ZFS system, as well as the JavaScript libraries employed for the web application frontend and backend.

It was also necessary for the team to familiarize themselves with the various possible technologies and systems available to choose from. The team approached this by identifying those that appeared most widely used and adopted in their relevant fields. These options were further compared and contrasted based upon criteria such as the quality of documentation, their complexity (both as perceived by the team, and as described by reports from users of these systems), and the teams perception of the difficulty in learning how to use these technologies at the level required.

The approach taken here largely followed the cyclical model described for DRS, with focus on the design and rigor cycles. This approach was felt to allow the team to "learn by doing", which was seen as an appropriate approach given the complexity of the engineering subjects in question as well as their practical, rather than purely theoretical, nature.

3.2 Hardware

3.2.1 Prusa 3D Printers

The 3D printers used in the project were all manufactured by Prusa Research. Their use was not a choice made by the developers, as Manulab already owned many Prusa 3D printers. However, since any 3D printer with an USB interface would suffice for a prototype, selecting another type or brand of printer was not necessary. This saved us from time spent waiting for delivery, and immediate and future monetary expenses.

3.2.2 Raspberry Pi

Raspberry Pi SBCs were used extensively in this project. Each individual printer in the farm system was connected to a Raspberry Pi 4B configured with OctoPrint. The Prusa 3D printers used in this project do not have hardware for network connectivity, and general purpose computers were therefore attached to the 3D printers via USB to provide a service to other computers on the network.

The SBC's were also used to build the server hardware, where they are configured as nodes in a kubernetes cluster. The Raspberry Pi 4B is a capable device when comparing processing power in relation to their size. While a single Raspberry Pi 4B would not be powerful enough to handle the processing load required by the server, a cluster of these devices would allow for horizontally scaling the server hardware in order to meet these requirements.

One reason for their selection was in part due to MANULAB already being in possession of a large stock of these devices. However, an evaluation of the suitability of the Raspberry Pi was done at the start of the project. As their use at MANULAB was largely due to the Raspberry Pi's status as the de facto choice of SBC in automation and IoT prototyping and hobbyist projects, the developers and product owners felt that a review of the alternatives was in order. Many other SBC products exist, some of which could potentially be more suitable and cost effective for the intended application.

When choosing the kind of computer, three criteria needed to be met.

First criteria is to include an RJ45 socket for wired connectivity, for two reasons. Firstly, because of the potentially high volume of printers on the same network, wireless was deemed too unreliable. A proper benchmark of network congestion and packet loss was not conducted since the Idea Lab did not have the hardware to perform such a test. Taking into consideration the reliability goal, the team decided to not introduce the risk of network congestion and packet loss in constrained environments. As the SBCs would also be mounted to racks with ample space for running cables, the team responsible for building these racks did not consider the need for ethernet cabling to be a problem. The second reason for choosing a wired ethernet network, was the added possibility of utilizing PoE. By powering the SBCs via PoE, a tradeoff could be made between the cost and added complexity of providing each SBC with a separate power adapter versus adding PoE hardware.

The second criteria of the SBC is to be ubiquitous. One advantage of ubiquitous hardware is stable supply chains, and therefore stable prices. In the event of a supply chain instability, the price increase and unavailability is not as severe as if another less common SBC had been chosen [12]. Another

advantage is replacability. Hardware failure is inevitable. By being able to respond quicker to hardware failure, the service stays up for longer which benefits all the users. The

The third criteria of the SBC is the ability to run Linux-based operating systems. Software today primarily runs on Windows, Mac OS and Linux. While the Nano and Core versions of Windows Server are engineered to run on low power hardware, the Windows family of operating systems are regulated under a paid license and proprietary code. In contrast, Linux-based operating systems are regulated under a free and open source license. This means choosing Linux will not burden the project with proprietary licenses.

The most ubiquitous SBC with an Ethernet port which runs Linux is the Raspberry Pi. The Raspberry Pi is a staple in IoT.

MicroSD cards

Raspberry Pis use microSD cards as primary storage. While microSD cards are ubiquitous, their slow read/write speed coupled with short lifespan makes microSD cards unfavorable in read/write intensive workloads.

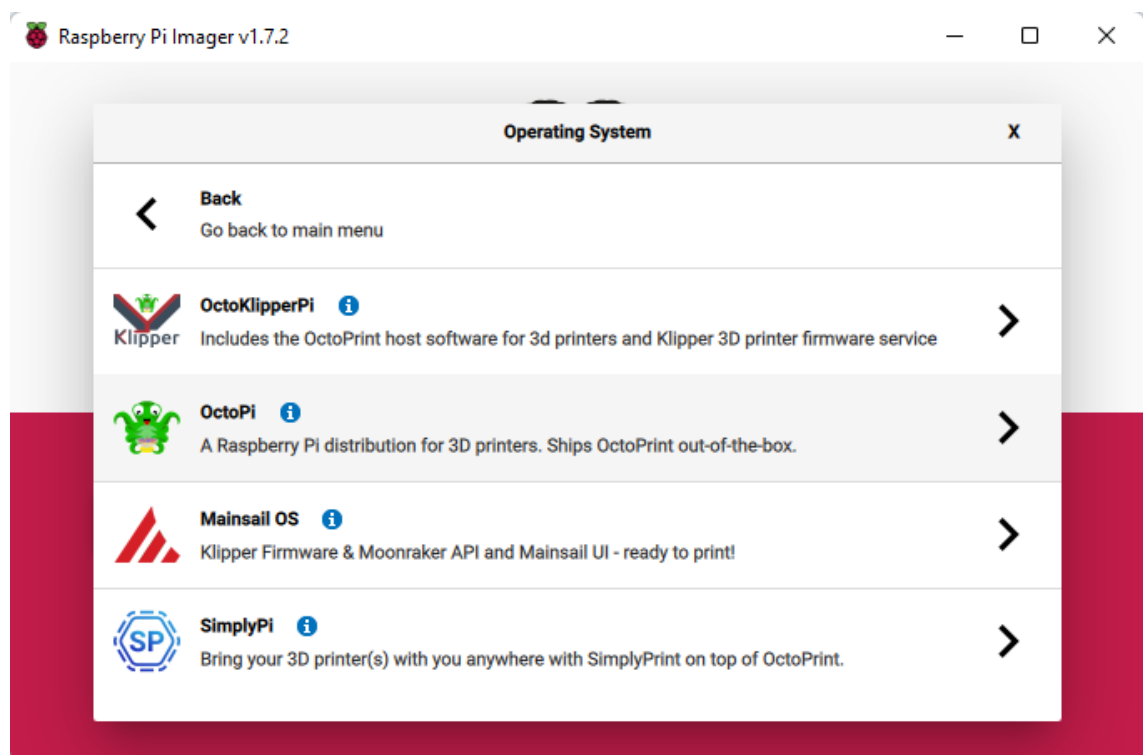


Figure 3.1: Raspberry Pi Imager for easily flashing SD cards

3.3 Frameworks and libraries

3.3.1 Containers

Software relies on the operating system application programming interface (API) to provide access to computer hardware. The API is implemented as software libraries. For all new versions of the operating system, some built-in libraries may change in incompatible ways, rendering dependent software non-functional. Containerization solves this by packaging the software with compatible libraries.

Software however rarely make use of all built-in operating system libraries. To decrease the size of the container, only the necessary libraries are packaged.

3.3.2 Kafka

Apache Kafka is a event streaming platform, combining an event stream processing system and an event store database. It is developed by the Apache Software Foundation. In the early stages of the project, Kafka was intended to be implemented in the project as part of the data and statistics collection part of the backend system.

Kafka was chosen for this because it is a widely used platform in the Industry 4.0 sector, and ideal for handling large systems of IoT devices. Kafka offers a system for handling large streams of sensor data, providing utilities for analytics and data processing. It's intended use in the product was to handle the streams of print job progress reports from the printers in the system. These reports would be processed to extract statistics for the farm administration dashboard, and the collated data would be stored for use in future data analysis projects at MANULAB.

However, Kafka was eventually cut as a requirement. This decision was made after talks between the team of this project and the team of a future project, which is intended to continue developing the data and statistical analysis systems at MANULAB. This was agreed upon partly because the integration of Kafka was not vital to the main goals of this product, as OctoPrint already provided API's that were sufficient for extracting data for print job monitoring. It was also agreed that moving the work pertaining to the design and implementation of the data collection system over to the team that would be using this system in their work, would grant this team greater autonomy in their project. The other team were not previously familiar with Kafka, and wished to explore the available possibilities as part of their work.

3.3.3 High availability

A goal of the PrintMan project was for the system to be highly available. Long delivery times for new parts and untrained personnel means a failure may last for more than 24 hours. Although downtime of the system is not critical at this stage, future extension of the system will be troublesome if the system cannot scale.

3.3.4 MQTT

MQTT was intended to be used as the messaging protocol between the 3D printers and the Kafka data processing system in the backend. MQTT was chosen for this purpose as it is well suited for transmitting sensor data from a large number of IoT devices. Due to its lightweight bandwidth requirements and focus on reliable machine to machine communication in scenarios with constrained networks, as well as its widespread use in Industry 4.0 systems, it was seen as an ideal solution that would allow for both scaling of the printer farm as well as the integration of future projects into the system. Another reason for its selection was the existence of an MQTT messaging plugin for the OctoPrint software. This plugin meant that the team would not have to implement this functionality from scratch, a task that would have added significant complexity to the project.

The use of the MQTT protocol was cut from the project requirements, as part of the decision to cut Kafka. MQTT was not required for transmitting print job progress reports, as OctoPrint already provides this through WebSocket protocols, and was only chosen originally for its integration with Kafka.

3.3.5 Nest

Using express directly in the project is disadvantageous. The express documentation leaves the structure of the project to be solved by the reader. Thus, the project maintainers are responsible for documenting the low-level plumbing. Furthermore, the project will experience a turnover of inexperienced developers every year. As a result, developers most likely end up spending much time learning a partially documented folder structure.

Nest is a web backend framework focusing on code structure. It delegates details about handling http requests and websocket traffic to the express framework, and builds a modular framework on top of it. To use Nest, Developers define modules, services and controllers which are connected automatically by the Nest framework. The burden of documentation is therefore mostly delegated to Nest developer documentation.

3.3.6 Turborepo

The PrintMan system currently consists of a web frontend and backend. To manage multiple projects, it was favorable to use a monorepo tool. Turborepo was chosen as the monorepo tool.

Shared models

Both the frontend and backend are made in TypeScript. Because both projects use the same programming language, it is possible to share the same data models between the projects. One advantage of shared models is no code duplication. If the code is duplicated, the representation of the same model can differ between the frontend and backend. If an incompatible change is introduced only to one version of the model, the frontend or backend will use the data erroneously.

Fast iterations

Running the frontend and backend code in parallel can be done with a single command. Furthermore, for every change in one of the projects, only the affected project restarts. This leads to faster time to start up, and less time to wait between changes.

3.4 Project organization

3.4.1 Discord

Both authors prefer using Discord for casual communication. Separating casual communication on Discord and formal agreements on Jira and Confluence leads to better organization, as important documents are stored in a single, organized location. The Discord platform is also widely used by the DOOCE faculty for communicating announcements to students, and as communication channels internally in project teams. At the request of the product owner, all digital communication between the team and the product owner was done through a dedicated channel on the MANULAB Discord server.

3.4.2 Google Drive

Google Drive was used as a cloud storage for collecting files that did not directly pertain to Confluence documents or were not suitable for storing in a code repository VCS. Instruction manuals, PDFs of books and reports used for literature surveys, templates for reports and thesis attachments, and various other files are stored in Google Drive. Google Docs was used at various stages for collaborating on document writing, creating presentation slideshows, spreadsheets etc.

3.4.3 Overleaf

For writing the thesis, either a word processor application or a document preparation system could be used.

GUI word processing applications are driven by buttons and menus. The functionality in a GUI driven application is easier to use and more intuitive as a first time user, since the buttons are exposed and available to experiment with without needing to consult the documentation at first. However, there are several disadvantages to using a GUI driven word processor. The GUI is rarely customizable or extendable. Since the GUI is designed to fit the average user, many specialized features become cumbersome to use. Neither Microsoft Word nor Google Docs, the two most popular word processor applications, are extendable in a way which is favorable for report writing.

On the other hand, LaTeX is a document preparation system. It is entirely driven by text commands, and is extendable by defining new commands. A significant advantage of LaTeX is being able to leverage community libraries. There are libraries for inserting table of contents, figures, code formatting, glossaries and more.

Overleaf is a web-based, real-time collaborative application for writing LaTeX documents. By using

Overleaf, the team has been able to write a professional-looking thesis with minimal resistance.

3.4.4 Scrum

The team was unfamiliar with many of the technologies used in the project. Thus, making a detailed five-month waterfall plan was not feasible. Instead, using Scrum to plan one week sprints at a time with monthly milestones gave the team time to discover future hindrances.

Roles

Since the team only consists of two people, both were assigned the Scrum master role.

3.4.5 Distribution of work

The project is split into three parts. First part was management of networking, provisioning Raspberry Pi OS to Raspberry Pis, and configuring OctoPrint with correct plugins. The second part was managing Kubernetes installation and configuration, and service deployments. The third part was developing the frontend and backend services.

The first and second part was wholly delegated to a single person, while the third part was shared between both people.

3.5 Software and applications

3.5.1 OctoPi

To interact with the 3D printer, client software and 3D printer drivers are needed. OctoPi is an operating system distribution which includes the required software and drivers. Since OctoPi includes all required software, it removes the burden of manually customizing an installation with self-made scripts. Another benefit is its ease of installation through the Raspberry Pi Imager software.

4 Results

4.1 Scientific results

The scientific methodology applied in this project was based on the cyclical approach described by the design science research (DSR) paradigm[10]. As this project was not a strict DSR project, these methods were adapted to fit the workflow of the team. These adaptations were mainly in the form of the required rigorousness in relation to the knowledge building process, where the team focused more on exploring and evaluating the existing possibilities, and did not always treat this as a scientific literature study.

4.1.1 Relevance

In the work of detailing the domain and requirements for the project, the team approached this mainly through qualitative methods, by engaging in dialogues and observation of the expected end user (students and staff at MANULAB). The team were provided with a collection of user reports from an assignment students had been given before the project started. The task of these assignments were to test and evaluate the early prototype that this project builds upon. The reports from this assignment identified several key points that were used to develop the initial requirements for the project. The findings from these qualitative studies are documented in the form of user stories in Jira, as well as in product requirement documents using a template in Confluence. See appendix C.1 and C for these documents.

4.1.2 Knowledge base building

During the knowledge building process, which constituted a significant part of the project, the team approached the technologies used with a design-rigor cycle approach. Technologies and systems that were to be used, or were being considered for use, in the project were investigated through small-scale trial implementations in order to gain insight into both the theoretical complexity and the implementational complexity of these.

Several technologies were investigated and tested by practical application, and their merits were compared. The results of these comparisons are detailed in two ways. Issues were created in the Jira system for those the team felt merited closer investigation. Comments were added to these issues, and also to the work log entries for these tasks, describing the findings from these investigations. Decision

reports were created in Confluence for those that were selected, describing the reason for their selection.

At times, the blog functionality in Confluence was used as a way to write logs of the work being done. This was used both as a way to ensure reproducibility of the findings from the exploratory work, as well as to track the thought process during difficult tasks in order to have a form of reference material for the report writing. These were often also written in the form of tutorials intended for others at our skill level, as this exercise was seen as a good way to evaluate if the understanding that had been gained was factually correct. See appendix G for examples of these.

4.1.3 User testing

The team had intended to perform user testing of the products developed in this project. The web application was to be tested both in small scale test, where the focus would be on gaining feedback from super user and administrators regarding the functionality of the features of the frontend and farm hardware systems. Large scale user test were also intended of the frontend system, here using larger groups of people from both the basic user and advanced user groups. This was intended both to be used as stresstesting of the system, as well as a comparison to the results found in the student assignment reports. Neither of these tests were performed, as the system never reached a state where these kinds of tests were possible.

4.2 Technical results

4.2.1 Raspberry Pi management

Netboot

If there is an available port in the PoE+ network switch, a raspberry pi may be plugged in and have Raspberry Pi OS automatically installed. At first, ability to provision both Raspberry Pi OS and OctoPi was planned. However, due to complexity, it was later decided to only provision Raspberry Pi OS.

Adding a Raspberry Pi as worker or server

After netbooting, the raspberry pi should have been assigned a k3s worker or server role. However, the team did not have time to investigate this.

4.2.2 Infrastructure scalability

One of the goals was making the backend scalable in terms of future expansion of the printer farm, and integration with other projects at the Manulab.

Service deployment with kubernetes

Kubernetes has been installed on Raspberry Pis. Additionally, services such as the frontend and backend may be installed on the Raspberry Pis by creating A YAML definition for a deployment, and applying it through the command `'kubectl apply -f <definition>.yaml'`. The services may replicate to multiple instances across several Raspberry Pis.

Exposing services with Traefik

However, these services are not exposed to the internet. Traefik requires configuration for exposing Kubernetes services. Although the documentation was read and a team member tried to solve this, the solution to this was not found.

4.2.3 Frontend

The web frontend is not yet available on <https://printman.manulab.net>. A wildcard certificate must first be generated with Certbot, then the certificate must be added as a Kubernetes secret. Upon entering the site, dummy data is presented. When pressing one of the cards, graphs and logs related to the print job should have been presented, but only dummy graphs are presented. When dragging and dropping a 3D model onto the drop zone, a modal is presented. Settings such as filament type and color cannot be adjusted. The job cannot be started.

4.2.4 Backend

Dummy data is generated and sent to the frontend. The backend can successfully connect to one or more OctoPrint instances, however the data is not aggregated. The camera feed from a printer is not mediated to the frontend.

4.2.5 Storage

A hard drive bay with five hard disk drives is connected to one of the Raspberry Pis. However, with the decision to drop the software storage solution, it is currently not in use.

Statistics

Capturing statistics would be used for analyzing and machine learning in the future. However, later the team did not want to dictate which software solution to use for storing the statistics data

4.2.6 MQTT

Initially, it was decided to run MQTT services on the cluster. Since the scope of the project was growing beyond what the time limit allowed, this requirement was dropped.

4.2.7 Printers, OctoPi and OctoPrint

Printer interface

OctoPrint successfully connects to a printer. When assigned a job, the printer prints. The camera shows a live feed from a designated URL.

MQTT from OctoPrint

Before deciding to drop the storage requirement, the MQTT plugin was installed on OctoPrint. To monitor the MQTT traffic, MQTT-Explorer was installed. However, due to difficulty configuring Traefik to expose MQTT endpoints, the MQTT traffic could not be monitored.

4.3 Administrative results

4.3.1 Scrum

The team was largely inexperienced with the use of SCRUM methods in project organisation. The team used the tools provided Confluence for documentation of requirements, design decisions and sprint retrospectives. Jira was used for issue tracking, where requirements from Confluence were created as user stories or tasks.

Sprint planning, sprint review, sprint retrospective were used the first four weeks. By the fifth week, sprint planning was not done. This period coincides with the research of Kubernetes and netbooting, and a period of high workload in other study activities that ran parallel to the project.

Roles

As the team only consisted of two people, the traditional role distribution used in SCRUM was not used. Both members functioned as developer, SCRUM master and product owner at all times, and decisions that would be made by these roles were made jointly.

4.3.2 Distribution of work

The project was split into three parts. First part was management of networking, provisioning Raspberry Pi OS to Raspberry Pis, and configuring OctoPrint with correct plugins. The second part was managing Kubernetes installation and configuration, and service deployments. The third part was developing the frontend and backend services.

The first and second part was wholly delegated to separate members, while the third part was shared between both people.

5 Discussion

5.0.1 Printers

Although the prototype was not developed to the extent that real 3D printers were tested, the 3D printers proved sufficient when the prototype eventually is developed enough. Although nozzles were clogged from time to time, the teaching assistants quickly fixed the problem when it happened.

5.0.2 Raspberry Pi

Because the project did not require hardware resources outside of merely hosting web services, Raspberry Pis were sufficient for prototyping purposes. The Idea Lab were already supplied with 15 Raspberry Pis.

5.0.3 Kubernetes

The decision to use Kubernetes is due to the goal of high availability. After careful considerations between different implementations of Kubernetes such as MicroK8s, Minikube, standard Kubernetes and K3s, the team found k3s to be the most viable alternative. This is partly because k3s has been recognized as a CNCF sandbox project [13], and because it had favorable features such as embedded etcd for cluster configuration. Although the choice to use Kubernetes was founded on solving a real problem, it severely impacted the resulting work and thesis. Seven weeks were spent learning Kubernetes, in which no other value was added to the project in that time frame. In hindsight, adding Kubernetes retroactively when a working MVP was implemented and outside the boundary of the bachelor thesis would have saved us from being stuck in research phase for so long.

HTTPS certificate with Let's Encrypt

Hosting a service over HTTPS on Kubernetes requires the certificate to be stored as a Kubernetes secret. Because all computers reject certificates not signed by a trusted root certificate authority, a third party CA must sign the certificate. The Let's Encrypt service can provide a signed certificate that all computers trust. The team did not have the time to generate and test such a certificate.

5.0.4 Frontend

NextJS

NextJS' nested folder structure for subpages was a good addition for the project. It is easy to understand the relationship between the pages without reading the code.

WebSocket

WebSocket was a good fit for sending printer data to the frontend. Although the team did not have any experience with Socket.IO, it seems easy to swap the plain WebSocket implementation to Socket.IO both for the frontend and backend.

5.0.5 Turborepo

Turborepo was easy to understand from their developer documentation. Running projects in parallel proved to be as easy as running a command.

NPM versus Yarn

Using NPM, local module resolution was a problem. The tsconfig.json file which resides in the packages/tsconfig folder could not be correctly imported. However, by switching to Yarn v1, this was not a problem. Future versions of Turborepo might solve outstanding issues relating to package managers.

Turborepo pruning

Pruning with Turborepo worked well. After pruning the web service for example, a dockerfile could package the pruned files into a complete image ready to deploy to the cluster.

5.0.6 Kafka

Using kafka in the project was scrapped for the same reason MQTT was scrapped. The team did not want to impose a storage solution on the master students who will eventually work on the statistics analysis.

5.1 Results

The product resulting from this project is far from complete. A simple prototype of the web interface for the frontend was developed early in the project, intended to be an MVP that could be used to show the product customer. The project itself hit a standstill around early april. Around this time, the team decided to add several requirements to the project, pertaining to the server infrastructure design.

Among these were to move the server hardware system, which had previously been planned to be a simple desktop computer, into a cluster of Raspberry Pi's. The team believed this was a good way to fulfill the requirements that had been stated regarding the foundation that the project was supposed to lay for future projects at MANULAB. The team decided to implement the server architecture as a Kubernetes cluster, and at the same time that the Raspberry Pi's being used should not boot from SD cards due to their unreliable nature. It was believed that netbooting the Pi's, both those used in the cluster, and those connected to the printers, would provide a more robust system by eliminating this possible source of hardware failure. The decision to develop a clustered server infrastructure came from the goal that the server should ideally be robust enough to handle the web traffic that the webapplication could be expected to handle. As a single desktop computer as server would have resulted in a single point of failure which could bring down the entire system, a cluster was considered to be both more robust, as well as providing options for horizontal scaling to meet future compute hardware requirements.

The server that was designed to provide the netbooting services for the cluster and printers proved a great challenge to implement. It was decided, for reasons of fault tolerance of the OS data of these systems, that ZFS should be set up using an external USB chassis with 5 hard drives. Many issues sprang from this part of the project, seemingly from untraceable and unsolvable hardware issues with the USB chassis itself. Great amounts of time was spent trying to debug and implement this system, but no conclusion was made about the source of the problem.

Configuring the operating systems for the raspberry Pi's also proved challenging, and work here was affected by the unreliability discovered in the server storage devices. A netbooting proof of concept was achieved eventually, but much vast amounts of time was spent doing this. Netbooting of Raspberry Pi's works well, but is a fairly new concept. Many confusing sources on the subject were investigated, many of which lead no closer to a working result. Among these sources were the official documentation of the Raspberry Pi itself. Results were only achieved here after great effort to assemble a working solution by combining and testing claims made by several sources, and this trial-and-error approach to problem solving proved too costly. The team should have accepted that this problem was outside of the scope of the project at an earlier stage, and should have adapted accordingly.

Further challenges in the way of achieveing the desired results came from the Kubernetes cluster itself. This proved to be a challenging technology system to understand, and the team had no previous experience with Kubernetes, and only a little experience with Docker containers. This is again a point that the team should have realised and acted upon much earlier, and shift the focus of the project back to the main requirements.

The sum of all these challenges is that there is no functional product as the result of this project. Many problems were solved regarding Kubernetes, linux system administrtion, network configuration and netboot OS provisioning, but ultimately there were too many issues left with no time to solve them.

6 Conclusion and recommendations further work

6.1 Conclusion

Regretfully, few conclusions can be drawn from the results of this project. Most of the planned features and requirements are only in a partial state of completion, and as such there has been little opportunity to put the criteria we have defined in user stories and our plans for user tests through any form of evaluation process.

The few conclusions the team feel comfortable making are mostly in regard to project planning and methodology. The team unequivocally believes that the poor, and in some cases non-existent, results of this project are fully due to their own failings in evaluating the workloads certain design choices and requirements brought to the project.

The team members started off working in a structured manner, with frequent sprint planning, story point poker for estimating workloads, and retrospectives. However, this structure became fleeting once the team started researching Kubernetes and Netbooting. Entire sprints were devoted to only research, and the lack of changing tasks made the members complacent to the fact that the research task we planned to spend a week on were still the main focus week after week.

Both members agree that the workloads, both in the project itself and in other studies parallel to the project, and lack of tangible progress in this work lead to a feeling of low morale and demotivation, and that these were felt seemingly as personal failures on our part. Neither member voiced this opinion at the time, but reflection and dialogue at the end of the project has revealed that these feelings were harbored by both, in increasing fashion, as the project dragged on.

As a symptom of this, the members were less and less active in the requisite tasks on Jira and Confluence, where timekeeping and work logging also suffered. Documentation of the system also was negatively affected by this. Blog posts were posted in the beginning, however as the team drowned in documentation and cooperated less, we did not hold each other as accountable as we should have.

The requirements the team set that resulted in the greatest workloads were all associated with the design and implementation of the server infrastructure. These were all tasks chosen by the team because of their deep interest in the subjects, and they were seen as great opportunities to dive into systems and technologies that had not been extensively covered by their curriculum of their studies.

6.1.1 Kubernetes

Researching kubernetes took a substantial amount of time. Instead of adding Kubernetes right away, implementing a fully working MVP and basing the thesis on that would have saved us a lot of time. The drawback of not using Kubernetes would be a single-node failure leading to downtime.

Traefik

The services on the cluster can be deployed. However, Traefik configuration for exposing the deployments do not work. See section F.1 for the definitions pertaining to the deployments of the frontend and backend services.

Automatic k3s setup

The Raspberry Pis currently can be netbooted. However, k3s is not automatically installed after first boot. To install k3s, an init script must run. Two OS images must be made, one for k3s server and another for k3s worker.

6.1.2 Raspberry Pi

The Raspberry Pis are relatively cheap and has adequate performance for hosting web services. This makes them great computers for the cluster. However, the work involved in setting up over a dozen linux systems for netbooting was greatly underestimated. The vastness of this task was not apparent at first, and as more hours were poured into it, a certain form of sunk-cost fallacy set in.

In all these cases, the team agrees that a more critical stance on their relevance to the project and product should have been taken much earlier on. The team believes it is not inconceivable that a fully functional farm management system could have been developed during this project, had priorities been chosen differently.

6.1.3 Further work

While the team has doubts regarding the usefulness of their work as foundation for further work related to the system, we have identified several possible features and systems that could be implemented

Storage

Eventually, master students will work on the storage system. To store statistics, a Pub/Sub protocol such as MQTT is favorable.

7 Effects on society

Bibliography

- [1] MANULAB. (). 'Manulab', [Online]. Available: <https://manulab.org/> (visited on 2022).
- [2] NTNU. (). 'Manulab', [Online]. Available: <https://www.ntnu.no/ivb/manulab> (visited on 2022).
- [3] Prusa Research. (). 'Prusa pro afs', [Online]. Available: <https://expo.prusa3d.com/>.
- [4] Mozilla. (2022). 'Html tag reference', [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element> (visited on 12th Dec. 2021).
- [5] Git. (2022). 'About', [Online]. Available: <https://git-scm.com/about> (visited on 12th Dec. 2021).
- [6] 802.11 WG - Wireless LAN Working Group. (2021). 'Ieee 802.11ax', [Online]. Available: <https://standards.ieee.org/ieee/802.11ax/7180/> (visited on 6th May 2022).
- [7] Schwaber, Ken and Jeff Sutherland. (). 'The scrum guide', [Online]. Available: <https://scrumguides.org/scrum-guide.html#purpose-of-the-scrum-guide> (visited on 2020).
- [8] Visual Paradigm. (). 'User story vs use case', [Online]. Available: <https://www.visual-paradigm.com/guide/agile-software-development/user-story-vs-use-case/> (visited on 12th Dec. 2021).
- [9] A. Hevner, A. R, S. March, S. T, Park, J. Park, Ram and Sudha, 'Design science in information systems research', *Management Information Systems Quarterly*, vol. 28, pp. 75–, Mar. 2004.
- [10] J. Iivari, 'A paradigmatic analysis of information systems as a design science', *Scandinavian Journal of Information Systems*, vol. 19, pp. 39–, Jan. 2007.
- [11] A. Hevner, 'A three cycle view of design science research', *Scandinavian Journal of Information Systems*, vol. 19, Jan. 2007.
- [12] Raspberry Pi. (). 'Supply chain, shortages, and our first-ever price increase', [Online]. Available: <https://www.raspberrypi.com/news/supply-chain-shortages-and-our-first-ever-price-increase/> (visited on 20th Oct. 2021).
- [13] CNCF. (). 'K3s graduated to cncf sandbox project', [Online]. Available: <https://www.cncf.io/projects/k3s/> (visited on 19th Aug. 2020).

Appendix

A Pre-project plan

Gruppe 6

**PrintMan
Forprosjektplan**

Versjon 0.1

Gruppe 6

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
17/01/22	0.1	Utkast før første møte med oppdragsgiver og veileder.	Olav og Simen
11/05/22	1.0	Utfylling av rapport basert på tilbakemelding fra veileder. Noe rettskriving.	Olav og Simen

Gruppe 6

Innholdsfortegnelse

1. Mål og rammer	4
1.1 Orientering	4
1.2 Problemstilling / prosjektbeskrivelse og resultatmål	4
1.3 Effektmål	4
1.4 Rammer	4
2. Organisering	4
3. Gjennomføring	4
3.1. Hovedaktiviteter	4
3.2. Milepæler	4
4. Oppfølging og kvalitetssikring	4
4.1 Kvalitetssikring	4
4.2 Rapportering	4
5. Risikovurdering	4
6. Vedlegg	5
6.1 Tidsplan	5
6.2 Adresseliste	5
6.3 Avtaledokumenter	5
6.3.1 Arbeidskontrakt for bachelor-gruppen	5
6.3.2 3-partsavtale	5

Gruppe 6

1. Mål og rammer

1.1 Orientering

Olav ble i høsten 2021 ansatt som studentassistent på på Manulab/IHB. Arbeidsoppgavene gikk ut på å utrede potensiale for videreutvikling av en prototype av Printer Manager, som hadde blitt utviklet under RFF-prosjektet (Regionalt Forskningsfond) *Design for Flexible Manufacturing* i sommeren 2021. Under utredningen av IHBs ønsker for videreutviklingen av systemet ble det fastslått at prosjektets omfang var av slik grad at det passet som et bachelorprosjekt for 2-3 studenter.

1.2 Problemstilling / prosjektbeskrivelse og resultatmål

Problemstilling:

Utvikle et system for bruk og administrering av 3D-printerfarmen ved IHB, NTNU Ålesund. Brukerportal skal gjøre daglig bruk av printere mer brukervennlig for studenter. En administratorportal skal gi ansatte ved Manulab (studentassistenter, lærere, driftspersonale) verktøy som bidrar til enklere vedlikehold og overvåkning av printerfarmen. Systemets backend skal være skalerbar mht. framtidig utviding av printerfarmens størrelse, og kunne integreres med andre prosjekter ved Manulab.

Resultatmål:

- Brukerportal for studenter, for bruk av 3D printere ved Manulab.
- Adminportal for ansatte ved manulabb/instituttet, for overvåkning og vedlikehold av printer farm.
- Backend som kan utvides og være grunnmur for fremtidige prosjekter ved Manulab Ålesund.
- Detaljert dokumentasjon og wiki for API og systemarkitektur.

1.3 Effektmål

Mål for gruppa:

- Tilegne seg mer kunnskap om systemadministrasjon
 - Linux, OS-provisjonering
 - kubernetes/docker
 - cluster-arkitektur for server hardware
 - nettverksadministrasjon
- Fordype kunnskap om 3D printing.
- Levere et produkt som studenter og ansatte ved MANULAB drar nytte av.

Mål for oppdragsgiver:

- Økt brukervennlighet av printer farm
- Grunnmur for utvikling av videre prosjekter

1.4 Rammer

Rom:

- Dedikerte arbeidsplasser på L101
 - Skrivebord satt opp med docking for privat laptop/PC.

Utstyr:

Gruppe 6

- En dedikert prototype av printer rack til prosjektet:
 - 3D printere m/ Raspberry Pi og OctoPi
 - Nettverksutstyr
 - Mobilbredbåndboks
 - Udo Bolt Mini PC som backend server

Tid:

- Prosjektstart 10/01/22
- Prosjektslutt ca. 20/05/22
- Deadline for sluttrapport 20/5/22

2. Organisering

Hvilke aktører er involvert i prosjektet.

Utviklere:

- Olav Valle
- Simen Nesse Wiik

Veileder:

- Kjell Inge Tomren

Oppdragsgiver:

- IHB/Manulab Ålesund
 - Kontaktperson: Paul Steffen Kleppe

Gruppe 6

3. Gjennomføring

3.1. Hovedaktiviteter

Oppstilling av hovedaktiviteter.

Oppgavenavn	Hva gjøres	Hvem gjør det	Hvorfor	Hvordan	Når	Forutsetninger	Dokumentasjon
Forprosjektrapport	Denne malen fylles ut	Olav og Simen	Obligatorisk innlevering	Utarbeides i fellesskap med oppdragsgivere	Deadline 28/01/22	Møte med oppdragsgiver og veileder	Dette dokumentet og møteteferat.
Utviklermiljø	Containerisering av utviklermiljø for frontend og backend. CI/CD og GitHub actions for repo. Automatisering av tester.	Olav og Simen	For å lette skalering og deployment av kode. Lettere samarbeid på kodebasen. Forenkle fremtidige prosjekter.	Docker og muligens Kubernetes	Første sprint.	Ingen.	Dokumentasjon i wiki.
Frontend UI	Grensesnitt for studenter og ansatte for å overvåke og sette i gang printjobber	Olav og Simen	Main feature til produktet	React og Typescript	Førsteprioritet for sprint 2 og videre.	Forprosjektrapport.	Rapporter fra sprint retrospectives og reviews. Dokumentasjon av kodebase og API i wiki.
Web server frontend	Servere web frontend. Brukerlogin.	Olav og Simen	Main feature til produktet	Node, Next/React, Express, websocket. Nginx.	Andreprioritet etter front end MVP.	MVP frontend. Hardware for webserver	Rapporter fra sprint retrospectives og

Gruppe 6

Oppgavenavn	Hva gjøres	Hvem gjør det	Hvorfor	Hvordan	Når	Forutsetninger	Dokumentasjon
	Filtere data fra backend etter brukerens use case (student eller admin). Håndtere input fra bruker (filer, kommandoer).					prototype. Nettverkløsning (mobilbredbånd med statisk IP).	reviews. Dokumentasjon av kodebase og API i wiki.
Printer Manager Backend	Automatisk administrering av OctoPrint instanser på lokalt nettverk. Videresending av kommunikasjon mellom OP og web server.	Olav og Simen	Main feature til produktet. For skalerbarhet, så må det være en form for automatisk oppdagelse av raspberry pi når man kobler en ny til nettverket	PXE boot RPI og å starte en kubernetes pod på disse RPI'ene.	Etter web server prototype.	Printer Rack Prototype. En virtuell prototype kan brukes tidlig i prosjektet.	Rapporter fra sprint retrospectives og reviews. Dokumentasjon av kodebase og API i wiki.
Brukertesting	Den fullstendige løsningen, fra login til å starte og overvåke en printjobb blir testet for brukervennlighet	Olav og Simen organiserer testsesjoner. Oppdragsgiver rekrutterer brukertestere fra instituttet.	Løsningen er ment for en målgruppe. Hvis løsningen ikke fungerer for målgruppen, har løsningen lite verdi.	Sanke inn personer som er villige til å teste løsningen	Mot slutten av frontend UI utviklingsfasen	Frontend MVP.	Brukertest referat i Confluence
Printer Rack Prototype	Rack med 3D printere,	Personale ved manulab og	Main feature til produktet	<Paul Steffen?>	Parallelt med prosjektet.	<???	Diagram i confluence,

Gruppe 6

Oppgavenavn	Hva gjøres	Hvem gjør det	Hvorfor	Hvordan	Når	Forutsetninger	Dokumentasjon
	Raspberry Pi's og nettverksutstyr bygges.	driftavdeling NTNU Ålesund. Olav og Simen med innspill i design og utføring.					brukermanual på hvordan koble ny RPi til skriver

Gruppe 6

3.2. Milepæler

Opplisting av kritiske datoer:

- Forprosjektplan:
 - Deadline for levering 28/01/22
 - Deadline for tilbakemelding fra veileder: 04/02/22
- Utviklerne jobber i 1 ukes lange sprinter
 - Sprintplanlegging på mandager
 - Retrospektiv og review innad i gruppen på fredager
 - Sprintmøter med veileder og oppdragsgiver annenhver uke:
 - uke 4, 6, 8, 10, 12, 14, 17, 19
 - Møtene legges til fredager, så langt det er mulig
 - Review av de to foregående sprintene: resultater, veien videre
- Sluttrapport:
 - Deadline for utkast 06/05/22
 - Deadline for tilbakemelding på utkast 11/05/22
 - Deadline for endelig levering 20/05/22

4. Oppfølging og kvalitetssikring

4.1 Kvalitetssikring

Tiltak for kvalitetssikring på sluttprodukt:

- Gruppens hovedrapport samskrives av begge medlemmene. Veileder gir tilbakemelding på utkast før endelig innlevering.
- Kravspesifikasjonen til produktet utarbeides mellom gruppen og oppdragsgiver.
 - Oppdragsgiver gjør vurdering av utført arbeid gjennom demonstrasjoner på sprintmøter annenhver uke.
 - Prosjektets effektmål og kravspesifikasjon kan oppdateres gjennom samtale mellom partene.
- Brukertesting og stresstesting med personer fra målgruppen.

4.2 Rapportering

All rapportering og loggføring gjøres gjennom Jira og Confluence. Oppdragsgiver og veileder skal ha full tilgang og innsikt i disse verktøyene.

Rapporteringsformer:

- Rapporter fra sprint review og retrospective
 - Gjennomgang med oppdragsgiver og veileder på møter, annenhver uke.
- Timeliste og arbeidslogg
 - Føres fortløpende som del av issue tracking og sprintplanlegging i Jira.

Gruppe 6

5. Risikovurdering

Risikoanalyse som vurderer sårbarheter i prosjektet (hendelse, sannsynlighet, konsekvens og tiltak).

Hendelse	Sannsynlighet	Konsekvens	Tiltak
Mangel på kompetanse blant studentene.	Lav	Høy. Sluttprodukt oppfyller ikke kravspesifikasjonen.	Nedskalering av prosjektmål underveis.
Nedstenging av campus/manulab pga. Covid-situasjonen.	Lav	Høy. Utviklingsarbeidet er til dels avhengig av tilgang til hardware (printer rack, servere), noe som kan medføre utfordringer i prosessen. Nedstenging av manulab vil føre til utfordringer knyttet til brukertesting av systemene.	Digitalisering/virtualisering av printer rack prototype kan kanskje erstatte fysisk hardware for utvikling og testing. Møter og samarbeid kan kjøres digitalt. Løsninger for digital brukertesting må utforskes. Større vekt på automatisert testing.
Alvorlig sykdom blant gruppe medlemmene / privatlivskriser.	Lav	Middels. Stor reduksjon i arbeidskapasiteten til gruppen.	Nedskalering av prosjektmål.
Feilestimering av gruppens arbeidskapasitet i individuelle sprinter	Middels (opp mot høy, i start av prosjektet)	Lav. Gruppen treffer ikke planlagte mål for en sprint.	Kritisk vurdering i sprint retrospektiv. Resterende arbeid blir førsteprioritet i neste sprint. Være mer konservativ på estimat av arbeidsmengden i oppgaver med mange ukjente faktorer.
Forsinkelse/vansker med levering på nødvendig hardware	Middels	Lav	Ved mangel på Raspberry Pi hardware vil prosjektet nedskaleres for å tilpasse tilgjengelige ressurser (f.eks. ett enkelt printer-rack

Gruppe 6

			istedenfor hele farmen). Ved forsinkelse i levering av annen nødvendig hardware, vil prosjektets fremgangsplan måtte omstilles for å tilpasse dette.
--	--	--	--

6. Vedlegg

Følgende dokumenter leveres som separate filer ved innlevering i Blackboard i januar (obligatorisk arbeidskrav), men ikke i endelige leveransen av hovedrapporten den 20. mai!

6.1 Tidsplan

Tar utgangspunkt i Tidsplan v1.0 fra blackboard. Mer detaljert tidsplan for sprinter og milepæler fastsettes i Confluence og Jira, etterhvert som kravspesifikasjon og arbeidsoppgaver blir utredet og oppdatert.

6.2 Adresseliste

Navn, firma, tlf., epost, adresse

Navn	Rolle	Institutt ved NTNU	Telefon	Epost
Olav Valle	Utvikler	IIR, NTNU Ålesund	900 10 510	olavval@ntnu.no
Simen Nesse Wiik	Utvikler	IIR, NTNU Ålesund	412 26 050	simenwii@ntnu.no
Paul Steffen Kleppe	Produkteier	IHB, NTNU Ålesund	415 26 987	paul.s.kleppe@ntnu.no
Irina-Emily Hansen	Produkteier	IHB, NTNU Ålesund	473 27 049	irina-emily.hansen@ntnu.no
Kjell Ingen Tomren	Veileder	IIR, NTNU Ålesund	986 92 297	kjell.i.tomren@ntnu.no

6.3 Avtaledokumenter

6.3.1 Arbeidskontrakt for bachelor-gruppen

Avtale for å definere/ beskrive hvordan bachelor-gruppen skal samarbeide gjennom prosjektet. Mal tilgjengelig på læringsplattformen.

Arbeidskontrakt for bachelorprosjekt 3D Printer Manager

Medlemmer: Olav Valle, Simen Nesse Wiik

Innledende tekst

Denne arbeidskontrakten bygger på et sett med typiske mål, oppgavefordelinger, prosedyrer og retningslinjer for interaksjoner for studentarbeider. Arbeidskontrakten er utfylt med *egne* fortolkninger av hva man mener med disse og hvordan man skal oppnå dette.

Det legges til eller fjernes punkter etter egen vurdering for tilpassing til oppgaven.

Roller og oppgavefordeling (Hvordan organiserer man arbeidet?)

Hvilke roller/ ansvarsområder er formålstjenlig for samarbeidet i prosjekt-gruppen?

- Product owner: Oppdragsgiver. Endelig avgjørelse på all kvalitetskontroll og kravene til sluttproduktet.
- Teamledelse: Sprint owner byttes på mellom sprinter. Sprint owner er "gruppeleder" for den sprinten, og rollen vil fylles av den på gruppen som har mest erfaring med, eller største planer for sprintens tema.
- Dokumentansvarlig: Olav. Fører referat, sender møteinnkalling. Sprintrapporter og hovedrapporten skrives i fellesskap.
- Kvalitetssikring: Begge. Sprint owner vil ha hovedansvar for leveransen for sprintperioden. Product owner har endelig avgjørelse på vurdering av kvalitet.

Prosedyrer (hvordan gjør man ting?)

- A. Møteinnkalling
 - a. Daglig standup i gruppen. Først og fremst fysisk, eller digitalt hvis nødvendig. Ingen innkalling. Daglig rutine kl 10 00.
 - b. Sprint møter etter hver 2-ukers sprint. Møteinnkalling på epost, til oppdragsgiver og veileder.
- B. Varsling ved fravær eller andre hendelser
 - a. Direktemelding i gruppechat. Dokumenteres i arbeidslogg.
- C. Dokumenthåndtering
 - a. Sluttrapport: Overleaf. Digital samskriving.
 - b. Møtereferat: Confluence.

Gruppe 6

- c. Sprint review og retrospective: Confluence.
- d. Timeliste: Confluence.
- e. Arbeidslogg: Jira issues (+Confluence?) og GitHub issues/commitmeldinger.

D. Innleveringer av gruppearbeider

- a. Rapporter for sprint reviews og retrospectives i Confluence
 - i. Innledning og mål for sprinten fylles ut ved start av sprintperiode. Resultater for arbeid fylles inn fortløpende i malen. Utkast av rapporter ferdigstilles før sprintmøte med oppdragsgiver.
- b. Arbeidskrav:
 - i. Forprosjektrapport leveres i Blackboard.
 - ii. Sluttrapport leveres i Inspira.

Interaksjon (*Hvordan opptrer man sammen?*)

7 Oppmøte og forberedelse

7.3 Kjernetid med oppmøte på L101 kl. 10:00-16:00, mandag til fredag. Ved behov i helger og på kveldstid.

7.4 Felles forberedelse for sakslisten til møter.

8 Tilstedeværelse og engasjement

8.3 Gruppemedlemmene jobber tett sammen i kjernetiden. Bruker parprogrammering og brainstorming for problemløsning.

9 Hvordan støtte hverandre

9.3 Ikke skyte ned ideer i brainstorming

9.4 Jobbe aktivt sammen, bruke parprogrammering og problemløsning i fellesskap.

10 Uenighet, avtalebrudd

10.3 Uenighet og avtalebrudd håndteres i første omgang gjennom dialog i gruppen. Veileder kalles inn hvis dialogen bryter sammen.

10.4 Akseptabelt avvik:

10.4.1 Forsovelse til oppmøte til kjernetid.

10.4.2 Sykdom.

10.4.3 Avtalt ferie/fravær.

Gruppe 6

10.4.4 3-partsavtale

Det skrives 3-partsavtaler mellom NTNU, oppdragsgiver og studenter. Mal tilgjengelig på læringsplattformen og innsida. (Standardavtale)

Se signert papirversjon.

B Requirement specification

B.1 Contents

1. Introduction 69
2. Use Case diagram 70
3. User Stories 70
4. Domain model 71
5. Wireframes 72

B.2 Introduction

This document details the product requirements as they were defined for the project.

B.3 Use Case diagram

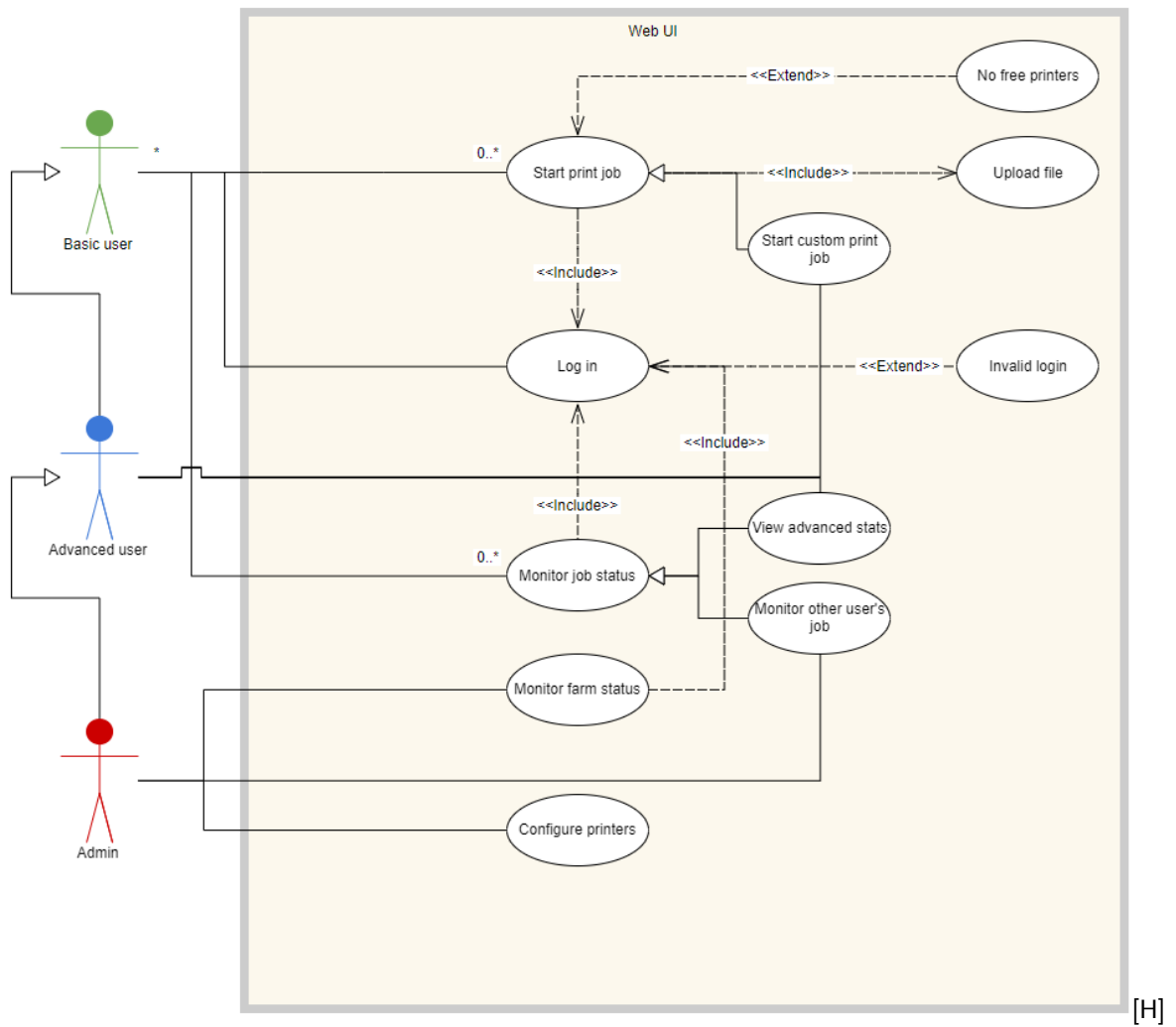


Figure B.1: Use case diagram

B.4 User Stories

See appendix C.1.

B.5 Domain model

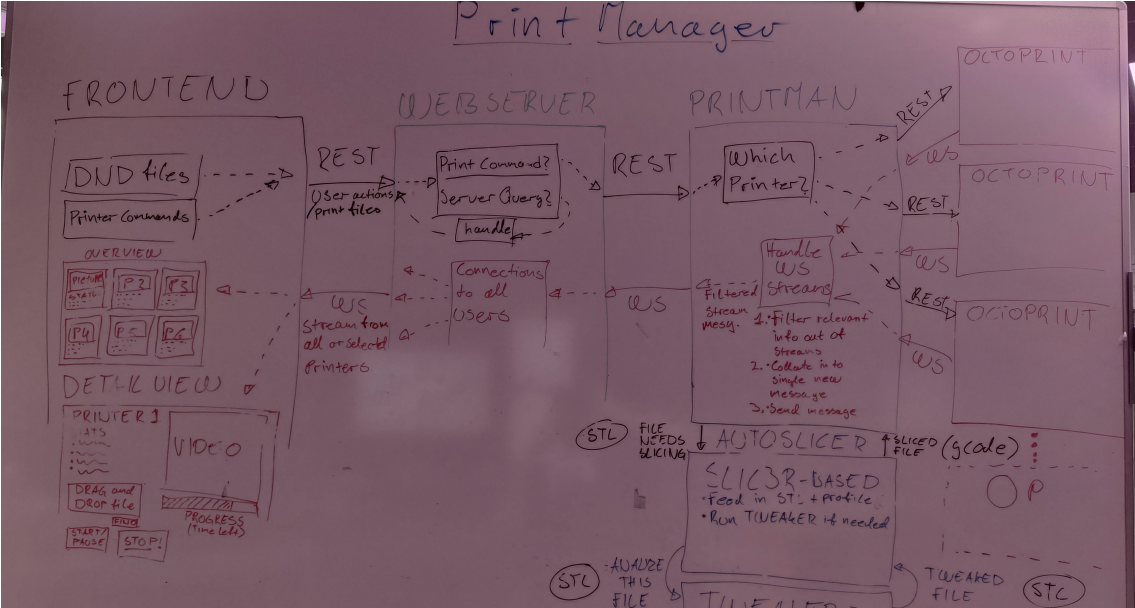


Figure B.2: Domain model diagram.

B.6 Wireframes

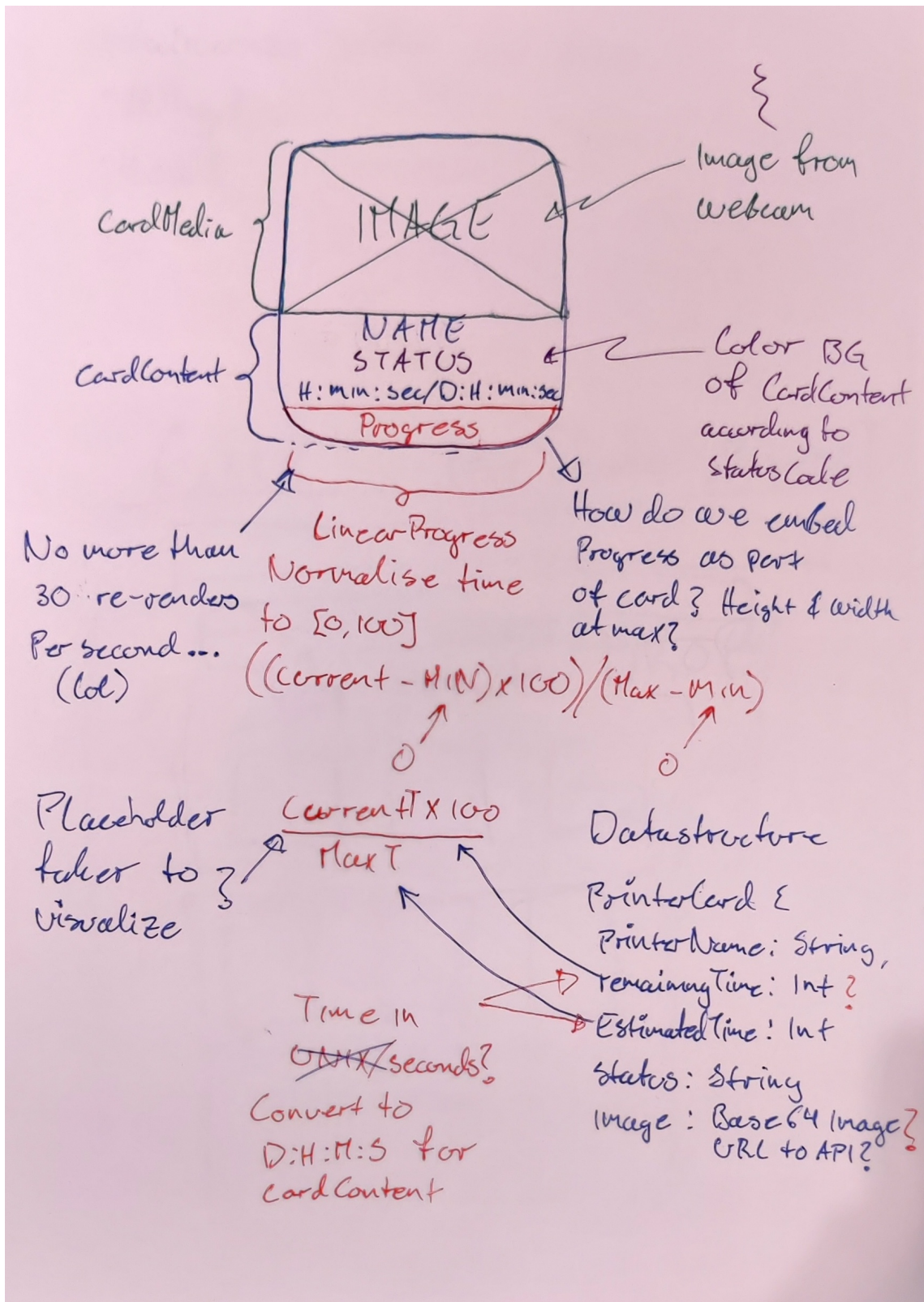


Figure B.3: Wireframe of card module for printer grid.

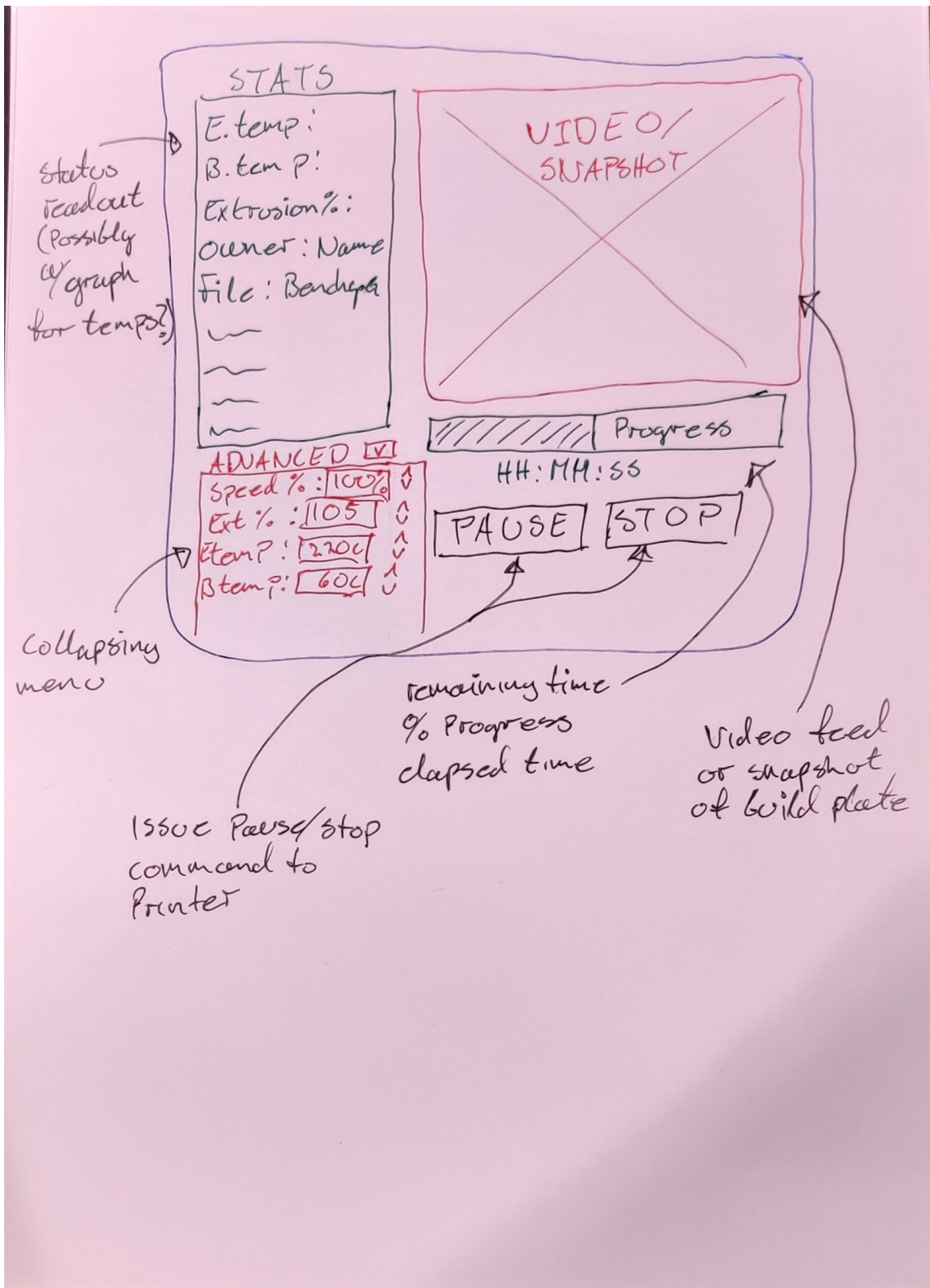


Figure B.4: Wireframe of the detail printer view.

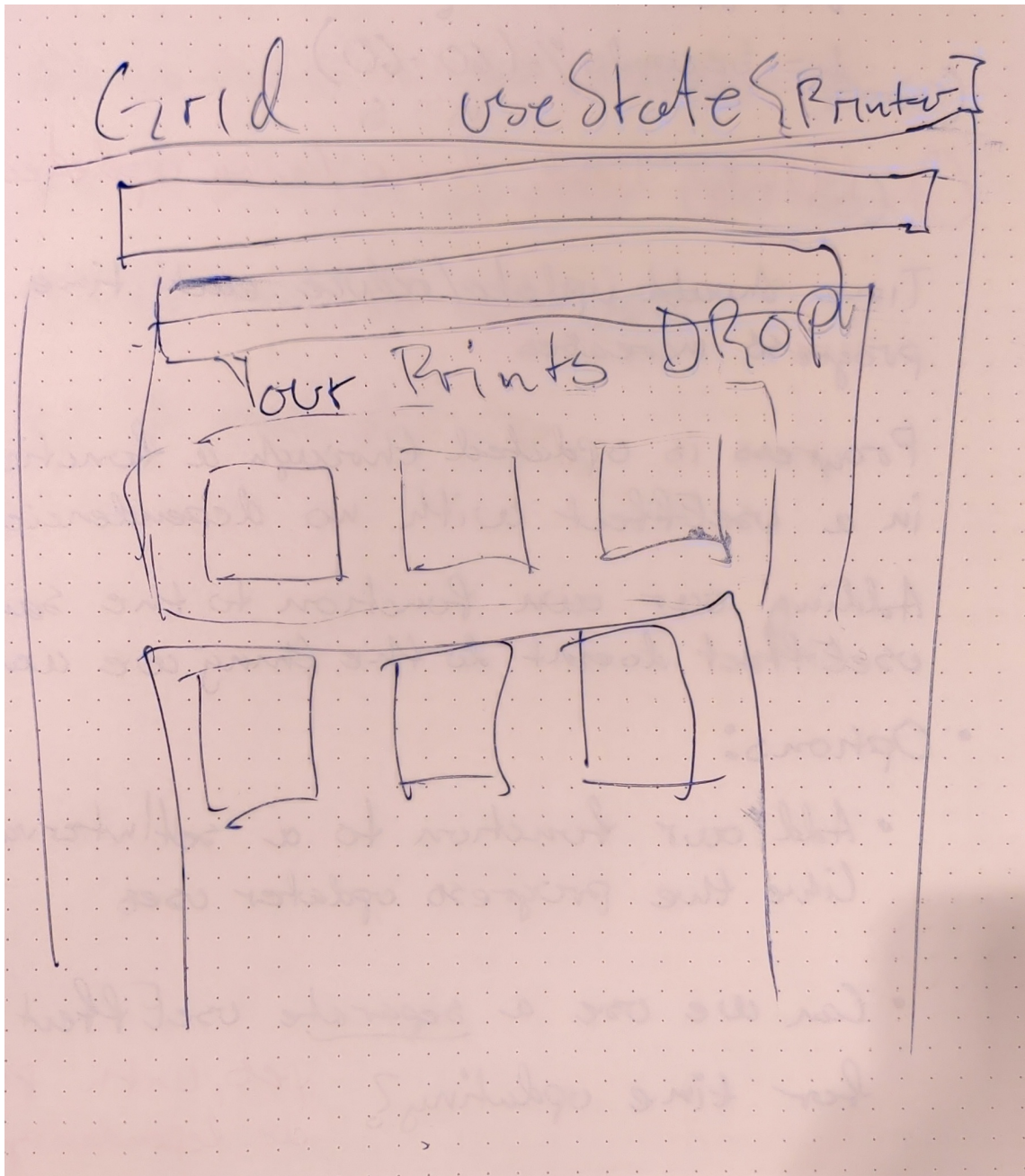


Figure B.5: Wireframe of the printer grid overview page layout.

C Research issues

[PRINTMAN-4] Explore existing implementations and competing products Created: 19/Jan/22 Updated: 26/Jan/22 Resolved: 26/Jan/22	
Status:	Done
Project:	Manulab 3D Printer Manager
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Task	Priority:	Medium
Reporter:	Olav Valle	Assignee:	Olav Valle
Resolution:	Done	Votes:	0
Labels:	None		
Remaining Estimate:	0 minutes		
Time Spent:	3 hours		
Original Estimate:	Not Specified		

Sprint:	PRINTMAN Sprint 0: Pre-project
----------------	--------------------------------

Generated at Fri May 20 08:38:53 CEST 2022 by Olav Valle using Jira 8.22.2#822002-sha1:165c8f51e3b1a891285d611f42f1d6c4389222ad.

[PRINTMAN-12] Why React? Created: 26/Jan/22 Updated: 07/Feb/22 Resolved: 07/Feb/22	
Status:	Done
Project:	Manulab 3D Printer Manager
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Task	Priority:	High
Reporter:	Olav Valle	Assignee:	Unassigned
Resolution:	Done	Votes:	0
Labels:	Documentation, decision-report		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Epic Link:	Decision reports
Sprint:	Sprint 1: Grid view MVP

Description

Contrast and compare React (Next.js) to alternative web frontend frameworks.

Notable alternatives:

- Vue
- Svelte
- Angular
- Next.js or just React?

Comments

Comment by [Olav Valle](#) [26/Jan/22]

React vs Next.js

<https://www.youtube.com/watch?v=6jWWKczzGM0>

Generated at Fri May 20 08:38:33 CEST 2022 by Olav Valle using Jira 8.22.2#822002-sha1:165c8f51e3b1a891285d611f42f1d6c4389222ad.

[PRINTMAN-14] Why Turborepo? Created: 26/Jan/22 Updated: 07/Feb/22 Resolved: 07/Feb/22	
Status:	Done
Project:	Manulab 3D Printer Manager
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Task	Priority:	High
Reporter:	Olav Valle	Assignee:	Unassigned
Resolution:	Done	Votes:	0
Labels:	Documentation, decision-report		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Epic Link:	Decision reports
Sprint:	Sprint 1: Grid view MVP

Description

Compare and contrast turborepo, and the use of a monorepo, to the alternatives

- Why a monorepo?
- NX vs Turborepo

Generated at Fri May 20 08:38:28 CEST 2022 by Olav Valle using Jira 8.22.2#822002-sha1:165c8f51e3b1a891285d611f42f1d6c4389222ad.

[PRINTMAN-15] Why containerised dev environment? Created: 26/Jan/22 Updated: 07/Feb/22 Resolved: 07/Feb/22	
Status:	Done
Project:	Manulab 3D Printer Manager
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Task	Priority:	High
Reporter:	Olav Valle	Assignee:	Unassigned
Resolution:	Done	Votes:	0
Labels:	Documentation, decision-report		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Epic Link:	Decision reports
Sprint:	Sprint 1: Grid view MVP

Description

Compare and contrast pro/con of containerising the dev environment

- Is it common practice?
- Weigh benefits vs cost
- Still feasible/valuable if not using a monorepo?

Generated at Fri May 20 08:38:46 CEST 2022 by Olav Valle using Jira 8.22.2#822002-sha1:165c8f51e3b1a891285d611f42f1d6c4389222ad.

[PRINTMAN-56] OctoPi vs Dockerized OctoPrint Created: 07/Feb/22 Updated: 01/Mar/22	
Status:	To Do
Project:	Manulab 3D Printer Manager
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Task	Priority:	Medium
Reporter:	Olav Valle	Assignee:	Unassigned
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Epic Link:	Decision reports
-------------------	----------------------------------

Generated at Fri May 20 08:38:19 CEST 2022 by Olav Valle using Jira 8.22.2#822002-sha1:165c8f51e3b1a891285d611f42f1d6c4389222ad.

[PRINTMAN-66] Research forking and expanding MQTT plugin Created: 01/Mar/22 Updated: 20/Apr/22	
Status:	To Do
Project:	Manulab 3D Printer Manager
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Task	Priority:	Medium
Reporter:	Olav Valle	Assignee:	Unassigned
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Epic Link:	Backend MVP
Sprint:	Sprint 4: Backend MVP

Description

The MQTT plugin for OP may need to be expanded with custom event hooks, to be able to get the type and format of data that we require.

See OP documentation and MQTT plugin github:

- <https://docs.octoprint.org/en/master/plugins/index.html>
- https://github.com/OctoPrint/OctoPrint-MQTT/blob/master/octoprint_mqtt/_init_.py

Comments

Comment by [Simen Nesse Wiik](#) [20/Apr/22]

Might not be necessary. I have found the root of the problem of not connecting to mosquitto.

These two lines must be present in the configuration file in order to connect to mosquitto

```
listener 1883
```

```
allow_anonymous true
```

However, this opens up for everyone being able to listen to mosquitto's output. So we must not set allow_anonymous, and rather set a password that clients must supply. Maybe it's possible to have some sort of ssh_key equivalent

Generated at Fri May 20 08:38:13 CEST 2022 by Olav Valle using Jira 8.22.2#822002-sha1:165c8f51e3b1a891285d611f42f1d6c4389222ad.

Live data from OctoPrint instances (PRINTMAN-67)

 [PRINTMAN-70] [Research mosquito.conf](#) Created: 01/Mar/22 Updated: 20/Apr/22

Status:	To Do
Project:	Manulab 3D Printer Manager
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	Priority:	Medium
Reporter:	Simen Nesse Wiik	Assignee:	Unassigned
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	1 day		
Original Estimate:	Not Specified		

Sprint:	Sprint 4: Backend MVP
----------------	-----------------------

Description

Do we have to change any default configuration?
<https://mosquitto.org/man/mosquitto-conf-5.html>

Generated at Fri May 20 08:37:47 CEST 2022 by Olav Valle using Jira 8.22.2#822002-sha1:165c8f51e3b1a891285d611f42f1d6c4389222ad.



Why containerised dev environment?

Details

Type: Task Status: **DONE** ([View Workflow](#))
Priority: ^ High Resolution: Done
Labels: Documentation decision-report
Epic Link: Decision reports
Sprint: Sprint 1: Grid view MVP

Description

Compare and contrast pro/con of containerising the dev environment

- Is it common practice?
- Weigh benefits vs cost
- Still feasible/valuable if not using a monorepo?

Attachments

Drop files to attach, or [browse](#).

Tempo



THERE ARE NO ACTIVITIES FOR YOU TO SEE FOR THIS ISSUE.

You have not recorded any activity on this issue.

[Log Time](#)

[Plan Time](#)

[Add Expense](#)

Activity

There are no comments yet on this issue.

People

Assignee:

Unassigned

[Assign to me](#)

Reporter:

Olav Valle

Votes:

0

Watchers:

1 [Stop watching this issue](#)

Dates

Created:

26/Jan/22 10:10 AM

Updated:

07/Feb/22 12:58 PM

Resolved:

07/Feb/22 12:58 PM

Collaborators

Agile


Completed Sprint:

[Sprint 1: Grid view MVP](#) ended 07/Feb/22

[View on Board](#)

C.1 User Stories

Requirement: Admin overview page

Target release	
Epic	 PRINTMAN-32 - Super user printer monitoring and statistics panel TO DO
Document status	DRAFT
Document owner	Olav Valle
Designer	Olav Valle Simen Nesse Wiik
Developers	Olav Valle Simen Nesse Wiik
QA	Olav Valle Simen Nesse Wiik

Goals

- Give administrators a high level overview of all printers and their state
- Provide a log of errors, events and system messages
- Attach notes/memos to printers to record and share issues and events that have been observed

Background and strategic fit

This page will serve as a useful tool to the administrators and technicians tasked with maintaining and servicing the printer farm. It plays a vital part in fulfilling the main goal of the product, viz. to streamline and assist in the running of the server farm.

The requirements detailed here will build a foundation of basic functionality, and is planned to be expanded with more functionality, e.g. data visualization to aid with preventative maintenance, in future projects.

It is therefore important that the features and systems implemented as part of this the work on this requirement will not saddle these future projects with technical debt.

Focus should be placed on developing basic overview/monitoring features, while advanced features (like remote calibration commands, status reports...) should be carefully considered, and only be implemented if a useful and robust solution is likely to result from the work.

Assumptions

Requirements

#	Title	User Story	Importance	Notes
1	Farm overview	As an administrator user, I want to be able to view detailed information about the status and performance of a specific printer, so that I can easily review the state of individual printers remotely.	Must have	<ul style="list-style-type: none">• Highlight printers with issues• Highlight print jobs with non-default settings/configs (high print speed, extrusion %...)
2	Admin commands	As an administrator user, I want to be able to issue advanced commands to a specific printer, so that I can perform supervision and maintenance tasks remotely.	Medium	<ul style="list-style-type: none">• Simplify execution of tasks that can be performed without physical access to printer<ul style="list-style-type: none">• Run calibration tests• Set printers as unavailable/out-of-order
3	Message log	As an administrator user, I want to be able to review a log of messages from OctoPrint and the printer software, so that I can receive important messages and warnings about the state of the printer.	Medium	<ul style="list-style-type: none">• Simplifies task of monitoring the printer farm

4	Statistics	<p>As an administrator user,</p> <p>I want to have access to the statistics of previous print jobs, their results, and any issues that occurred as part of their execution,</p> <p>so that I can evaluate the overall performance and maintenance status of the printers in the farm.</p>	<p>Medium</p> <p>(Low for advanced features)</p>	<ul style="list-style-type: none"> • Many of the more advanced possibilities discussed for this feature will rely on data visualization and analysis, and may be outside the scope of this project. • Care should be taken to not implement systems that will leave future projects with technical debt. Any feature/system that must be removed by future projects will constitute a wasted effort on behalf of both the current and future teams.
---	------------	---	--	---

User interaction and design


Questions

Below is a list of questions to be addressed as a result of this requirements document:

Question	Outcome

Not Doing

Requirement: Hardware infrastructure

Target release	
Epic	 PRINTMAN-61 - Raspberry Pi based MicroK8s cluster TO DO
Document status	DRAFT
Document owner	Olav Valle
Designer	Olav Valle Simen Nesse Wiik
Developers	Olav Valle Simen Nesse Wiik
QA	Olav Valle Simen Nesse Wiik

Goals

- Provide the server hardware for the product
- Serve as an expandable foundation for systems developed in future projects
- Provide a robust hardware solution to avoid single point of failure
 - Computer cluster
 - Avoid using microSD cards for OS installations (unreliable and wear quickly)
- Provide a set of tools to simplify system administration and maintenance tasks

Background and strategic fit

The 3D printer management system being developed will require both a webserver for the frontend, as well as a backend system to orchestrate communications between printers and users. Close physical proximity between the backend's server hardware and the printers is a requirement, for reasons of hardware constraints (wiring and construction of the printer farm and its mobile printer racks), as well as from a maintenance viewpoint (administrators of the system are also repair technicians for the printer hardware).

While a simple desktop computer would suffice for the low traffic and processing load the system is expected to sustain, a single machine also represents a single point of failure for the system. A computer cluster architecture allows for both high availability of services running on the hardware, as well as providing redundancy and robustness to the server itself (since a cluster can continue to function in a reduced state if hardware is damaged). This eliminates, or at least mitigates, the risks presented by a single point of failure. This cluster architecture also allows for scaling the compute power of the server to meet future requirements (by adding more Raspberry Pi's to the cluster).

Another deciding factor is the requirement that the system being developed will form the foundation for future projects at Manulab Ålesund. As this work will inevitably involve research, exploration and testing of prototype machinery and software, having full control of the system hardware is essential for the rapid prototyping approach that students and researchers at Manulab employ. Relying on servers and systems that are administrated by cloud service providers or the IT department of NTNU could present unwanted hurdles in the progress of these future projects.

Assumptions

- Student assistants and teachers at manulab will serve as system administrators and service technicians for both the hardware and software.
 - The computer skill set of these people is expected to at minimum be above average for students at the later stages of their engineering programs.
 - Their demographic will be from machine/mechanical, electronics/automation and computer engineering backgrounds.
 - While students from machine engineering backgrounds are not expected to have extensive programming or system administration experience, their colleagues among the automation and, especially, the computer engineering students are expected to be familiar with both the command line, and server hardware infrastructure (at least those at 2nd/3rd year of a bachelors program).
- As not all students interacting with the system are at the same level of technical computer skill, tools should be developed to simplify and automate regular tasks as much as possible.
- System administration and maintenance will take place exclusively at the premises of Manulab, and remote access to the system is not considered a priority. SSH access to the administration terminal can be established in those cases where solely software/OS level tasks are required.

Requirements

#	Title	User Story	Importance	Notes
---	-------	------------	------------	-------

1	Hardware provisioning	As a system administrator, I want a streamlined method for configuring an operating system for one or more machines, to either replace faulty machines, or to expand the system with new machines.	Must have	<ul style="list-style-type: none"> • A baseline OS (Raspberry Pi OS), configured with as many default settings as possible • A script that automates setup and config of an OS tailored to the required use case, with minimal interaction. <ul style="list-style-type: none"> • Inputs: RPi's serial number, MAC address, and task (cluster node or OctoPrint instance) • Ability to have replacement hardware inherit the role and tasks of the machine it is replacing.
2	Container orchestration	As a system administrator, I want to be able to administrate and set up containerized versions of the services required for the system, so that these services can be orchestrated across the available server hardware.	Must have	<ul style="list-style-type: none"> • Kubernetes based orchestration • Scripts that automate Docker image creation of system services • Scripts/config files for setup of orchestration
3	Documentation	As a system administrator, I want to have documentation that details and explains how to use the scripts and tools provided to me, to make my tasks easier and more understandable.		<ul style="list-style-type: none"> • How-to's for normal tasks • Technical documentation explaining the systems in use, their purpose and design. • Documentation in scripts explaining functionality

User interaction and design

Questions

Below is a list of questions to be addressed as a result of this requirements document:

Question	Outcome
How can we make sysadmin tasks easy and repeatable for inexperienced users?	<ul style="list-style-type: none"> • Make scripts that automate the process. • Document the process well, so that it's understandable to the users (not "black magic"). • Make the system robust, to reduce the frequency of repair/maintenance tasks.

Not Doing

Requirement: Landing page and Printer Overview

Target release	
Epic	 PRINTMAN-19 - Landing page and printer grid view IN PROGRESS
Document status	DRAFT
Document owner	Olav Valle
Designer	Olav Valle Simen Nesse Wiik
Developers	Olav Valle Simen Nesse Wiik
QA	Olav Valle Simen Nesse Wiik

Goals

- Serve as landing page for the Print Manager service
- Allow navigation to other parts of website (admin page, printer detail view, own account page?)
- Provide an overview of all printers in the system

Background and strategic fit

This is the most basic part of the product: a web app that allows users (students and teachers at NTNU) to easily check printer availability, start prints, and monitor the progress of their own prints. Implementation of the "must have" requirements of this page serves as the most fundamental MVP of the final product.

Assumptions

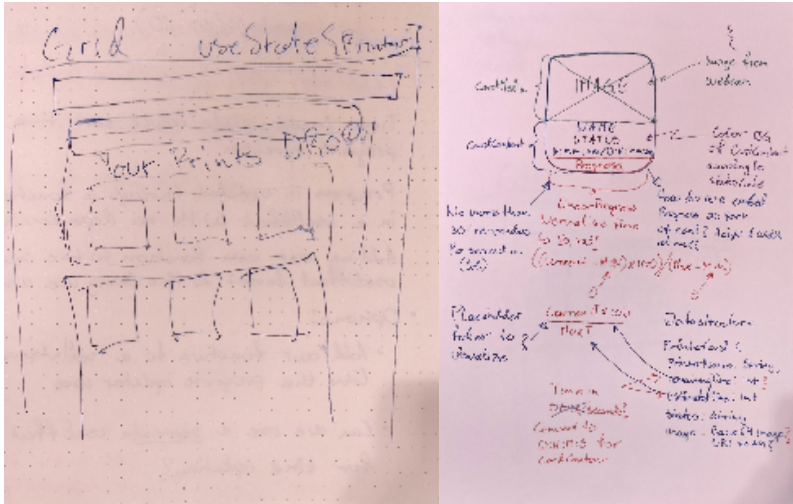
- Users will primarily access this webpage from a laptop or desktop computer, as this is what they use to design their 3D models.
- Some users may wish to access the page from mobile devices, to quickly check the state of a print they started earlier.
- Cases where users may wish to start a print from a mobile device will be a rare exception.
- Users can be split into 3 categories:
 - Basic users: Students requiring the use of the printers for various projects. Technical computer skill is assumed to be average for that of an engineering student. Technical skill with 3D printing technology will vary, from complete beginners (most likely first year students) up to very advanced (pre-/postgrad students, teachers and student assistants).
 - Advanced users: Students and teachers requiring the use of the printers for various projects. Technical computer skill is assumed to be average for that of an engineering student, or possibly higher depending on education level and/or profession. High experience with 3D printer technology, and require greater and more granular control over their print jobs and their current status/progress.
 - Administrators: Teachers and student assistants hired by the department. Serve as service technicians for the printers, administrators for the software system, and provide assistance to other users as needed. High level of technical computer skill. Very high level of experience with 3D printer technology. Require access to detailed reports on the state of all the printers in the system, as well as the ability to control all printers in the system.

Requirements

#	Title	User Story	Importance	Notes
1	Printer overview grid	As a basic user, I want to have a place where I can get an at-a-glance overview of all printers in the farm, so that I can easily see if/how many free printers there are.	Must have	<ul style="list-style-type: none">• Sort printers by status (error, finished, busy, clear)• Sort busy status groups by time remaining• Separate grid of user's own print jobs (hide if none)
2	Availability statusbar	As a basic user, I want a statusbar that shows how many printers are free, or how long until the next printer is free, so that I can know if I am able to start a print job or not.	Must have	<ul style="list-style-type: none">• shows counts for ready, busy, error printers• Shows time until next printer is free

3	Printer card	As a basic user, I want to have a compact view of the most basic info about a printer, so that I can easily identify which printer suits my needs at the moment.	Must have	<ul style="list-style-type: none"> Show snapshot from printer's camera Printer state with semantic colouration Print time/progress bar Click card to reach detail view of printer
4	Printer camera snapshot	As a basic user, I want to be able to see a snapshot from the printer's web camera, so that I can manually check that the print plate is clear before starting a print remotely.	Must have	<ul style="list-style-type: none"> Helps users avoid interacting with printers that are in use with others
5	Drag and Drop to upload files	As a basic user, I want to be able to easily drag and drop in my 3D models and have them start up on an automatically selected, appropriate printer, so that I can have my print jobs started quickly.	Must have	<ul style="list-style-type: none"> Lowers barrier to entry for use of printers
6	Job confirmation dialogue	As a basic user, I want to see a confirmation dialogue that shows a detail view of the printer that my model will be printed on, so that I can manually assert that the printer is working and ready.	Must have	<ul style="list-style-type: none"> Very important to ensure safe operation of printers <ul style="list-style-type: none"> Currently no way to automate printer readiness check Should involve a two-step confirmation
7				
8				

User interaction and design




Questions

Below is a list of questions to be addressed as a result of this requirements document:

Question	Outcome

Not Doing

Requirement: Printer job details view

Target release	
Epic	 PRINTMAN-27 - Basic user single printer detail view TO DO
Document status	DRAFT
Document owner	Olav Valle
Designer	Olav Valle Simen Nesse Wiik
Developers	Olav Valle Simen Nesse Wiik
QA	Olav Valle Simen Nesse Wiik

Goals

- Provide users a way to monitor the status and progress of their printing jobs
- Provide users a way to pause/stop their jobs in case of problems.
- Restrict printer interaction to owner of job, or admin.
- Video stream from printer camera?
- Start print job on specific printer?
- Issue commands to change printing speed, flow rate, extruder/printplate temperature for advanced users.

Background and strategic fit

A detailed view of the printer and the job it is performing will allow users to more closely monitor the progress of their prints. A live camera feed will allow for visual inspection for defects or faults in the print, while stats like temperature and other readouts from the printer hardware will give advanced users better insight into the progress. More advanced users may also wish to access controls and commands that are available through the physical interface of the printer, e.g. to set printing speed, temperature etc.

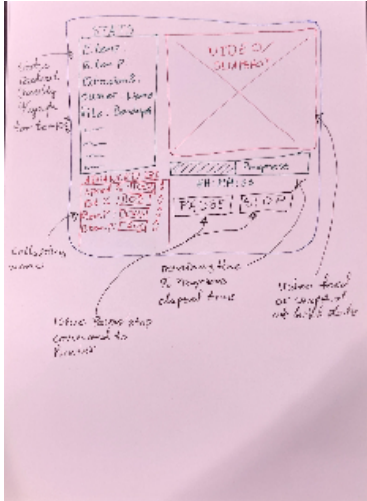
Assumptions

- Most users, basic, advanced and admins, will at some point want to remotely monitor the progress and status of their print jobs.
- Advanced users may wish to tweak settings for their jobs
- Admins may wish to inspect jobs owned by basic and advanced users, and will need the ability to override jobs and/or settings made by these users.

Requirements

#	Title	User Story	Importance	Notes
1	Detailed job progress	As any type of user, I want to be able to see detailed readouts of the status of the printer hardware, and the progress of my print job, to better be able to detect potential issues that have occurred.	Must have	<ul style="list-style-type: none">• Relevant statistics:<ul style="list-style-type: none">• Extruder and bed temperature
2	Print job control	As an advanced user, I want to be able to issue commands to the printer to either pause/stop a job in progress, or to change printer settings like print speed or extrusion level, to have a printing experience as close to physically interacting with the printer as possible.	Important	<ul style="list-style-type: none">• Relevant commands:<ul style="list-style-type: none">• Pause /cancel a job• Set printing speed• Set extrusion %
3	Job access restriction	As an administrator user, I want to users of the system to be restricted to only being able to issue commands for their own jobs, to prevent accidental or malicious tampering with print jobs owned by other users.	Must have	<ul style="list-style-type: none">• Connect job ownership to user accounts• Admins have control over all jobs
4				

User interaction and design



Questions

Below is a list of questions to be addressed as a result of this requirements document:

Question	Outcome
What printer commands does the product owner want to be restricted?	
How much bandwidth would a camera stream from 25-30 printers require? Possibly too much for our system?	

Not Doing

- Auto-slicing is not a priority, but a known solution exists and can be integrated in a future release.
- Starting print job on a specific printer through the details view page is planned for the future.

Requirement: System backend, server, and OctoPrint connection

Target release	
Epic	PRINTMAN-73 - MVP for backend code and logic (Nest.js, Kafka, MQTT) TO DO
Document status	DRAFT
Document owner	Olav Valle Simen Nesse Wiik
Designer	Olav Valle Simen Nesse Wiik
Developers	Simen Nesse Wiik Olav Valle
QA	Olav Valle Simen Nesse Wiik

Goals

- Connect to known OctoPrint instances
 - Collate print job and printer hardware stats over WS or MQTT from OP
 - Issue commands from web UI users to correct OP instance
 - Transfer print job files from frontend user to correct OP instance
- Serve frontend interaction API endpoints over REST
- Serve live data collated from printers to frontend over WS
- User account creation and user authentication for print job actions and ownership
- Send .stl, .3mf, etc. files to the auto slicing system, when/if this is operational.
- Interoperate with a data storage system for statistics tracking of the printer farm.

Background and strategic fit

The backend serves as the interface between the web UI and the OctoPrint instances connected to printers in the farm. OctoPrint transmits both print job status and printer state [information over WS](#), which can be connected to through authentication REST endpoints. OctoPrint also serves [REST endpoints](#) for issuing commands related to print jobs, printer settings, and uploading of files for print jobs. The role of the backend system will be to route these commands from the web UI to an appropriate printer for the job.

An important part of the backend system will be to handle user authentication and authorization, to ensure that users are not able to interact or interfere with other users' jobs, either by accident or on purpose. In the first version of this system, user account creation and credentials authentication will be handled by implementations in the system itself (e.g. bcrypt and a local password/user database), but may be moved to external auth systems, e.g. Feide and the OAuth framework in the future. As access to the Feide APIs requires a lengthy application process to Dataporten, this work is regarded as probably being outside the scope of this project.

Assumptions

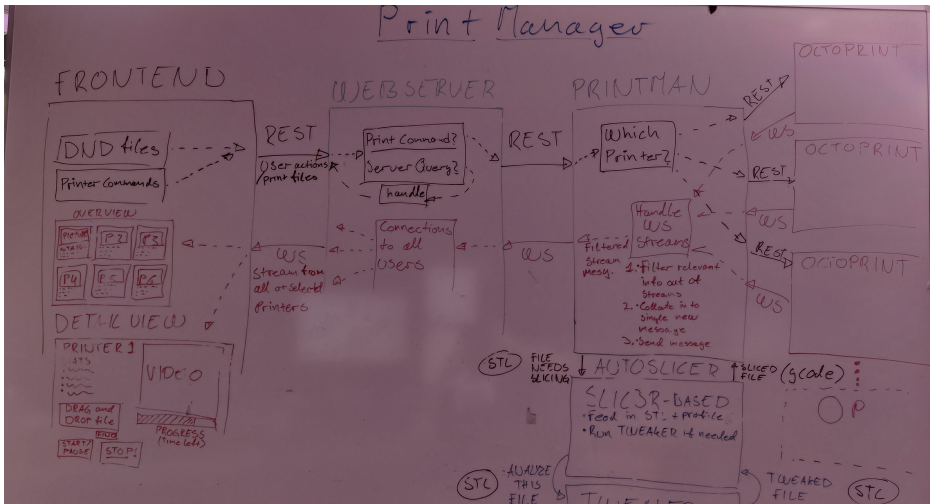
- The system will be expanded upon by work done by future student projects at Manulab, and will require thorough documentation.
- The system implementation is not expected to be interacted with by students other than those hired by Manulab for the express purpose of continuing the development work.
- The system will run as containerized services
- The system is not expected to handle requests/interactions from more than a few dozen unique users on a daily basis
- The system is expected to be able to handle 20-30 printers at minimum (the current number of printers in the farm), and to handle an increasing number as the printer farm at Manulab is expanded.
- The system will interact with two types of printers at the start (Prusa i3 mk3+ and Prusa Mini+), but may be required to handle printers of other types (e.g. resin printers), design (e.g. custom built projects) or manufacturer (e.g. Ultimaker) in the future.

Requirements

#	Title	User Story	Importance	Notes
1	OctoPrint instance WS connection	As a system administrator, I want the system to be able to handshake and connect to an arbitrary number of OctoPrint instances over websocket, so that live data from all the printers in the farm can be made available on the web UI.	Must have	<ul style="list-style-type: none">• DB to store and add known OP instances the system should communicate with• Ability to distinguish, sort and collate several WS streams from multiple OP instances at once

2	OctoPrint instance REST connection	As a system administrator, I want the system to be able to access the REST endpoints of the OctoPrint API for an arbitrary number of OP instances, so that printer commands and print job files can be transferred from users of the web UI to the correct printer.	Must have	<ul style="list-style-type: none"> DB to store and add known OP instances the system should communicate with Able to distinguish instances to ensure commands and jobs are sent to correct printer
3	OctoPrint instance database	As a system administrator, I want a database solution for storing known OctoPrint instances, the hardware running it (IP, MAC, serial number), and the printers they are connected to, to ensure that there is a way to keep track of the printer and computer hardware in the system, and their relations.	Must have	<ul style="list-style-type: none"> Database will be used by system for identifying OctoPrint instances and their related printers and OS provisioning status. Should be expandable or interoperable with future projects relating to data analysis and statistics tracking.
4				

User interaction and design



Questions

Below is a list of questions to be addressed as a result of this requirements document:

Question	Outcome
How complicated is the application process for Feide access?	Very. The IHB faculty will have to handle this process in the future if they find the requirement to be of value.
How well does the existing auto-slicer system work? Is it feasible to expand/improve upon this implementation?	Does not work very well. Will require extensive rewriting to be brought to a stable and production ready state.
Will OctoPrint instances behave differently if other printer models or other printer types are used in future?	Octoprint seems quite universal in it's API implementation, and any implementation that follows its documentation should only require minor adjustments if future printers differ from current ones.

Not Doing

- Feide login. Requires application made by the faculty.
- Auto-slicer. System will only handle pre-sliced .gcode files to start with.

D Decision Reports

Clustering Raspberry Pi hardware for system server

Status	DECIDED
Stakeholders	Simen Nesse Wiik Paul Steffen Kleppe
Outcome	Build a compute cluster, using Raspberry Pi hardware as nodes. Netboot the nodes in the cluster, as well as the Raspberry Pi's used for OctoPrint, to avoid the unreliability and failure rate of microSD cards when used for write-heavy operating system scenarios. Implement a netbooting system administration toolset (scripts, software, OS distributions etc.) that streamline administration and future scaling of the system.
Due date	15 Feb 2022
Owner	Olav Valle

Background

The 3D printer management system being developed will require both a webserver for the frontend, as well as a backend system to orchestrate communications between printers and users.

Close physical proximity between the backend's server hardware and the printers is a requirement, for reasons of hardware constraints (wiring and construction of the printer farm and its mobile printer racks), as well as from a maintenance viewpoint (administrators of the system are also repair technicians for the printer hardware).

While a simple desktop computer would suffice for the low traffic and processing load the system is expected to sustain, a single machine also represents a single point of failure for the system.

An approach that is architecturally similar to the infrastructure in many modern data centers can be achieved by clustering several Raspberry Pi computers. This allows for both high availability of services running on the hardware, as well as providing redundancy and robustness to the server itself (since a cluster can continue to function in a reduced state if hardware is damaged). This eliminates, or at least mitigates, the risks presented by a single point of failure. This cluster architecture also allows for scaling the compute power of the server to meet future requirements (by adding more Raspberry Pi's to the cluster).

Another deciding factor is the requirement that the system being developed will form the foundation for future projects at Manulab Ålesund. As this work will inevitably involve research, exploration and testing of prototype machinery and software, having full control of the system hardware is essential for the rapid prototyping approach that students and researchers at Manulab employ. Relying on servers and systems that are administrated by cloud service providers or the IT department of NTNU would present unwanted hurdles in the progress of these future projects.

The developers of the Printer Management system have also expressed an interest in the learning possibilities presented by designing and implemented a such a computing cluster, as this subject is not something that has been extensively covered in the syllabus of their study programs.

Action items



Containerized dev environment

Status	DECIDED
Stakeholders	Simen Nesse Wiik
Outcome	To containerise the Turborepo project in a Docker-Compose, so that current and future developers can all be sure that they work in a reproducible development environment.
Due date	07 Feb 2022
Owner	Olav Valle

Background

The mantra "but it works on my machine" is well known by all software developers working in collaborative projects. Using a container like Docker to encapsulate and normalize the development environment in a reproducible way can mitigate many of these issues, and in essence ensures that every developer uses the "same" machine for their work.

Action items



Why Kafka and MQTT as local protocol?

Status	DECIDED
Stakeholders	Simen Nesse Wiik Paul Steffen Kleppe
Outcome	Use MQTT and Kafka + Connect. MQTT is the industry standard for IoT device communication protocols. Apache Kafka is widely used in Industry 4.0 companies for data streaming and analytics in distributed hardware systems, and provides easy options for creating scaleable infrastructure. Both Kafka and MQTT integrate well with NestJS, and in combination these make for a robust solution for our backend architecture.
Due date	13 Feb 2022
Owner	Olav Valle

Background

The Printer Manager server will manage status traffic and issue files/job commands to and from a large number of printers. OctoPrint serves this data as a WS stream to its own frontend, but this solution lacks granularity and will require processing of the message streams to reduce unnecessary traffic loads. The MQTT protocol is widely used for two-way broadcast/subscriber type communications in large hardware systems, like networks of sensors and smart devices. Nest.js supports the MQTT protocol through Kafka, a framework for handling and managing MQTT networks.

Action items



Remove Kafka and MQTT requirement

Status	DECIDED
Stakeholders	Simen Nesse Wiik Paul Steffen Kleppe Olav Valle
Outcome	All work on implementing Kafka and the MQTT protocol has been removed as a requirement. The development of this feature will be handed off to the data analysis project team at the end of May 2022.
Due date	02 May 2022
Owner	Olav Valle

Background

The use of Kafka as a data streaming, analytics and storage platform was planned. The reason behind this was to allow for collection and analysis of data (printer status/progress and statistics) from the 3D printer farm, to serve as a foundation for future projects at Manulab.

Kafka was recommended by the developers, after researching various options, because it is widely used in the industry, as is the MQTT IoT communications protocol. It was decided by stakeholders at the time to add this as a requirement to the project.

However, a project that will continue development of the analytics system has been scheduled to start in May/June of 2022. The team of this future project has requested that the decision to implement Kafka and MQTT be reviewed (to better understand the reasoning and consequences of this decision), or for the decision to ultimately be left at their discretion.

Action items



Why Nest.js for backend architecture?

Status	DECIDED
Stakeholders	Simen Nesse Wiik
Outcome	Use NestJS for backend application. NestJS allows us to leverage our knowledge of OOP application design as well as providing a robust framework for scaleable MVC and microservice architectures.
Due date	07 Feb 2022
Owner	Olav Valle

Background

The backend will have a fairly complicated architecture, with scalability and modularity as part of its main goals. Good architecture is critical to achieve this, and there exist certain frameworks that specialize in simplifying the process of architecting software projects as microservices and using the MVC model.

Action items



Why Octoprint for printer connection?

Status	DECIDED
Stakeholders	Simen Nesse Wiik
Outcome	Use OctoPI as the OS for printer connected Raspberry Pis. This allows us to leverage OctoPrint's robust REST API and popular plugin ecosystem.
Due date	13 Feb 2022
Owner	Olav Valle

Background

What alternatives are there to Octoprint for handling printer stats and commands?

Action items



Using a monorepo (Turborepo) or a multirepo project structure.

Status	DECIDED
Stakeholders	Simen Nesse Wiik
Outcome	Use a monorepo through Turborepo. As every part of the project is written in Node/TS, the monorepo structure provided by Turborepo offers many features and functions that will help the project.
Due date	07 Feb 2022
Owner	Olav Valle

Background

The structure and method for organising the project code repositories can have a large impact on the workflow of the project. Monorepos are used by many of the largest software companies in the world, but have traditionally required custom software and complicated routines to manage larger projects. Turborepo is a tool that aims to simplify the setup and management of Node/TS based projects, along with providing improvements for the dev, build and deployment process in the form of smarter and more efficient use of system resources.

Action items

-

Web UI Framework

Status	DECIDED
Stakeholders	Simen Nesse Wiik
Outcome	The team has decided to use the React framework and the Material UI component library. Furthermore, Next.js will be used instead of "vanilla" react, as it provides a framework for structuring web applications and interaction with APIs.
Due date	07 Feb 2022
Owner	Olav Valle

Background

The JS ui framework used for the website frontend plays a massive part in the application tech stack of the project. It is therefore important to select a framework that is suitable for the project and the developers on the team.

Action items

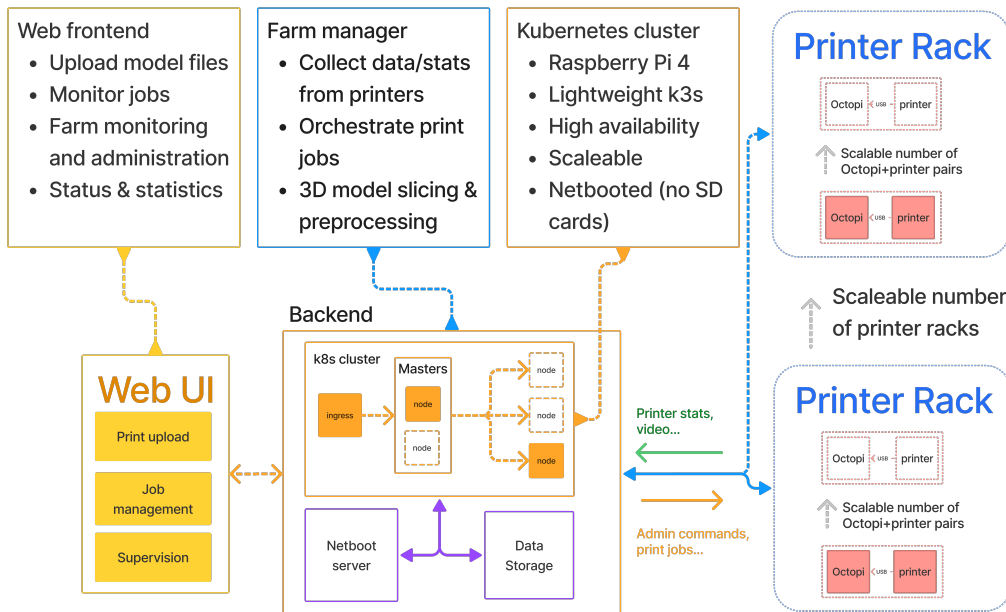


E System documentation

E.1 Introduction

This document documents the system architecture, code and how to perform maintenance tasks.

E.2 Hardware Architecture



[H]

Figure E.1: Hardware system diagram.

A Networking

The network consists of a router, two switches and attached devices.

B Raspberry Pi

The entire PrintMan system runs on Raspberry Pis. The Pis are connected to a PoE+ ethernet switch, supplying the Pis with both power and data transfer in the same cable. This eliminates the need for a wall-plugged power adapter.

When connected to the switch with an ethernet cable, the raspberry pi is PXE booted. PXE booting is a process where the operating system is not loaded from internal storage such as SD cards, but instead fetched from the network.

E.3 Software architecture

A Frontend

The frontend uses the javascript frontend framework Next for serving the website, and the UI framework MUI for styling and layout. It fetches printer data through websockets.

B backend

The backend uses the Nest framework. See <https://nestjs.com/>

E.4 Code repository

The repository for the source code produced for the web application in this project can be found at: <https://github.com/NTNU-manulab/printman>

E.5 Containers

A docker-compose file to build a container for the development environment is found in the code repository.

E.6 Kubernetes

The PrintMan services run inside containers on kubernetes. This section describes how to connect to the cluster

A Using Kubectl to administer the cluster

Kubectl (pronounced *kube control* or *kube cuttle*) lets an administrator perform operations on the cluster such as creating new deployments.

Imperative vs declarative

With kubectl, imperative and declarative commands can be executed. With an imperative command, all parameters are specified

B Adding nodes to the cluster

E.7 Security

No code requiring security, authentication or authorization implementation has been produced by this project.

Public-private key pairs were used for SSH connections to the Raspberry Pi's in the cluster and printer racks.

E.8 Installation

Clone the repository and run using 'turborepo run dev'. No installable binaries are produced by this project.

E.9 Source code documentation

No source code documentation has been produced for the webapp MVP. See G for documents providing walkthroughs and explanations of the work done for kubernetes cluster setup and hardware cluster sysadmin and provisioning work.

F Source Code and Bash Scripts

The code can be found in the Manulab github organization

<https://github.com/NTNU-manulab/printman>

F.1 Kubernetes YAML definitions

A IngressRoute

```
apiVersion: traefik.containo.us/v1alpha1
kind: IngressRoute
metadata:
  name: web
spec:
  routes:
  - kind: Rule
    match: Host(`manulab.net`)
    services:
    - kind: Service
      name: web
      passHostHeader: true
      port: 80
      scheme: http
      serversTransport: transport
```

```
apiVersion: traefik.containo.us/v1alpha1
kind: IngressRoute
metadata:
  name: api
spec:
  routes:
  - kind: Rule
    match: Host(`api.manulab.net`)
    services:
    - kind: Service
```

```
    name: api
    passHostHeader: true
    port: 80
    scheme: http
    serversTransport: transport
```

B Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
  labels:
    app: web
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
        - name: revosw/printman-web
          image: revosw/printman-web:latest
          ports:
            - containerPort: 80
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: api
  labels:
    app: api
spec:
  replicas: 1
  selector:
    matchLabels:
      app: api
  template:
    metadata:
      labels:
```

```
    app: api
spec:
  containers:
  - name: revosw/printman-api
    image: revosw/printman-api:latest
    ports:
    - containerPort: 80
```

C Service

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: web
  name: web
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 3000
  selector:
    app: web
```

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: api
  name: api
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 3001
  selector:
    app: api
```

G Work diaries

Work Log: Creating baseline RPi file systems

Goal

Create a baseline `/boot` and `/rootfs` that the bootserver scripts can modify to create unique fs's for each Pi.

PRINTMAN-75 - RPi provisioning script IN PROGRESS

Resources

[The kernel-command-line](#): Details the arguments used in `cmdline.txt`

[Usermod](#): Built in Linux utility for modifying user accounts: changing usernames, setting passwords, configuring home directories, etc.

[Groupmod](#): Modifies definitions for usergroups.

[Install](#): A fancy and convenient way of creating and copying files and directories, while setting their owners, permission modes etc.

[sed](#): The GNU **s**tream **e**ditor. Used to perform basic transformations on text, either from files or pipes (streams). Basically a find-and-replace function for the command line. Beware: Hic Sunt Regex

[openssl](#): A cryptography toolkit. Used to hash user passwords through `openssl passwd`. We use the SHA512 algorithm, as [recommended by the Raspberry Pi Foundation](#)

Todo

- Figure out what we need from the `firstboot.sh` script that the Raspberry Pi Imager adds when you flash an OS to an SD card.
- Modify `firstboot.sh` to suit our needs
 - Assign RPi serial number as hostname.
 - Assign username based on OS:
 - `busk` for k3s cluster Pi
 - `printer` for Octoprint Pi
 - Generate and hash password for user
 - Same as username?
- Preconfigure all the things that every system will have in common, and remove from script
 - SSH config, bootserver pubkey
 - Timezone, language and localization config
- Modify `cmdline.txt`
 - Remove `resizefs` script. There is no partition to resize.
 - Set `root` argument to `/dev/nfs`
 - Add `nfsroot=[netboot-server-IP]:/path/to/nfsroot/PI-SN#,vers=4.1,proto=tcp`
 - Add `cgroup_enable=cpuset cgroup_memory=1 cgroup_enable=memory` (at least for k3s Pi's)
- Modify `/etc/fstab`
 - Remove device mounts for `/boot` and `/`
 - Add `nfs` mounts for `/boot` with `[netboot-server-IP]:/path/to/tftpboot/PI-SN# /boot nfs defaults,_netdev,vers4.1,proto=tcp 0 0`

What does Firststrun.sh actually do?

The script below is generated by the RPi Imager when the OS is flashed to an SD card.

firstrun.sh

```
#!/bin/bash

set +e

CURRENT_HOSTNAME=`cat /etc/hostname | tr -d " \t\n\r"`
echo bootsrv >/etc/hostname
sed -i "s/127.0.1.1.*$CURRENT_HOSTNAME/127.0.1.1\tbootsrv/g" /etc/hosts
FIRSTUSER=`getent passwd 1000 | cut -d: -f1`
FIRSTUSERHOME=`getent passwd 1000 | cut -d: -f6`
install -o "$FIRSTUSER" -m 700 -d "$FIRSTUSERHOME/.ssh"
install -o "$FIRSTUSER" -m 600 <(printf "YOUR_SSH_KEY") "$FIRSTUSERHOME/.ssh/authorized_keys"
echo 'PasswordAuthentication no' >>/etc/ssh/sshd_config
if [ -f /usr/lib/userconf-pi/userconf ]; then
    /usr/lib/userconf-pi/userconf 'bootsrv' '$5$SAs192S7Ar$3ubTjCtGAz37fdcuzNopNlj3Sw3TFcx3dnK1D.xCjs5'
else
    echo "$FIRSTUSER: '$5$SAs192S7Ar$3ubTjCtGAz37fdcuzNopNlj3Sw3TFcx3dnK1D.xCjs5' | chpasswd -e
    if [ "$FIRSTUSER" != "bootsrv" ]; then
        usermod -l "bootsrv" "$FIRSTUSER"
        usermod -m -d "/home/bootsrv" "bootsrv"
        groupmod -n "bootsrv" "$FIRSTUSER"
        if grep -q "^autologin-user=" /etc/lightdm/lightdm.conf ; then
            sed /etc/lightdm/lightdm.conf -i -e "s/^autologin-user=.*\/autologin-user=bootsrv/"
        fi
        if [ -f /etc/systemd/system/getty@tty1.service.d/autologin.conf ]; then
            sed /etc/systemd/system/getty@tty1.service.d/autologin.conf -i -e "s/$FIRSTUSER/bootsrv/"
        fi
        if [ -f /etc/sudoers.d/010_pi-nopasswd ]; then
            sed -i "s/^$FIRSTUSER /bootsrv /" /etc/sudoers.d/010_pi-nopasswd
        fi
    fi
fi
systemctl enable ssh
rm -f /etc/localtime
echo "Europe/Oslo" >/etc/timezone
dpkg-reconfigure -f noninteractive tzdata
cat >/etc/default/keyboard <<'KBE0F'
XKBMODEL="pc105"
XKBLAYOUT="us"
XKBVARIANT=""
XKBOPTIONS=""

KBE0F
dpkg-reconfigure -f noninteractive keyboard-configuration
rm -f /boot/firstrun.sh
sed -i 's| systemd.run.*||g' /boot/cmdline.txt
exit 0
```

The script is run via `cmdline.txt`, after the root partition has been resized, as part of the first boot setup process of the RPiOS.

cmdline.txt

```
console=serial0,115200 console=tty1 root=PARTUUID=0ee3e8a8-02 rootfstype=ext4 fsck.repair=yes rootwait quiet
init=/usr/lib/raspi-config/init_resize.sh systemd.run=/boot/firstrun.sh systemd.run_success_action=reboot
systemd.unit=kernel-command-line.target
```

Here we can see that the following happens

- The root partition is resized to fill the entire SD card by the `init_resize` script. We don't need this, as there is no SD card and no partition to resize.
- `firstrun.sh` is executed via `systemd.run`
- If the script executed successfully, the system reboots via `systemd.run_success_action`

The script itself performs several actions:

First, it replaces the default hostname with our chosen hostname in `/etc/hostname` and `/etc/hosts`:

set hostname

```
CURRENT_HOSTNAME=`cat /etc/hostname | tr -d " \t\n\r"`
echo bootsrv >/etc/hostname
sed -i "s/127.0.1.1.*$CURRENT_HOSTNAME/127.0.1.1\tbootsrv/g" /etc/hosts
```

It then finds the username and home directory of the "first user", i.e. the one we want to be the default user:

get "first user"

```
FIRSTUSER=`getent passwd 1000 | cut -d: -f1`
FIRSTUSERHOME=`getent passwd 1000 | cut -d: -f6`
```

The script uses these values to add our SSH key to the `authorized_keys` file of this user, and to configure the SSH Daemon of the system:

SSH config

```
install -o "$FIRSTUSER" -m 700 -d "$FIRSTUSERHOME/.ssh"
install -o "$FIRSTUSER" -m 600 <(printf "YOUR_SSH_KEY") "$FIRSTUSERHOME/.ssh/authorized_keys"
echo 'PasswordAuthentication no' >>/etc/ssh/sshd_config
```

After this, the script modifies the "first user" account, setting the password, permissions, sudoers rights, etc, for the user account. It does this either using the `/usr/lib/userconf-pi/userconf` script (if it exists), or via a block of code that performs basically the same tasks as the `userconf` script would:

```
if [ -f /usr/lib/userconf-pi/userconf ]; then
    /usr/lib/userconf-pi/userconf 'bootsrv' '$5$SAs192S7Ar$3ubTjCtGAz37fdcuzNopN1j3Sw3TFcx3dnK1D.xCjs5'
else
    echo "$FIRSTUSER:" '$5$SAs192S7Ar$3ubTjCtGAz37fdcuzNopN1j3Sw3TFcx3dnK1D.xCjs5' | chpasswd -e
    if [ "$FIRSTUSER" != "bootsrv" ]; then
        usermod -l "bootsrv" "$FIRSTUSER"
        usermod -m -d "/home/bootsrv" "bootsrv"
        groupmod -n "bootsrv" "$FIRSTUSER"
        if grep -q "^autologin-user=" /etc/lightdm/lightdm.conf ; then
            sed /etc/lightdm/lightdm.conf -i -e "s/^autologin-user=.*\/autologin-user=bootsrv/"
        fi
        if [ -f /etc/systemd/system/getty@tty1.service.d/autologin.conf ]; then
            sed /etc/systemd/system/getty@tty1.service.d/autologin.conf -i -e "s/$FIRSTUSER/bootsrv/"
        fi
        if [ -f /etc/sudoers.d/010_pi-nopasswd ]; then
            sed -i "s/^\$FIRSTUSER /bootsrv /" /etc/sudoers.d/010_pi-nopasswd
        fi
    fi
fi
```

As the `userconf` script exists in our case, the code in the `else` clause does not come into play for us. The `userconf` script looks like this:

/usr/lib/userconf-pi/userconf

```
#!/bin/sh

rename_user () {
    usermod -l "$NEWNAME" "$FIRSTUSER"
    usermod -m -d "/home/$NEWNAME" "$NEWNAME"
    groupmod -n "$NEWNAME" "$FIRSTGROUP"
    for file in /etc/subuid /etc/subgid; do
        sed -i "s/^$FIRSTUSER:$NEWNAME:/" "$file"
    done
    if [ -f /etc/sudoers.d/010_pi-nopasswd ]; then
        sed -i "s/^$FIRSTUSER /$NEWNAME /" /etc/sudoers.d/010_pi-nopasswd
    fi
}

if [ $# -eq 3 ]; then
    FIRSTUSER="$1"
    FIRSTGROUP="$1"
    shift
else
    FIRSTUSER="$(getent passwd 1000 | cut -d: -f1)"
    FIRSTGROUP="$(getent group 1000 | cut -d: -f1)"
fi

NEWNAME=$1
NEWPASS=$2

if [ "$FIRSTUSER" != "$NEWNAME" ]; then
    rename_user
fi

if [ -n "$NEWPASS" ]; then
    echo "$NEWNAME:$NEWPASS" | chpasswd -e
fi

/usr/bin/cancel-rename "$NEWNAME"
```

As we can see, it takes in a username and password (line 24-25), and modifies the "first user" account with these values via `usermod` and `groupmod`, which are built in Linux commands (lines 5-7). On line 9 it changes the entries in `/etc/subuid` and `/etc/subgid` to match our username. On line 11-13, it changes the `010_pi-nopasswd` file to also reflect our chosen username.

- Note: the entry in `010_pi-nopasswd` makes it so that our user account doesn't have to provide the sudo password when executing commands via `sudo`. **This may be a bad thing...**

After all the user account configuration stuff is done, the script activates SSH on the system:

enable ssh

```
systemctl enable ssh
```

This, in combination with adding our SSH key and configuring the SSH Daemon earlier, is required to allow us to connect via SSH.

The script then sets localization, language, keyboard and time zone options:

```

rm -f /etc/localtime
echo "Europe/Oslo" >/etc/timezone
dpkg-reconfigure -f noninteractive tzdata
cat >/etc/default/keyboard <<'KBEOF'
XKBMODEL="pc105"
XKBLAYOUT="us"
XKBVARIANT=""
XKBOPTIONS=""

KBEOF
dpkg-reconfigure -f noninteractive keyboard-configuration

```

The localization and language configuration is not vitally important to us. Setting the time zone is important, however, to ensure that timestamps generated by the system reflect our local time (and that all the Pi's in the system create time stamps that match).

Finally, the script performs some clean up. It deletes the script itself (since it's not needed again), and removes the kernel command line arguments that caused the script to execute in the first place:

clean up

```

rm -f /boot/firstrun.sh
sed -i 's| systemd.run.*||g' /boot/cmdline.txt
exit 0

```

The script exits with code 0, which lets `systemd` know that it's appropriate to perform the reboot command as specified by `systemd.run_success_action=reboot` in `cmdline.txt`.

- Note: If we want to add a safety check, and perform some specific action if the `firstrun.sh` script fails, we can add a `systemd.run_failure_action=` argument. This will run if the `firstrun.sh` script returns anything else than exit code 0, and could be used e.g. for logging or clean up/recovery.

When the system reboots, `cmdline.txt` no longer contains the `systemd.run=firstrun.sh` argument, and boots into the OS normally.

What should *our* `firstrun.sh` do?

There are several things that *all* systems running the same OS will have in common, and a lot of it can be set in the config files of the baseline image. The things we need our script to do are:

- The username (and password...?)
 - This eliminates the need for everything that has to do with setting user rights etc., as this can be configured in the baseline image.
 - To do this, we have to find the files that `usermod` and `groupmod` make changes to.
 - Or, we can add a file named `userconf.txt` to the `/boot` directory. The contents of the file should be a username and hashed pw in the format `username:password-hash`. This will trigger the RPi OS to create this user with the default settings (autologin, sudoers NOPASSWD, etc..)
- Hostname (Pi's serial number)
 - We can preconfigure these values into `/etc/hosts` and `/etc/hostname` as part of the script that creates the OS file systems.
- SSH stuff
 - The pub key of the bootserver can be preconfigured in `/home/$FIRSTUSER/.ssh/authorized_keys`, as can the settings in `sshd.config`.
 - **SSH still has to be activated through `systemctl enable ssh`, or:**
 - SSH can also be activated by placing a file named `ssh` in `/boot`. This triggers the OS to automatically activate SSH on the next system boot, after which the file is deleted.
 - All localisation settings.
 - Parameters in `/etc/timezone` and `/etc/default/keyboard` can be preconfigured.
 - **We must still run `dpkg-reconfigure -f noninteractive for tzdata and keyboard-configuration!`**

This leaves us with a pared down version of `firstrun.sh` that looks a little like this:


```
#!/bin/bash

set +e

# Timezone
dpkg-reconfigure -f noninteractive tzdata

# Keyboard layout
dpkg-reconfigure -f noninteractive keyboard-configuration

# Cleanup
rm -f /boot/firststrun.sh
sed -i 's| systemd.run.*||g' /boot/cmdline.txt
exit 0
```

Yeah. Turns out there's now a whole lot we can't just preconfigure through files. God I love linux.

So, here's what we have to do:

Create a user

This can be done without a script. We simply place a file named `userconf.txt` in the `/boot` directory of the baseline image. The file should contain a single line consisting of the username and the hashed and salted password of the user, in the format `USERNAME:SALTED-HASHED-PW`. If this file is present at first boot, the Pi will automatically set this as the default user of the system.

The hash *can* be generated as part of the provisioning script, or we can just use the same password for all the systems. This is sort of bad practice, but the Pi's won't be accessible from outside the network, and it's not the default password in any case. Using the same password only means that anyone who manages to infiltrate the network and crack the password will have access to *all* the Pi's, and not just the one they crack the password for. This is unlikely to happen (SHA512 is computationally infeasible to brute force), and using the same pw for all systems saves us from having to store 40-50 different passwords in a secure way.

The thing we *do* have to change from the default user configuration that RPi OS creates, is to disable auto login and passwordless sudo.

Disable Auto Login

(Not required. See conclusion at end of section.)

Auto login might not actually be enabled. If it is, we should find the file `/etc/systemd/system/getty@tty1.service.d/autologin.conf`, which probably contains something like this:

```
[Service]
ExecStart=
ExecStart=-/sbin/agetty --autologin $USER --noclear %I \${TERM}
```

Source: [StackExchange](#)

This is the config file of a service that starts a `tty` with the default user automatically logged in. We don't want this.

To disable this service, either remove the file, or rename it to something like `autologin.conf.bak` if you want to be able to easily re-enable auto login later.

```
# The mv command moves a file from one location to another.
# In this case, the {} at the end will be expanded to provide both the source (autologin.conf) and destination
# (autologin.conf.bak) parameters of the command.

sudo mv /etc/systemd/system/getty@tty1.service.d/autologin.conf{,.bak}
```

On the next system boot, you should be prompted for login credentials.

Doing this remove/rename in the `firststrun.sh` script doesn't seem to work, though. First guess is that the `autologin.conf` service is created *after* `firststrun.sh` is executed. A likely candidate is `raspi-config` (found in `/usr/bin/raspi-config`). This utility is used to configure several different parts of the RPi's OS, including the "boot behaviour". From `raspi-config`:

```

1377 ? do_boot_behaviour() {
1378 ?   if [ "$INTERACTIVE" = True ]; then
1379 ?     BOOTOPT=$(whiptail --title "Raspberry Pi Software Configuration Tool (raspi-config)" --menu "Boot
Options" $WT_HEIGHT $WT_WIDTH $WT_MENU_HEIGHT
? \
1380 ?       "B1 Console" "Text console, requiring user to login" \
1381 ?       "B2 Console Autologin" "Text console, automatically logged in as '$USER' user" \
1382 ?       "B3 Desktop" "Desktop GUI, requiring user to login" \
1383 ?       "B4 Desktop Autologin" "Desktop GUI, automatically logged in as '$USER' user" \
1384 ?       3>&1 1>&2 2>&3)
1385 ?   else
1386 ?     BOOTOPT=$1
1387 ?     true
1388 ?   fi
1389 ?   if [ $? -eq 0 ]; then
1390 ?     case "$BOOTOPT" in
1391 ?       B1*)
1392 ?         systemctl --quiet set-default multi-user.target
1393 ?         rm -f /etc/systemd/system/getty@tty1.service.d/autologin.conf
1394 ?         ;;
1395 ?       B2*)
1396 ?         systemctl --quiet set-default multi-user.target
1397 ?         cat > /etc/systemd/system/getty@tty1.service.d/autologin.conf << EOF
1398 ? [Service]
1399 ? ExecStart=
1400 ? ExecStart=-/sbin/agetty --autologin $USER --noclear %I \${TERM}
1401 ? EOF
1402 ?         ;;
1403 ?       B3*)
1404 ?         if [ -e /etc/init.d/lightdm ]; then
1405 ?           systemctl --quiet set-default graphical.target
1406 ?           rm -f /etc/systemd/system/getty@tty1.service.d/autologin.conf
1407 ?           sed /etc/lightdm/lightdm.conf -i -e "s/^autologin-user=.*/#autologin-user=/"
1408 ?           disable_raspi_config_at_boot
1409 ?         else
1410 ?           whiptail --msgbox "Do 'sudo apt-get install lightdm' to allow configuration of boot to
desktop" 20 60 2
1411 ?           return 1
1412 ?         fi
1413 ?         ;;
1414 ?       B4*)
1415 ?         if [ -e /etc/init.d/lightdm ]; then
1416 ?           systemctl --quiet set-default graphical.target
1417 ?           cat > /etc/systemd/system/getty@tty1.service.d/autologin.conf << EOF
1418 ? [Service]
1419 ? ExecStart=
1420 ? ExecStart=-/sbin/agetty --autologin $USER --noclear %I \${TERM}
1421 ? EOF
1422 ?           sed /etc/lightdm/lightdm.conf -i -e "s/^(#\|\\)autologin-user=.*#autologin-user=$USER/"
1423 ?           disable_raspi_config_at_boot
1424 ?         else
1425 ?           whiptail --msgbox "Do 'sudo apt-get install lightdm' to allow configuration of boot to
desktop" 20 60 2
1426 ?           return 1
1427 ?         fi
1428 ?         ;;
1429 ?     *)
1430 ?       whiptail --msgbox "Programmer error, unrecognised boot option" 20 60 2
1431 ?       return 1
1432 ?       ;;
1433 ?     esac
1434 ?     systemctl daemon-reload
1435 ?     ASK_TO_REBOOT=1
1436 ?   fi
1437 ? }

```

Here we can see that option B2 (lines 1396 - 1401) creates the service config for autologin.conf.

The script also contains references to functions that can disable "raspi-config at boot".

```
1073 ? disable_raspi_config_at_boot() {
1074 ?   if [ -e /etc/profile.d/raspi-config.sh ]; then
1075 ?     rm -f /etc/profile.d/raspi-config.sh
1076 ?     if [ -e /etc/systemd/system/getty@tty1.service.d/raspi-config-override.conf ]; then
1077 ?       rm /etc/systemd/system/getty@tty1.service.d/raspi-config-override.conf
1078 ?     fi
1079 ?     telinit q
1080 ?   fi
1081 ? }
```

Neither of the files concerned are present in the pure Raspberry Pi OS image, nor on one that has been flashed to SD and booted (where the user was created, with autologin, using `/boot/userconf.txt`).

It's possible (likely?) that the `raspi-config` script is run when the OS itself is booted for the first time (which would only happen *after* `firstrun.sh` has been executed and the system reboots, as per `cmdline.txt`). If we can figure out how and when `raspi-config` is run this way, we can hopefully figure out a way to prevent autologin from being activated. As of yet, I have not found out how this works. Web searches for "disable raspberry pi autologin" have so far only turned up how to disable it for *existing* users (by renaming or removing the getty service as described above), with nothing about how to prevent it from being activated in the first place. Hopefully the official documentation can shed some light on this, if we dig deeper.

In the end, having autologin enabled is not a *massive* issue. All it means is that the default user will be automatically logged in to a local terminal on boot. Without physical access (i.e. with a keyboard and a monitor plugged in), there's not much this can actually be used for. Securing the system through requiring sudo password and only allowing SSH auth with pubkey should lock down most, if not all, of the probable attack vectors. Disabling autologin is therefore not considered a priority at this time. If a straightforward solution is discovered, we'll cross that bridge when we get to it.

Disable sudoers nopasswd

To require a password for commands run with `sudo`, simply remove the `/etc/sudoers.d/010_pi-nopasswd` file.

If you want to keep it around, to make it easier to re-enable `nopasswd` if you ever need to, you can instead rename the file to `010_pi-nopasswd.bak`. Use the `mv` command, just like above:

```
sudo mv /etc/sudoers.d/010_pi-nopasswd{,.bak}
```

Either way you do it, the change will become active instantly. No need to reboot or restart any services or processes.

Test that the password is now required by running any command with `sudo` (e.g. `sudo cat /etc/sudoers`). You should be given a warning about the dangers of misusing root privileges, and asked for the login password of your user. Hopefully you remember what it is.

We should probably do this as part of the `firstrun.sh` script:

firstrun.sh disable nopasswd

```
if [ -f /etc/sudoers.d/010_pi-nopasswd ]; then
  mv /etc/sudoers.d/010_pi-nopasswd{,.bak}
  touch nopasswd-success # leaves a file in / to indicate that operation was successful.
fi
```

Set hostname

The hostname of the system has to be set in `/etc/hosts` and `/etc/hostname`. The hostname should be the serial number of the Pi in question. This will have to be done as part of the provisioning script, and we can basically reuse the code from the original `firstrun.sh`.

Our provisioning script therefore needs to contain something along these lines:

```
PISERIAL=01234567 # should be passed as parameter to the provisioning script
CURRENT_HOSTNAME=`cat /icebox/nfsroot/$PISERIAL/etc/hostname | tr -d " \t\n\r"`
echo $PISERIAL >/icebox/nfsroot/$PISERIAL/etc/hostname
sed -i "s/127.0.1.1.*$CURRENT_HOSTNAME/127.0.1.1\$PISERIAL/g" /etc/hosts
```

This will set configure the hostname of the system to the Pi's serial number.

The path to the `hostname` and `hosts` files (set as `/icebox/nfsroot/$PISERIAL` above) will need to be set to the full, absolute path of the NFS share where the root fs of the system should be stored in a directory named after the serial number of the Pi (we will create this directory, and copy the baseline OS files, at the start of the provisioning script. We'll get to that in a bit.)

Installation of k3s server and agent nodes

This document describes the steps to set up 3 control-plane nodes and 5 data-plane nodes in a k3s HA environment

Control-plane: the set of nodes (physical machines, raspberry pis) that manage the worker nodes by keeping track of the lifecycle of worker nodes and correcting the course in the event of failure

Data-plane: the set of nodes that run containers, scheduled by for example running `kubectl create deployment <name> --image=manulab/mqtt`

How to understand the formatting of this document

Blue text denote file names and paths

Green text denote text inside a file

Purple text denote shell commands

Prerequisites

Before installing k3s, you need to edit

`/boot/cmdline.txt`

add this at the end of the line

```
cgroup_enable=memory cgroup_memory=1
```

Server nodes

Tip: The k3s installation bundles `k3s-uninstall.sh` and `k3s-agent-uninstall.sh`. If something goes wrong, just uninstall and try again.

On the very first server, run this command. This initializes the node to become a cluster that other master nodes can connect to.

```
curl -sL https://get.k3s.io | INSTALL_K3S_EXEC="--cluster-init" K3S_TOKEN=test sh -
```

Then on another server node, run this command to connect to the cluster.

```
curl -sL https://get.k3s.io | INSTALL_K3S_EXEC="--server https://192.168.1.150:6443" K3S_TOKEN=test sh -
```

All commands hereafter apply to both first and second (and more) control-plane nodes.

After installation, copy the file

`/etc/rancher/k3s/k3s.yaml`

to the home folder at location

`~/.kube/config`

by running the command

```
sudo cp /etc/rancher/k3s/k3s.yaml ~/.kube/config
```

Then change the permissions of the config file to be readable and writable by the user

```
sudo chown busk:busk .kube/config
```

Then set the KUBECONFIG variable in `.bashrc` to point to the config file

```
export KUBECONFIG=~/.kube/config
```

Now you can try to list all nodes in the cluster

```
kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
buskpi00	Ready	control-plane,etcd,master	23m	v1.22.7+k3s1
buskpi01	Ready	control-plane,etcd,master	21m	v1.22.7+k3s1
buskpi02	Ready	control-plane,etcd,master	2m22s	v1.22.7+k3s1

Agent nodes

On all agent nodes, run this command

```
curl -sfL https://get.k3s.io | K3S_URL=https://192.168.1.150:6443 K3S_TOKEN=test sh -
```

Where `https://192.168.1.150:6443` should be one of your master's ip address

WARN[0000] Cluster CA certificate is not trusted by the host CA bundle, but the token does not include a CA hash.
Use the full token from the server's node-token file to enable Cluster CA validation.

Raspberry Pi Netboot

All the Raspberry Pi's being used, both in the kubernetes cluster and the ones running OctoPrint, will be netbooted. Netbooting is a way of provisioning a device with an OS that is being hosted on a file server. During development, the file server is itself an RPi. This RPi boots from a USB disk, and uses DNSMASQ to serve files to other RPi's on the network.

The OS files are stored on an Icybox HDD bay, containing 5 Seagate drives set up in a ZFS RAIDz2 pool. Using ZFS RAIDz2 ensures both hardware and software level redundancy for the data. The OS files are served to the RPi's over TFTP (Trivial File Transfer Protocol, via a DHCP server by DNSMASQ) for the boot file system, and over NFS for the root file system.

The server

dnsmasq

Instead of adding on to the default `dnsmasq.conf` file, we can instead create our own `.conf` files in `/etc/dnsmasq.d/`, which are automatically appended to the default `.conf`. [Separation of concerns](#) is a good thing.

`/etc/dnsmasq.d/netboot-dhcp.conf`

```
interface=eth0 # Network
interface to operate DHCP/DNS service on. eth0 is the primary Ethernet interface.
dhcp-option=3,192.168.1.1 # Option 3 points DHCP clients
to the IP of the network router.
server=8.8.8.8 # External DNS
server that we want DHCP clients to use. 8.8.8.8 is Google Public DNS
no-resolv # Ignore /etc
/resolv.conf, and use only the above DNS IP
dhcp-range=192.168.1.200,192.168.1.250,5m # IP's to assign dynamic IP leases from
through DHCP
log-dhcp # Extra logging
for DHCP: log all the options sent to DHCP clients and the tags used to determine them.
log-queries # Log the
results of DNS queries handled by dnsmasq.
```

`/etc/dnsmasq.d/netboot-tftp.conf`

```
# TFTP server options
enable-tftp # Enable
dnsmasq's built in TFTP server
tftp-root=/icebox/tftpboot # Absolute path of directory
for boot file systems for clients

#
# CHECK IF THESE ARE ACTUALLY NECESSARY. RPi netboot is not strictly speaking PXE...?
#
pxe-prompt="Boot Raspberry Pi",1 # Setting this provides a prompt to be
displayed after PXE boot. If the timeout is given then after the timeout has elapsed with no keyboard input,
the first available menu option will be automatically executed.
pxe-service=0,"Raspberry Pi Boot " # This specifies a boot option which may
appear in a PXE boot menu. If an integer boot service type (the 0 at the start) then the PXE client will search
for a suitable boot service for that type on the network.
```

/etc/dnsmasq.d/netboot-dhcp-static.conf

```
# Static IP assignments. Should not be from the range defined for dynamic lease above.
dhcp-host=dc:a6:32:46:48:01,set:boot,192.168.1.102      # Assigns static IP to MAC addr, and sets boot flag
to indicate client should be served a bootfs over TFTP
```

Add a dhcp-host line for each Pi being netbooted:

add-dhcp-host.sh

```
#!/usr/bin/env bash

# todo: take input from file with SN#'s or as parameter
export sn=(dc:a6:32:46:48:bb dc:a6:32:4c:cf:da dc:a6:32:46:48:01 dc:a6:32:46:48:e5 dc:a6:32:46:42:2b dc:a6:32:
46:47:65)

for i in "${!sn[@]}"
do
    # todo: find starting IP to assign from highest IP in netboot-dhcp-static.conf.
    echo "dhcp-host=${sn[i]},set:boot,192.168.1.10${i}" | sudo tee -a dnsmasq.d/netboot-dhcp-static.conf
done
```

TFTP

Create a directory at the path specified under `tftp-root` above. In our case, the ZFS pool is mounted as `/icebox`, and holds all our bootfs and rootfs files for the netboot clients:

```
sudo mkdir /icebox/tftpboot
```

Then, for each RPi that will be netbooted, create a directory with the Pi's serial number as name:

```
# Declare a bash array with serial numbers
export ser=(1753bfde bffa8dc5 a677c184 6c738d60 95b9a7be 951b67d8)

# Create directories for the Pi's bootfs in a for loop
for S in "${ser[@]}"
do
    # The -p flag creates parent directories if they do not exist. This avoids messages like "mkdir: cannot
create directory '/icebox/nfsroot/SN': No such file or directory."
    sudo mkdir -p /icebox/tftpboot/$S
done
```

Tip: If you have a lot of Pi's to set up, put all the serial numbers in a file separated by newline, then use that file instead of the array for the loop.

NFS

We need to serve the root FS over NFS. The directory structure is similar to that of the TFTP server, `/icebox/nfsroot/PI-SN`. We might as well perform both actions at once:

```
# Create directories for the Pi's rootfs in a for loop
for S in "${ser[@]}"
do
    sudo mkdir -p /icebox/{nfsroot,tftpboot}/$S
done
```


OS files

Todo:

- Create baseline bootfs:
 - Modify `cmdline.txt`
 - change `root=` to `nfsroot=[netboot-server-IP]:/icebox/nfsroot/PI-SN#`
 - Remove `firstboot` script invocation, or modify it to suit our purpose.
 - Figure out what, if anything, should be modified in `firstboot` script
 - Disable `resizefs` script (there is no partition to resize)

The clients

Work log: Troubleshooting ZFS pool

Attempt to fix issues with ZFS pool.

Apparent cause is a faulty disk (probably SN:35YY, but possibly others).

Issue manifests by all disks (or possibly just 4 of the 5, which may indicate more than one disk at fault) in the icybox chassis disappearing from the device list a few seconds into a large rsync write to the pool. ZFS suspends the pool, citing "IO failures" as the reason for the suspend.

The chicken-and-egg question is whether the disk are disconnecting due to hardware/IO issues, or if ZFS is reporting IO issues because the disks are being disconnected.

The output of `dmesg` may also contain interesting information (both concerning USB devices disconnecting, and call traces from ZFS commands. See: `dmesg-zfs-out.txt`), but it's mostly greek to me at the moment.

The pool (or VDEV, rather) never failed outright. Checksums of files present on the FS passed (tested on a raspberry pi OS image), and ZFS reported "no data errors" at the time. However, data errors are now being reported after the rsync write fails.

After a suspend/disconnect/IO error, any ZFS or ZPOOL commands will hang forever, the process sitting in a 'disk-sleep' state. As this state is uninterruptible, it's impossible to kill the process, and only a hard reboot fixes this. Pool can be destroyed after first clearing the ZFS 'SUSPEND' state.

Steps taken thus far include:

- Removing disk 35YY, formatting it, and using it as a replacement for itself in the VDEV. This seemed to work repair the pool/vdev (DEGRADED state was corrected), until the rsync write fail was discovered.
- Testing each disk for SMART errors and benchmarking its performance (no issues found).
- Destroying the pool, formatting each disk separately, and then rebuilding pool, using all 5 disks.

However, all the issues detailed above still remain, in the exact same manifestation. ZFS suspends the pool a few seconds into a large write operation, and the USB devices disconnect and reconnect.

Possible further steps:

- Format and partition disks (e.g. as ext4), and test for IO errors using rsync. This eliminates ZFS issues, and tests icybox hardware for possible IO /bandwidth or power issues using only one drive at a time.
- Test all drives as an MD based raid. This tests the icybox hardware, in case the issue is caused by faults here (short circuits, IO board issues, powersupply).
- Test drives as separate single device VDEVs. Again tests single drive IO, but with ZFS in the picture. Start with 35YY?
- Test by adding single device VDEVs to pool, one by one, testing for IO issues after each one.
- Test with icybox plugged in to other PC, to eliminate issues with USB ports, drivers, or other system specific issues.

[PRINTMAN-74](#) - Recreate netboot server POC. DONE

[PRINTMAN-85](#) - Debug and fix ZFS issues DONE



`dmesg-zfs-out.txt`

