LEON PRASETYA MULYA

DEVOPS ENGINNER

SKILLS

Docker
Jenkins
Prometheus
AWS Cloud
Terraform
Ansible
Kubernetes
Linux
PHP
UI/UX Design
Python
Bash

CONTACT ME

087763295677
leonprasetya1@gmail.com
linkedin.com/in/leonprasetya
Trenggalek, Jawa Timur

PERSONAL PROFILE

I am a DevOps Enginner who like learn new things. I am also a Linux Enthusiast, familiar with Docker, Jenkins and Cloud Services, i can write code too especially PHP and Python.

EDUCATION HISTORY

Training

Dumbways.id Bootcamp Devops Batch 1
OCTOBER- NOVEMBER 2020

- Setup AWS Server, including instances, security groups, VPC, Subnets, etc.
- Learning about Docker, Ansible, Prometheus Monitoring and CI/CD using Jenkins.
- Deployment Node.JS app into server.
- Learning about security server.

Formal Educations

UNP Kediri
Bachelor's Degree of Information Technology | 2016 - 2020

- Learning about Python, PHP and MySQL.
- Learning about system analyst.
- Member of Abimanyu Robotic Club.

SMKN 2 Trenggalek
Software Enginnering | 2013 - 2016

- Learning about Basic C++, PHP and MySQL.
- Learning about system analyst.

WORK EXPERIENCE

Web Developer at PT. Gita Pratama Nusantara
Intership | FEBRUARY - MARCH 2019

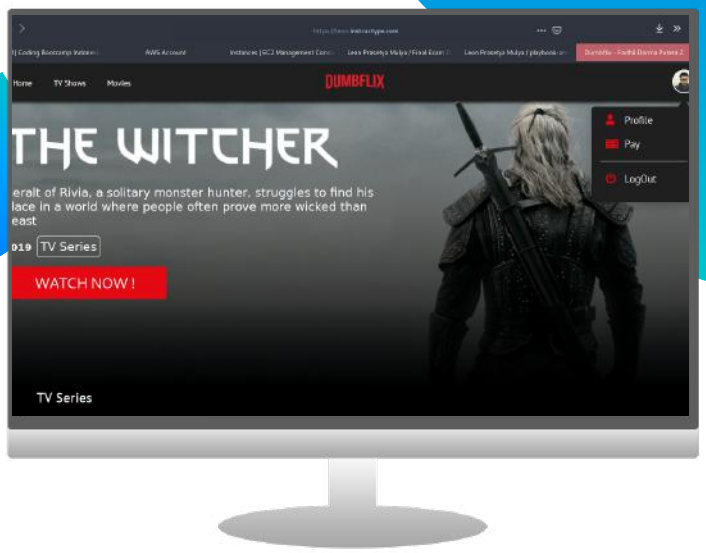
- Make improvements in User Interface on Travel Agent Website using Bootstrap.
- Build a web system using CodeIgniter.

PT. Pranala Sistem ITS
Intership | JUNE - SEPTEMBER 2015

- Colecting data about education games.
- Learning about QT Programming language.

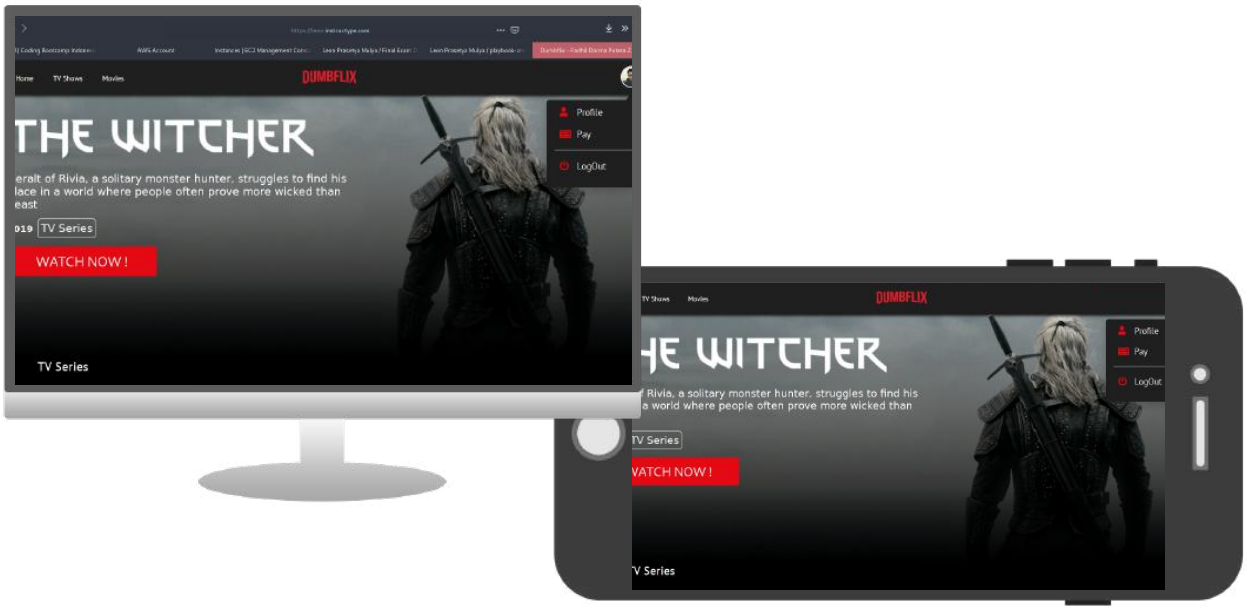
Deployment Node.JS Application

Dumb Flix



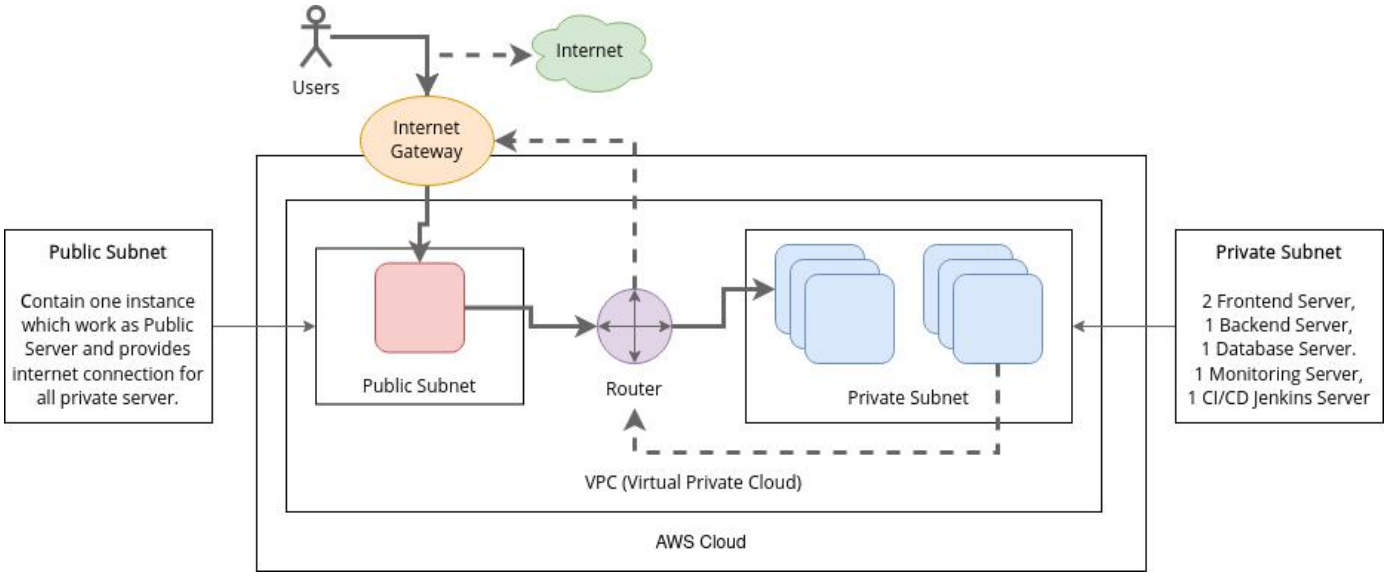
About the Application

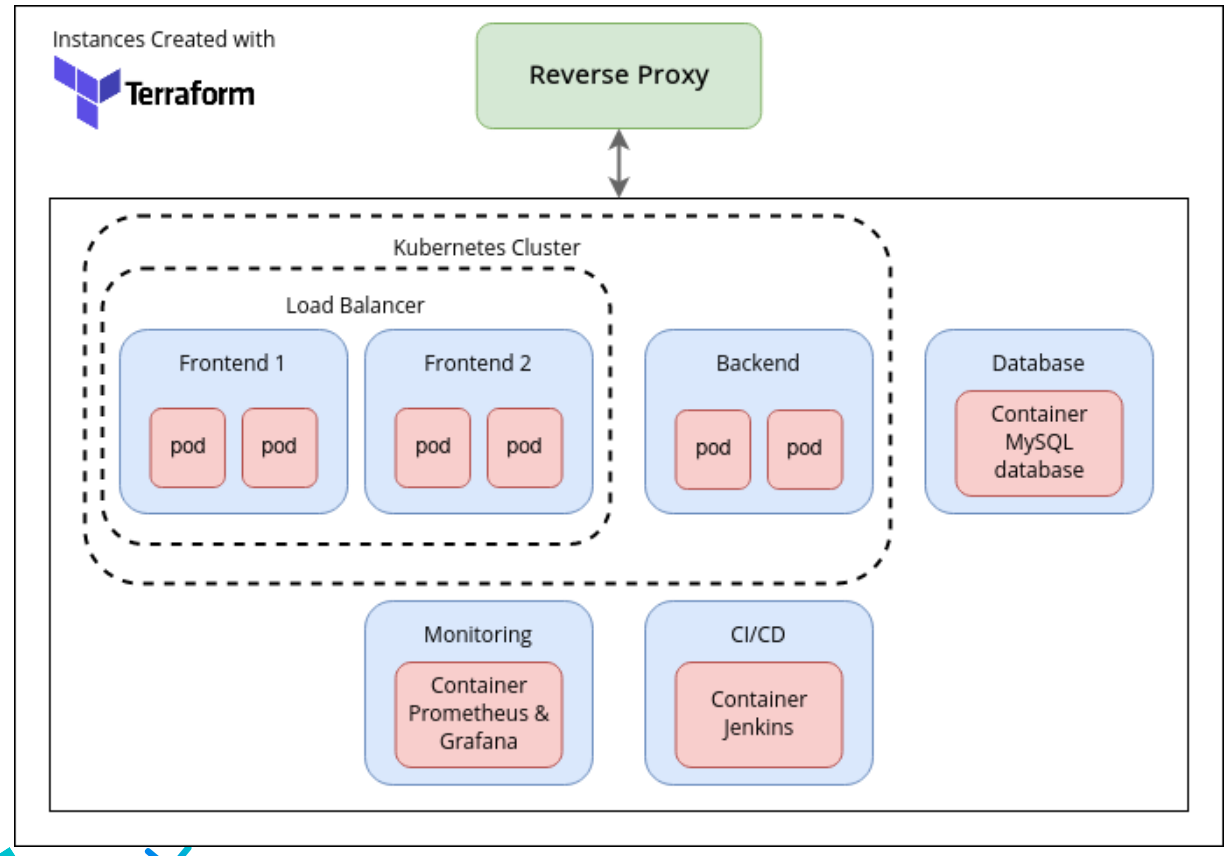
Dumbflix is film streaming application. user can watch film after register & subscribe. This application can be accessed from mobile or desktop.



Architecture

Application running at AWS Cloud. Implementing load balancing for frontend server, the goal is the server will share the load and stabilize server and if one server down the others can work as a backup server. Server Creations I use Terraform and in the Application Deployment I use Kubernetes. All applications are packed in container using Docker. Automated update using CI/CD Jenkins. Server monitored by Prometheus and Grafana.





Setup Server

Setup Server will be executed by Terraform, the reason is, server can be controled automatically by just add a simple configurations.

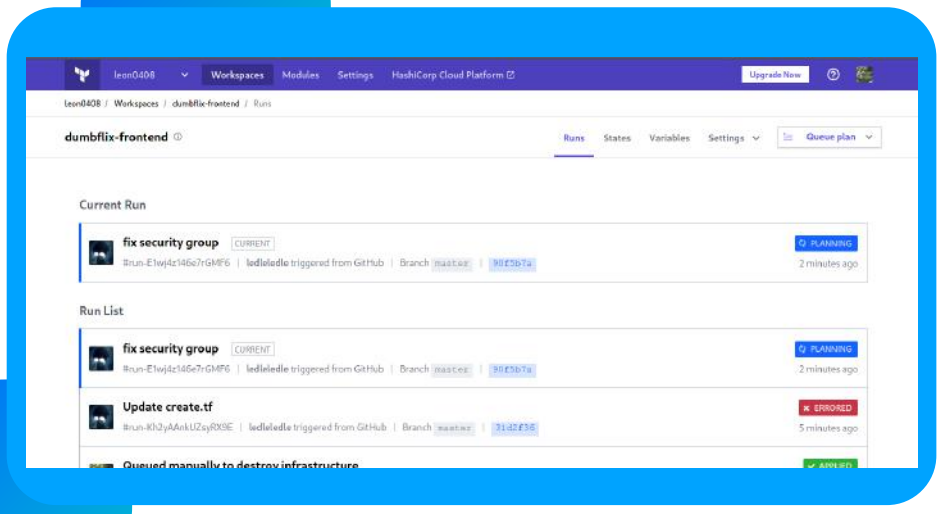
```
resource "aws_instance" "public" {
  ami = "ami-00ddb0e5626798373"
  instance_type = "t2.micro"
  associate_public_ip_address = false
  source_dest_check = false
  key_name = "bruh"
  subnet_id = "subnet-05b40422a2186ed88"
  security_groups = ["sg-0b2f616c069dee233"]
  tags = {
    Name = "node-1"
  }
}
```

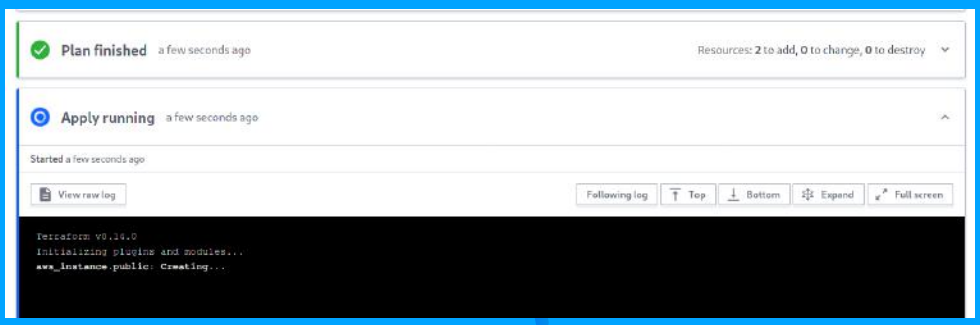
Set the requirements

Terraform will need configuration files to determine the work to be done. The configuration file will be stored in Github.

Run the configurations

Terraform will automatically create a plan for server. User just have to confirm a server changes.





Logs

Terraform will provide logs file.

Setup Server Requirements

Server will need various application and configurations, in this process **Ansible** will take part to help setup requirements for servers.

```
- hosts: jenkins
  tasks:
  - name: jenkins volume dir
    file:
      path: /home/jenkins/jenkins_home
      state: directory
      owner: 1000
      group: 1000

  - name: Pull jenkins
    docker_image:
      name: jenkins/jenkins:lts
      pull: yes

  - name: Container jenkins
    docker_container:
      name: jenkins
      image: jenkins/jenkins
      ports:
        - 8080:8080
        - 50000:50000
      volumes: /home/jenkins/jenkins_home:/var/jenkins_home
```

Set the requirements

Ansible will need configuration files to determine the work to be done. For example, I can install new packages or just update and upgrade system.

```
ok: [172.31.26.82]
ok: [172.31.80.185]
ok: [172.31.80.103]
ok: [172.31.80.7]
ok: [172.31.80.36]
ok: [172.31.80.14]
ok: [172.31.80.213]

TASK [Update & Upgrade] ********************************************************************************************
changed: [172.31.80.103]
changed: [172.31.80.7]
changed: [172.31.26.82]
changed: [172.31.80.185]
changed: [172.31.80.36]
changed: [172.31.80.14]
changed: [172.31.80.213]

TASK [Install Req Docker] ******************************************************************************************
changed: [172.31.26.82]
changed: [172.31.80.103]
changed: [172.31.80.7]
changed: [172.31.80.36]
changed: [172.31.80.185]
changed: [172.31.80.14]
changed: [172.31.80.213]

TASK [GPG Key Docker] **********************************************************************************************
changed: [172.31.80.7]
changed: [172.31.80.36]
changed: [172.31.80.185]
changed: [172.31.80.103]
changed: [172.31.26.82]
changed: [172.31.80.14]
changed: [172.31.80.213]

TASK [Repo Docker] *************************************************************************************************
changed: [172.31.80.103]
changed: [172.31.26.82]
changed: [172.31.80.7]
changed: [172.31.80.185]
changed: [172.31.80.36]
changed: [172.31.80.14]
changed: [172.31.80.213]

TASK [Install Docker] **********************************************************************************************
changed: [172.31.80.103]
changed: [172.31.80.7]
changed: [172.31.80.185]
changed: [172.31.80.36]
changed: [172.31.26.82]
changed: [172.31.80.14]
changed: [172.31.80.213]
```

Run the configurations

The created configurations will be executed to apply the server requirements.

Application Deployments

I use **Kubernetes** for application deployments, Kubernetes will choose server with lowest resources automatically for deployment application containers.

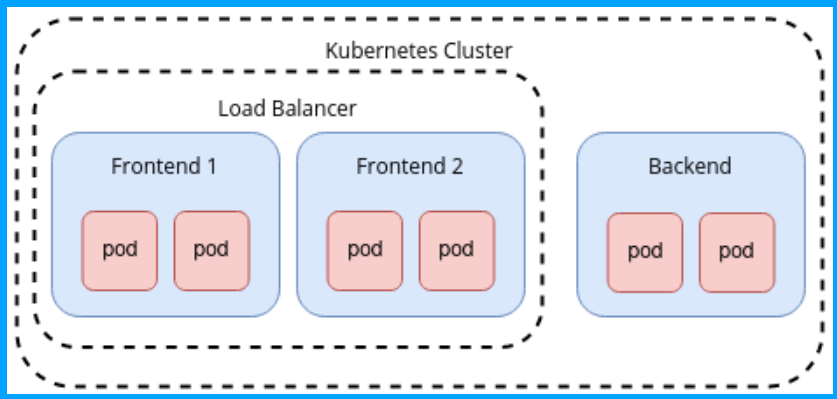
```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      name: nginx
  template:
    metadata:
      name: nginx
      labels:
        name: nginx
    spec:
      containers:
        - name: nginx
          image: nginx
          ports:
            - containerPort: 80
```

Set the requirements

Kubernetes will need configuration files to determine the work to be done. Configuration contain how many pods to deploy and which ports to deploy.

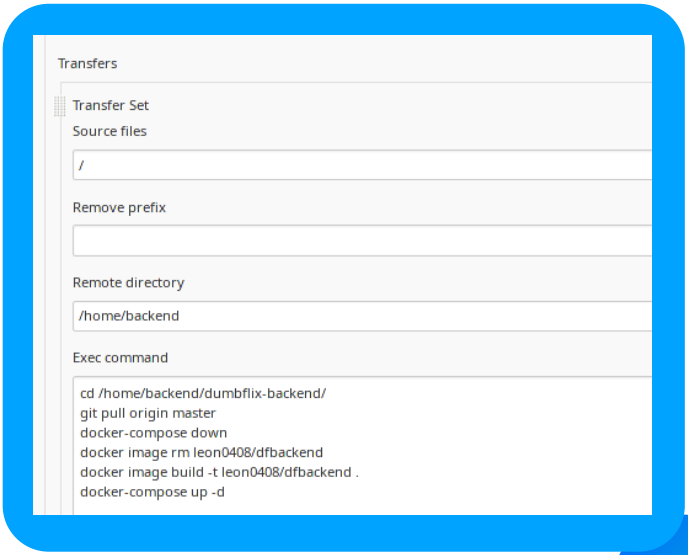
Run the configurations

Kubernetes will choose server with lowest resources automaticallyfor deployment application containers.



CI/CD

Applications code will be continuously updated and requires automatic deployment, deployment will be triggered by change of source code.

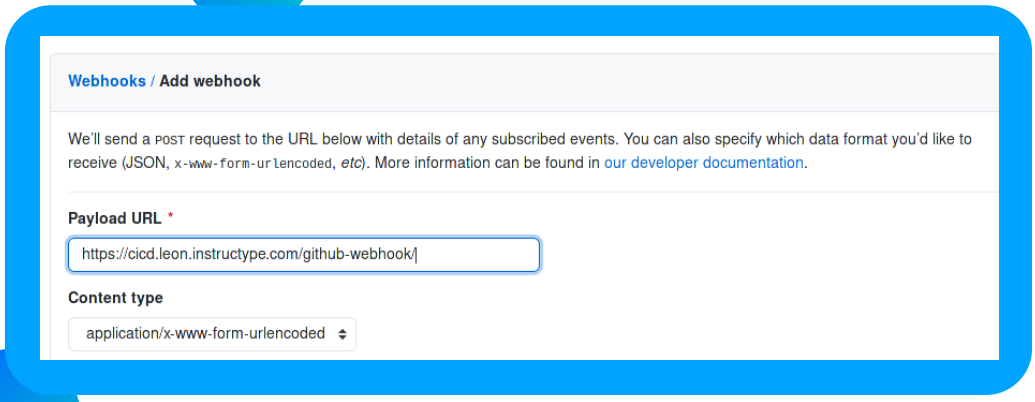


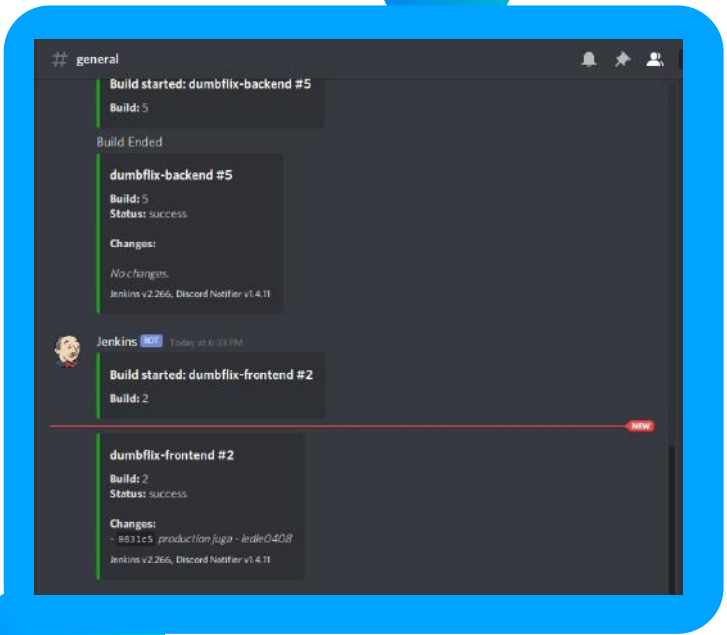
Jenkins Jobs

The job type used in this project is the Jenkins **Freestyle project** which will trigger the Github repository for the **dumbsound** application.

Webhook

Integrate Github repository with Jenkins Job. So that every source code changed. Webhook will trigger Jenkins Job.



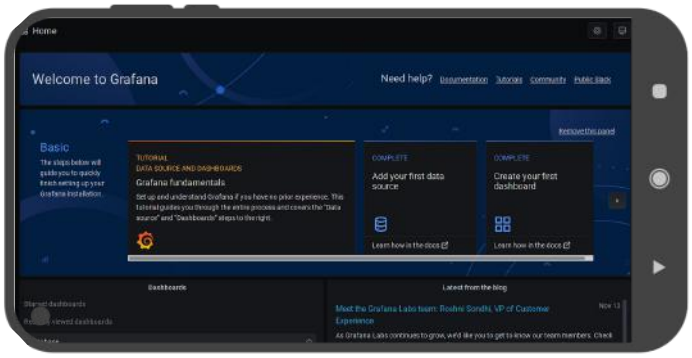


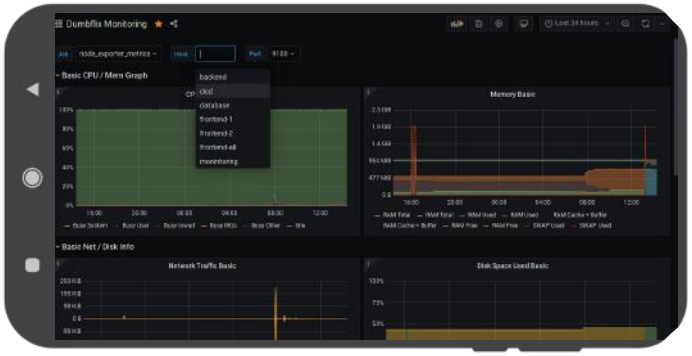
Notifications

Jenkins will send **Discord** notification every Building a Job has started. If the build process is completed Jenkins will send a message about the status (Success or Failed).

Monitoring

Server will be monitored monitored both CPU, memory and disk using **Prometheus** and for better visualisation **Grafana** will be the best choice.





Tools Used

docker git ANSIBLE Grafana CLOUDFLARE Let's Encrypt Terraform aws kubernetes Jenkins

My Repository

GitLab

More detailed documentation

https://gitlab.com/ledleledle/trainning-dumbways.id