

UNIVERSITI TEKNOLOGI MALAYSIA

Internship Presentation
13/8/2024 - 20/1/2025

Company SV: Mr Su Yong Yao

Internship SV: Dr. Zuriahati Binti Mohd Yunos

CHIH ZHEN EN

A21EC0167

innovative • entrepreneurial • global

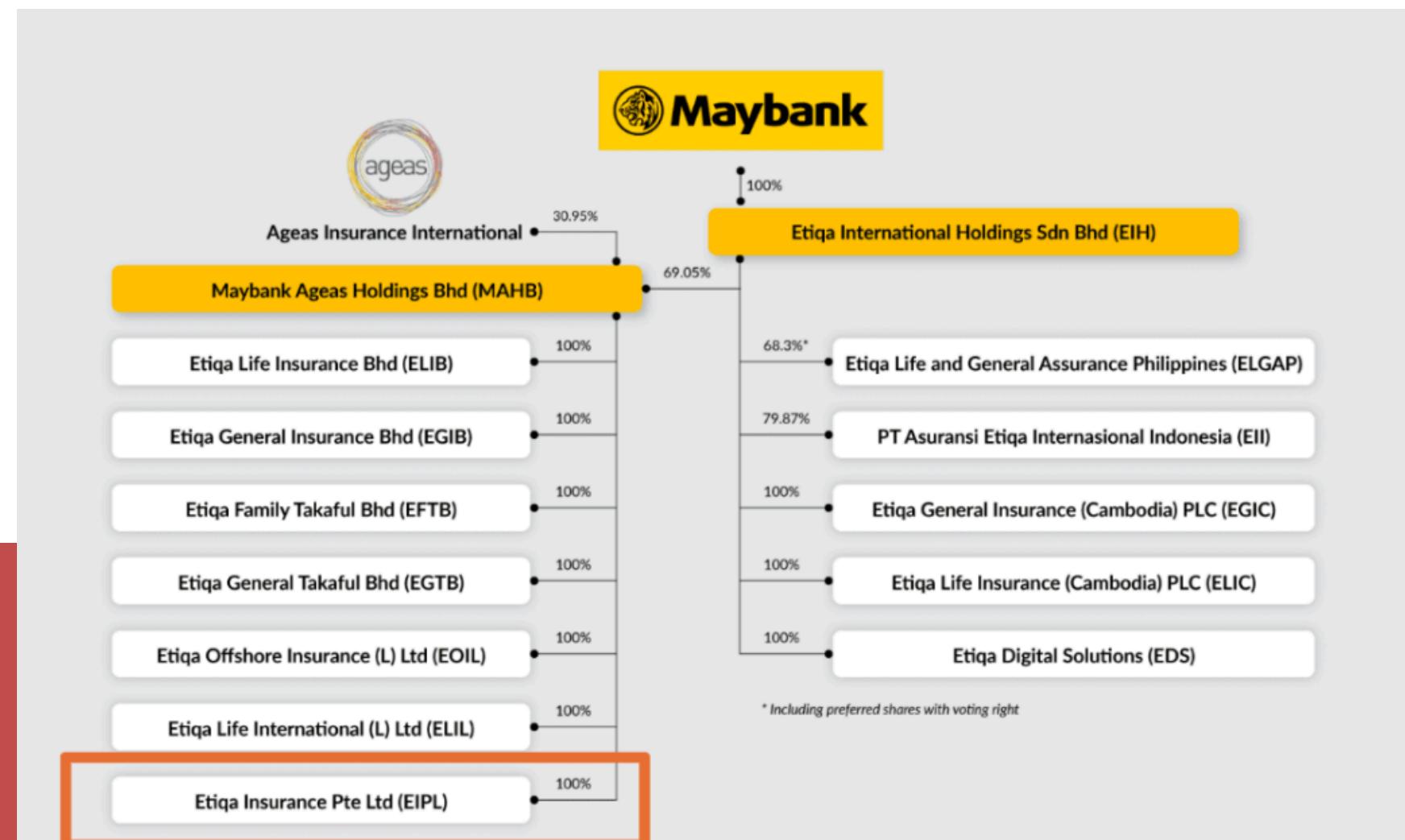


UTM JOHOR BAHRU

Company Information

- Name: Etiqa Insurance
- Industry: Insurance and Financial Services
- Parent Company: Part of Maybank Group
- Products and Services
 - Life insurance
 - General insurance
 - Takaful products (Shariah-compliant offerings)
 - Corporate and personal solutions

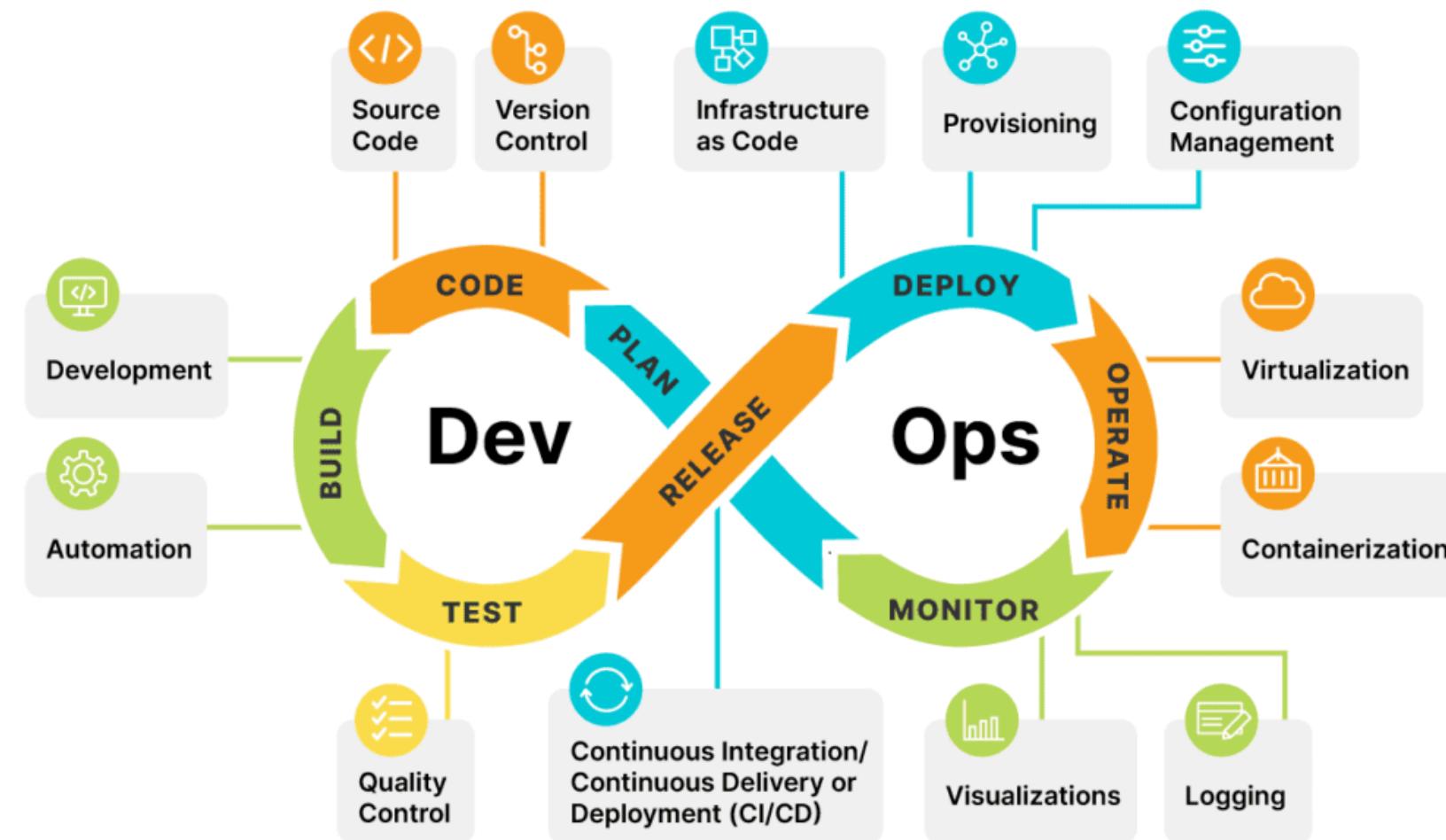
Etiqa is a leading insurance and takaful provider in ASEAN and a part of the Maybank Group, one of the largest banking groups in the region. Etiqa stands out as a forward-thinking company that actively invests in digital transformation to improve customer access, making it a trusted insurance provider in ASEAN. Etiqa is actively expanding its business into other countries, including the Philippines, Indonesia, and Cambodia



Internship Task

What is DevOps

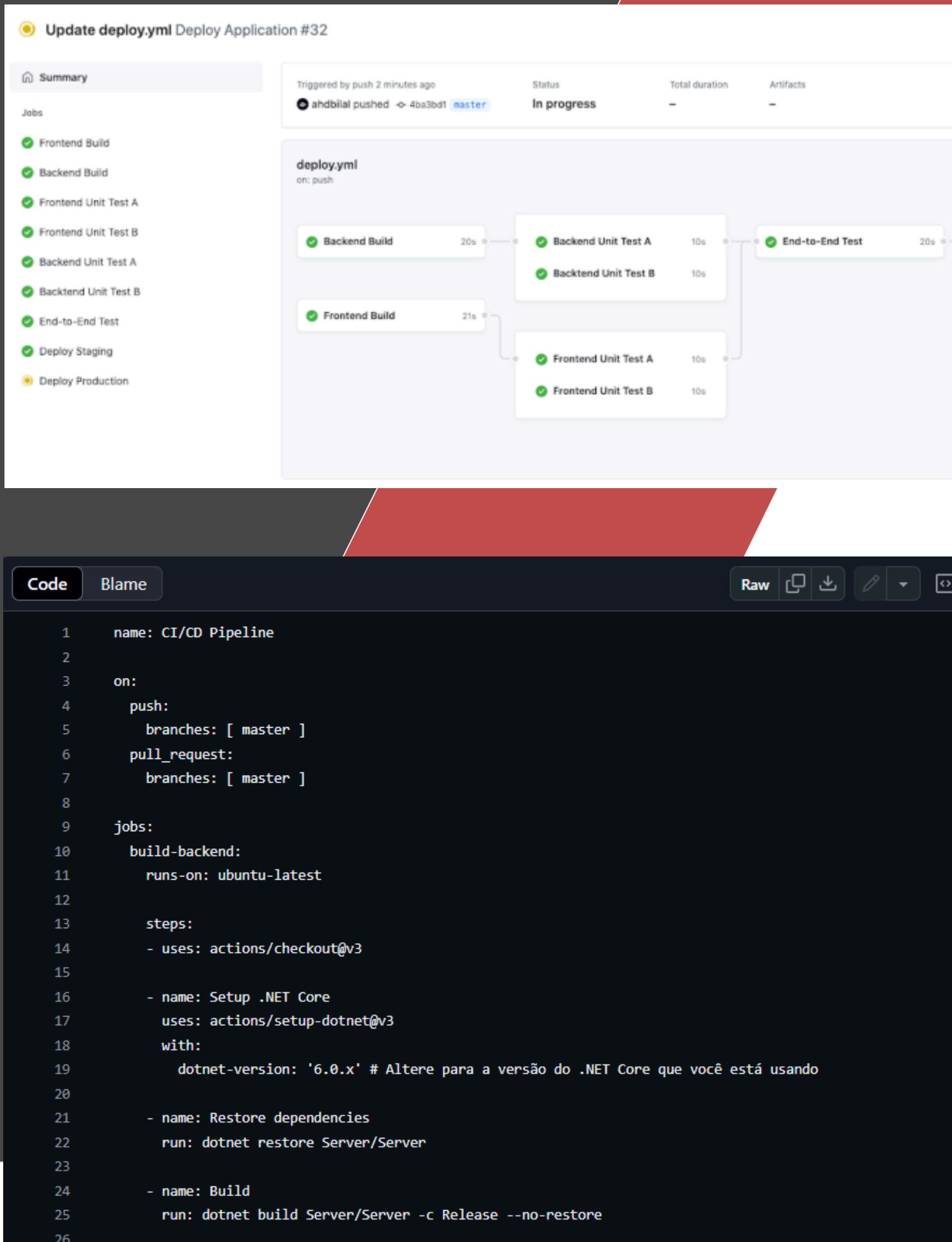
- DevOps is a culture and set of practices that combines software development (Dev) and IT operations (Ops).
- The goal is to shorten the development lifecycle and provide continuous delivery with high software quality.



Software used

- MobaXterm
- MySQL
- Visual Studio
- Postman
- Github Actions
- WinSCP
- Github
- Nexus Registry
- Nginx

Deployment automation



The image shows a screenshot of a GitHub Actions pipeline. At the top, a summary bar indicates a recent push triggered 2 minutes ago, with the status 'In progress' and a total duration of 0s. Below this, a list of jobs is shown: Frontend Build, Backend Build, Frontend Unit Test A, Frontend Unit Test B, Backend Unit Test A, Backend Unit Test B, End-to-End Test, Deploy Staging, and Deploy Production. The 'deploy.yml' file is then displayed, showing a dependency graph where the Backend Build and Frontend Build jobs feed into Backend Unit Test A, Backend Unit Test B, and Frontend Unit Test A, which in turn feed into the End-to-End Test job. The Deploy Staging and Deploy Production jobs are shown as separate entities at the bottom. The bottom half of the image shows the raw YAML code for the 'CI/CD Pipeline'.

```
1  name: CI/CD Pipeline
2
3  on:
4    push:
5      branches: [ master ]
6    pull_request:
7      branches: [ master ]
8
9  jobs:
10    build-backend:
11      runs-on: ubuntu-latest
12
13      steps:
14        - uses: actions/checkout@v3
15
16        - name: Setup .NET Core
17          uses: actions/setup-dotnet@v3
18          with:
19            dotnet-version: '6.0.x' # Altere para a versão do .NET Core que você está usando
20
21        - name: Restore dependencies
22          run: dotnet restore Server/Server
23
24        - name: Build
25          run: dotnet build Server/Server -c Release --no-restore
26
```

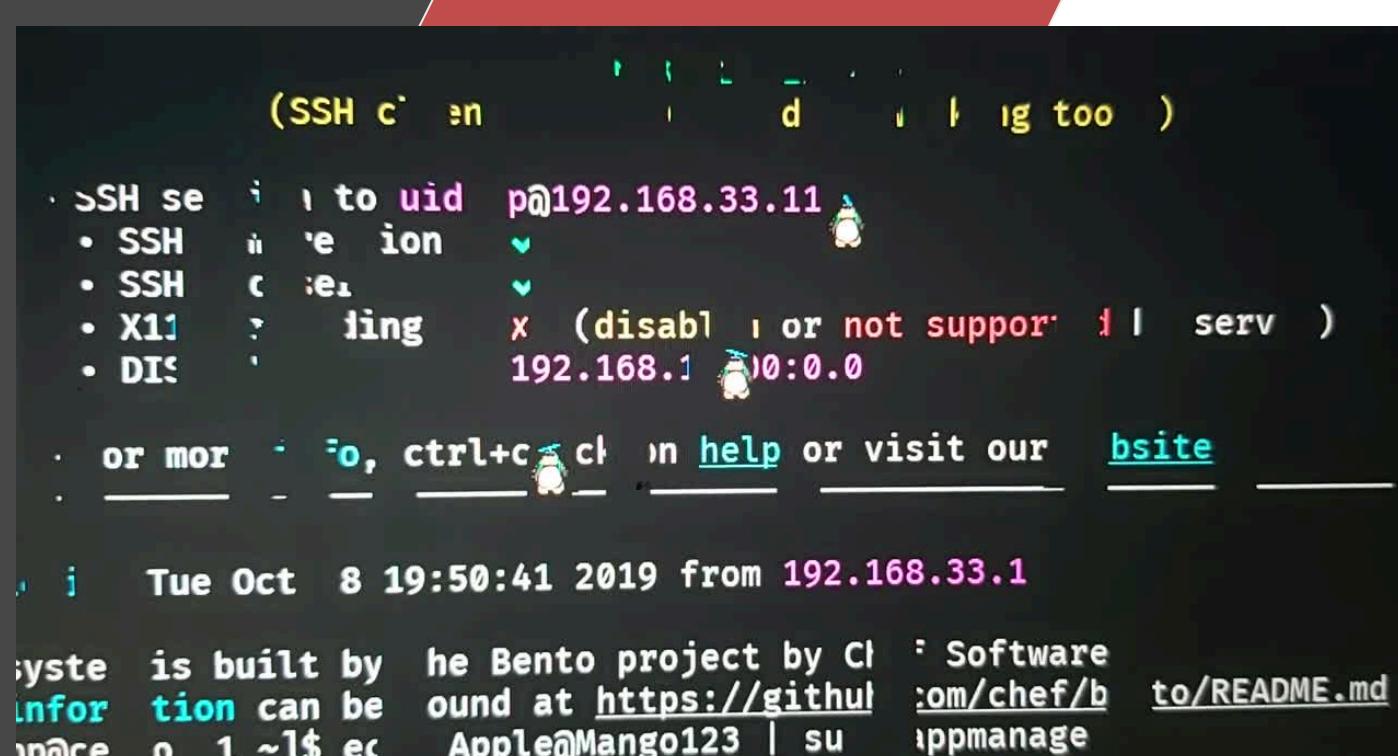
- Compiling developer code and creating automated deployment processes.
- Using GitHub Actions to automate build and deployment with a single button.
- Uses YAML code to define and provision infrastructure, making it consistent and version-controlled.
- Deployment pipelines define the sequence of automated steps, from building the application to deploying it.
- Artifacts are built and uploaded to the server.

Server configuration

```
linuxtldr@linux:~$ systemctl cat boot-efi.mount
# /run/systemd/generator/boot-efi.mount
# Automatically generated by systemd-fstab-generator

[Unit]
Documentation=man:fstab(5) man:systemd-fstab-generator(8)
SourcePath=/etc/fstab
Before=local-fs.target
Requires=systemd-fsck@dev-disk-by\x2duuid-BC1D\x2d7F85.service
After=systemd-fsck@dev-disk-by\x2duuid-BC1D\x2d7F85.service
After=blockdev@dev-disk-by\x2duuid-BC1D\x2d7F85.target

[Mount]
Where=/boot/efi
What=/dev/disk/by-uuid/BC1D-7F85
Type=vfat
Options=umask=0077
linuxtldr@linux:~$
```



Getting the Server Ready:

- Preparing and configuring server environments for service and UI deployment.
- Setting up unit file configuration (systemd) for the service to run as a background process.
- Configure proxy servers to route requests between the frontend and backend.
- Defining the ports and routing rules for the service.
- Ensuring proper reverse proxy setup and routing to the correct service.

Familiarity with vi Commands:

- Using vi editor to make modifications on Linux servers.
- Updating file permissions, copying files etc.

Monitoring server

```
[kubelet-check] It seems like the kubelet isn't running or healthy.
[kubelet-check] The HTTP call equal to 'curl -sSL http://localhost:10248/healthz' failed with error: Get "http://localhost:10248/healthz": dial tcp 127.0.0.1:10248: connect: connection refused.
[kubelet-check] It seems like the kubelet isn't running or healthy.
[kubelet-check] The HTTP call equal to 'curl -sSL http://localhost:10248/healthz' failed with error: Get "http://localhost:10248/healthz": dial tcp 127.0.0.1:10248: connect: connection refused.
[kubelet-check] It seems like the kubelet isn't running or healthy.
[kubelet-check] The HTTP call equal to 'curl -sSL http://localhost:10248/healthz' failed with error: Get "http://localhost:10248/healthz": dial tcp 127.0.0.1:10248: connect: connection refused.
[kubelet-check] It seems like the kubelet isn't running or healthy.
[kubelet-check] The HTTP call equal to 'curl -sSL http://localhost:10248/healthz' failed with error: Get "http://localhost:10248/healthz": dial tcp 127.0.0.1:10248: connect: connection refused.

Unfortunately, an error has occurred:
timed out waiting for the condition

This error is likely caused by:
- The kubelet is not running
- The kubelet is unhealthy due to a misconfiguration of the node in some way (required cgroups disabled)

If you are on a systemd-powered system, you can try to troubleshoot the error with the following commands:
- 'systemctl status kubelet'
- 'journalctl -xeu kubelet'

Additionally, a control plane component may have crashed or exited when started by the container runtime.
To troubleshoot, list all containers using your preferred container runtimes CLI.
Here is one example how you may list all running Kubernetes containers by using crictl:
- 'crictl --runtime-endpoint unix:///var/run/containerd/containerd.sock ps -a | grep kube | grep -v pause'
Once you have found the failing container, you can inspect its logs with:
- 'crictl --runtime-endpoint unix:///var/run/containerd/containerd.sock logs CONTAINERID'
error execution phase wait-control-plane: couldn't initialize a Kubernetes cluster
To see the stack trace of this error execute with --v=5 or higher
ubuntu@ip-172-31-0-199:~$
```

Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>



Monitoring Services

- Ensuring services are Healthy:
- Regularly checking the health status of all running services.
- Viewing service logs to troubleshoot issues and ensure proper functionality.
- Zoom notification integration

Firewall configuration

- Controls incoming and outgoing network traffic based on security policies
- Open ports and IPs to integrate tools like other team API calls or network services

Positive Aspects:

- Excited to learn new tools and concepts, such as server setup, vi commands, GitHub Actions, and nginx configurations.

Challenges:

- Sometimes stressful because the server is a very important part of the whole software development pipeline and any careless mistake may burden my team
- Learning to manage pressure and keep services alive was a crucial part of the experience.

Special thanks to :

Mr Foo Lim Dick



Mr Soo Yong Yao





THANK YOU

Kerana Tuhan untuk Manusia