Instalasi Jenkins di Server Ubuntu

1. Lakukan update package dengan perintah

```
$ sudo apt-get update
```

2. Install java

```
$ sudo apt install openjdk-11-jdk
```

3. Add Jenkins repository

```
$ curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee
/usr/share/keyrings/jenkins-keyring.asc > /dev/null

$ echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]
https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list >
/dev/null
```

4. Lakukan update repository

```
$ sudo apt update
```

5. Install Jenkins

```
$ sudo apt install Jenkins
```

6. Start Jenkins dan lakukan enable system Jenkins agar langsung berjalan Ketika server di restart

```
$ sudo systemctl start jenkins

$ sudo systemctl enable jenkins.service
```

7. Akses Jenkins pada browser dengan port 8080 contoh http://192.168.1.3:8080 (ip private local host)
8. Untuk mendapatkan password administrator print file password dengan perintah

```
$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

   Paste password yang di dapat di password administrator

9. Pilih install suggested plugin
10. Tunggu prosesnya sampai selsai
11. Isi account admin user yang baru
12. Pada Jenkins url isi dengan http://192.168.1.3:8080

13. Save and finish
14. Lakukan login menggunakan admin user yang baru

Install terraform on ubuntu

1. Install package

```
$ sudo apt-get update && sudo apt-get install -y gnupg software-properties-common
```

2. Install the HashiCorp GPG key.

```
$ wget -O- https://apt.releases.hashicorp.com/gpg | \
gpg --dearmor | \
sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg
```

3. Verify the key's fingerprint.

```
$ gpg --no-default-keyring \
--keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg \
--fingerprint
```

4. Add the official HashiCorp repository to your system

```
$ echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | \
sudo tee /etc/apt/sources.list.d/hashicorp.list
```

5. Download the package information from HashiCorp.

```
$ sudo apt update
```

6. Install Terraform from the new repository.

```
$ sudo apt-get install terraform
```

7. Verify that the installation worked

```
$ terraform -help
```

Install AWS CLI

1. Download file installasi

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

2. Unzip installer

```
$ unzip awscliv2.zip
```

3. Run the install program

```
$ sudo ./aws/install
```

4. Confirm the installation

```
$ aws --version
```

Install kubectl

1. Download the latest release

```
$    curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

2. Install kubectl

```
$ sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

3. Cek kubectl jika berhasil di install

```
$ kubectl version --client
```

Deploy Terraform

1. Clone repository https://github.com/muhalisurad/tes-tlab.git

```
$ git clone https://github.com/muhalisurad/tes-tlab.git
```

2. Masuk ke dalam directory terraform-eks

```
$ cd tes/terraform-eks
```

3. Jalankan perintah terraform init

```
$ terraform init
```

   Tunggu prosesnya sampai selesai

4. Cek konfigurasi file terraform dengan perintah

```
$ terraform validate
```

   Pastikan hasil output “success! The configuration is valid.”,

5. Lakukan provision resources terraform

```
$ terraform apply -auto-approve
```

   Tunggu prosesnya sampai selesai bisa menghabiskan wakti sekitar 15 – 20 menit dalam membuat resource 3 node dan 1 cluster

6. Setelah selesai lakukan konfigurasi aws cli dengan perintah

```
$ aws configure
```

   Masukan aws_access_key_id dan aws_secret_access_key yang didapatkan dari security credential access key di aws console

7. Lakukan update kubectl agar terkoneksi dengan cluster yang telah di buat dengan perintah

```
$ aws eks update-kubeconfig --region ap-southeast-3 --name eks-tes
```

8. Jalankan perintah kubectl get nodes untuk memastikan sudah terkoneksi dengan cluster yang telah dibuat

```
$ kubectl get nodes
```

Install postgres on Kubernetes

1. Create namespace dev

```
$ kubectl create namespace dev
```

2. Dari folder git hasil clone Masuk ke dalam folder postgres-kubernetes

```
$ cd postgres-kubernetes
```

3. Jalankan perintah kubectl apply untuk membuat deployment, service dan configmap

```
$ kubectl apply -f postgres-configmap.yaml -n dev
$ kubectl apply -f postgres-deployment.yaml -n dev
$ kubectl apply -f postgres-service -n dev
$ kubectl apply -f postgres-storage.yaml -n dev
```

Tunggu prosesnya sampai selesai

Install redis on Kubernetes

1. Masuk ke dalam folder redis-kubernetes

```
$ cd redis-kubernetes
```

2. Jalankan perintah kubectl apply untuk membuat deployment,service dan configmap

```
$ kubectl apply -f redis-conf.yaml -n dev
$ kubectl apply -f redis-pod.yaml -n dev
$ kubectl apply -f redis-service.yaml -n dev
```

Tunggu prosesnya sampai selesai

Running Jenkins file dan ingress

Untuk service results-app,vote-worker dan web-vote-app di edit env postgres dan redis nya dan disesuaikan fe nya agar mengarah ke cluster ip postgre, redis, results-app dan web vote yang telah dibuat, contoh seperti ini di vote worker

```
Class.forName("org.postgresql.Driver");
String url = "jdbc:postgresql://172.20.24.27:5432/postgres";
```

Sebelum melakukan CI terlebih dahulu deploy ingress-nginx agar dapat diakses dari luar

Masuk ke dalam folder git yang telah di clone, jalanken perintah di bawah ini

```
$ kubectl apply -f deploy-ingress.yaml

$ kubectl apply -f ingress.yaml -n dev
```

Berikut Jenkins file untuk melakukan CI, di sini saya menggunakan ecr sebagai repository image, disini saya sudah configurasi aws cli yang mengarah ke cluster Kubernetes aws dan menginstall docker di user Jenkins serta setting kubectl agar terkoneksi dengan cluster yang telah di buat

```
pipeline {
   agent any

   stages {
      stage('start'){
         steps{
            git branch: 'main', url: https://github.com/muhalisurad/tes-tlab.git'
         }
      }
      stage('build and push to ecr') {
         steps {
            sh ‘aws ecr get-login-password --region ap-southeast-3 | docker login --username AWS --
password-stdin 114988434576.dkr.ecr.ap-southeast-3.amazonaws.com
            sh ‘docker build -t results-app -f application/results-app/Dockerfile .’
            sh ‘docker tag result-app:latest 114988434576.dkr.ecr.ap-southeast-
3.amazonaws.com/results-app:latest’
            sh ‘docker push 114988434576.dkr.ecr.ap-southeast-3.amazonaws.com/results-app:latest’

            sh ‘docker build -t vote-worker -f application/vote-worker/Dockerfile .’
            sh ‘docker tag vote-worker:latest 114988434576.dkr.ecr.ap-southeast-
3.amazonaws.com/vote-worker:latest’
            sh ‘docker push 114988434576.dkr.ecr.ap-southeast-3.amazonaws.com/vote-worker:latest’

            sh ‘docker build -t web-vote-app -f application/web-vote-app/Dockerfile .’
            sh ‘docker tag web-vote-app:latest 114988434576.dkr.ecr.ap-southeast-
3.amazonaws.com/web-vote-app:latest’
```

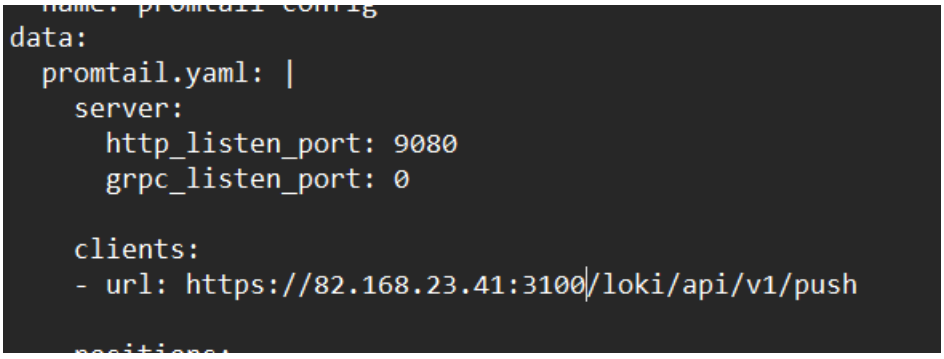
```
            sh 'docker push 114988434576.dkr.ecr.ap-southeast-3.amazonaws.com/web-vote-app:latest'

      stage('deploy to kubernetes') {
        steps {
          sh 'kubectl apply -f application/results-app/deployment.yaml -n dev
          sh 'kubectl apply -f application/results-app/service.yaml -n dev
          sh 'kubectl apply -f application/vote-worker/deployment.yaml -n dev
          sh 'kubectl apply -f application/vote-worker/service.yaml -n dev
          sh 'kubectl apply -f application/web-vote-app/deployment.yaml -n dev
          sh 'kubectl apply -f application/web-vote-app/service.yaml -n dev

        }
      }
      stage('Cleaning') {
        steps {
          sh 'docker rmi  result-app:latest 114988434576.dkr.ecr.ap-southeast-
3.amazonaws.com/results-app:latest'
          sh 'docker rmi vote-worker:latest 114988434576.dkr.ecr.ap-southeast-
3.amazonaws.com/vote-worker:latest'
          sh 'docker rmi web-vote-app:latest 114988434576.dkr.ecr.ap-southeast-
3.amazonaws.com/web-vote-app:latest
          sh 'docker rmi $(docker images -a -q)'
        }
      }
    }
}
```

Install promtail

Promtail digunakan sebagai agent yang akan mengirim semua log pod,node ke server loki yang di install di server Grafana, berikut Langkah deploy, sebelum melakukan deploy pastikan url server loki sudah diganti seperti contoh di bawah ini mengarah ke ip server ubuntu loki



1. Masuk ke dalam direktori loki-promtail-kubernetes

```
$ cd loki-promtail-kubernetes
```

2. Deploy daemonset promtail ke Kubernetes

```
$ kubectl apply -f daemonset-promtail.yaml -n dev
```

Tunggu prosesnya sampai selesai

Install node exporter

Node exporter digunakan sebagai agent untuk mengirim semua resource node yang ada di cluster, kemudian resource yang di tangkap akan di ambil oleh server Prometheus server, berikut Langkah instalasinya

1. Masuk ke dalam direktori node-exporter-kubernetes

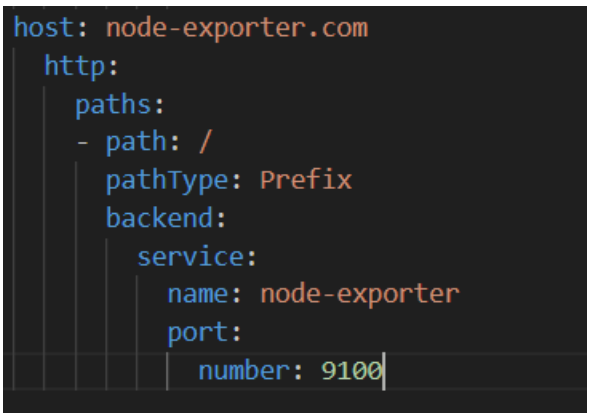
```
$ cd node-exporter-kubernetes
```

2. Deploy daemonset node-exporter ke Kubernetes

```
$ kubectl apply -f daemonset.yaml
```

Tunggu prosesnya sampai selesai

3. Untuk mengakses dari public sudah di setting di ingress lewat domain



Install Prometheus

1. Update package

```
sudo apt update
```

2. Membuat Prometehus Users dan node_exporter

```
sudo useradd --no-create-home --shell /bin/false prometheus
```

3. Membuat Prometheus Directory

```
sudo mkdir /etc/Prometheus
sudo mkdir /var/lib/Prometheus
```

4. Download Prometheus

```
Wget https://github.com/prometheus/prometheus/releases/download/v2.30.3/prometheus-2.30.3.linux-amd64.tar.gz
```

5. Extract Prometheus ang telah di download

```
tar xzf prometheus-2.30.3.linux-amd64.tar.gz
```

6. Copy Prometheus dan promtool binary file ke /usr/local/bin

```
cp prometheus-2.30.3.linux-amd64/{prometheus,promtool} /usr/local/bin/
```

7. Change the ownership file dengan menggunakan command

```
chown prometheus:prometheus /usr/local/bin/{prometheus,promtool}
```

8. Kemudian copy console dan console_libraries directory ke Prometheus configuration director /etc/Prometheus

```
cp -r prometheus-2.30.3.linux-amd64/{consoles,console_libraries} /etc/prometheus/
```

9. Membuat Prometheus Configuration file

```
sudo nano /etc/prometheus/prometheus.yml
```

Copy dan paste script di bawah ini

```
# my global config
global:
  scrape_interval:     15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
```

```
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      - “alert-exporter.com”

# Load rules once and periodically evaluate them according to the global 'evaluation_interval’.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: 'prometheus'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
    - targets: ['94.74.95.229:9090']

     # menambahkan node_exporter di cluster kubernetes
    - job_name: 'node_exporter'
    scrape_interval: 5s
    static_configs:
      - targets: ['node-exporter.com', '94.74.95.229:9100']
```

10. Set user dan group ownership Prometheus configuration director, /etc/Prometheus to Prometheus

```
chown -R prometheus:prometheus /etc/prometheus
```

11. Set user dan group ownership Prometheus data director, /var/lib/Prometheus/ to Prometheus

```
chown prometheus:prometheus /var/lib/prometheus
```

12. Lakukan pengecekan configuration Prometheus file

```
prometheus --config.file=/etc/prometheus/prometheus.yml
```

13. Open port Prometheus

```
ufw allow 9090/tcp
```

14. Membuat Prometheus system service file

```
Sudo nano /etc/systemd/system/prometheus.service
```

Paste code di bawah ini

```
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus \
  --config.file /etc/prometheus/prometheus.yml \
  --storage.tsdb.path /var/lib/prometheus/ \
  --web.console.templates=/etc/prometheus/consoles \
  --web.console.libraries=/etc/prometheus/console_libraries

[Install]
WantedBy=multi-user.target
```

15. Kemudian reload system configuration files dan start and enable Prometheus to run on system boot

```
systemctl daemon-reload
systemctl enable --now Prometheus
```

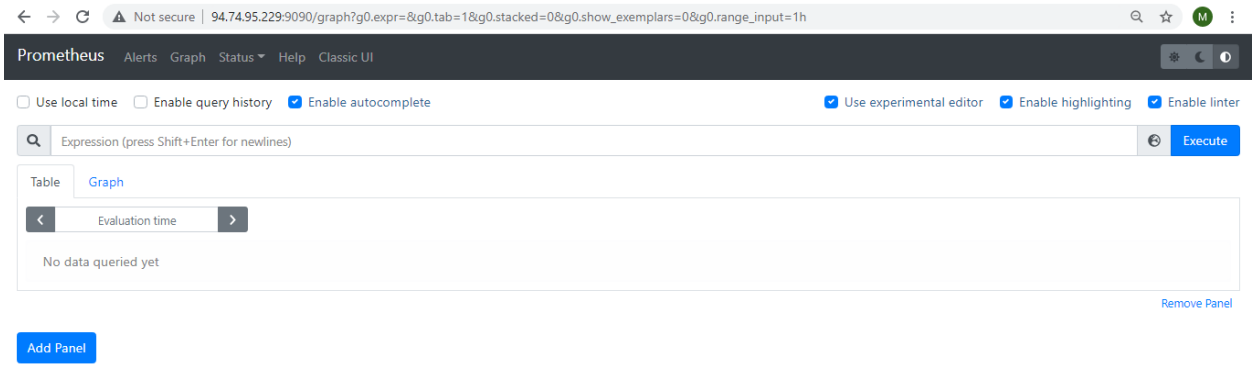
16. Cek status Prometheus service

```
systemctl status Prometheus
```

pastikan Prometheus active (running) seperti info status Prometheus di bawah ini

```
● prometheus.service - prometheus
   Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; vendor pr>
   Active: active (running) since Tue 2021-10-26 12:47:47 CST; 1 day 2h ago
 Main PID: 7172 (prometheus)
    Tasks: 8 (limit: 1073)
   Memory: 96.1M
   CGroup: /system.slice/prometheus.service
           └─7172 /usr/local/bin/prometheus --config.file /etc/prometheus/pro>
```

17. Acces Prometheus menggunakan browser dengan address http://server-IP-or-Hostname:9090



Untuk cek status node cluster kubernetes, navigate to **Status** > **Targets**.

Install Loki

1. Download installer dan setting loki di server ubuntu

```
$ curl -O -L https://github.com/grafana/loki/releases/download/v2.8.2/loki-linux-amd64.zip

$ unzip "loki-linux-amd64.zip"

$ chmod a+x "loki-linux-amd64"

$ sudo mv loki-linux-amd64 /usr/local/bin/loki

$ sudo useradd --system loki

$ sudo mkdir -p /etc/loki /etc/loki/logs

$ sudo curl -o /etc/loki/loki-local-config.yaml -L
https://gist.githubusercontent.com/theLazyCat775/ceaf475fc369e25a2d04501f8a7c0a59/raw/fdf2cbc14d540085b68911972f1eb1f8fa9de908/loki-local-config.yaml

$ sudo chown -R loki: /etc/loki

$ sudo systemctl enable loki.service

$ sudo systemctl start loki.service

$ sudo systemctl status loki.service
```

2. Buat Loki-local-config.yaml untuk konfigurasi server loki

```
$ sudo nano /etc/loki/loki-local-config
```

Paste konfigurasi di bawah ini di file konfigurasi yang dibuat

```
auth_enabled: false

server:
  http_listen_port: 3100
  grpc_listen_port: 9096

common:
  instance_addr: 127.0.0.1
  path_prefix: /tmp/loki
  storage:
    filesystem:
      chunks_directory: /tmp/loki/chunks
      rules_directory: /tmp/loki/rules
  replication_factor: 1
  ring:
    kvstore:
      store: inmemory

query_range:
  results_cache:
    cache:
      embedded_cache:
        enabled: true
        max_size_mb: 100

schema_config:
  configs:
    - from: 2020-10-24
      store: boltdb-shipper
      object_store: filesystem
      schema: v11
      index:
        prefix: index_
        period: 24h

ruler:
  alertmanager_url: http://localhost:9093

# By default, Loki will send anonymous, but uniquely-identifiable usage and configuration
# analytics to Grafana Labs. These statistics are sent to https://stats.grafana.org/
#
# Statistics help us better understand how Loki is used, and they show us performance
```

```
# levels for most users. This helps us prioritize features and documentation.
# For more information on what's sent, look at
# https://github.com/grafana/loki/blob/main/pkg/usagestats/stats.go
# Refer to the buildReport method to see what goes into a report.
#
# If you would like to disable reporting, uncomment the following lines:
#analytics:
#  reporting_enabled: false
```

3. Buat service loki agar berjalan di server

```
$ sudo nano /etc/system/system/loki.service
```

Paste konfigurasi di bawah ini di file yang telah dibuat

```
[Unit]
Description=Loki service
After=network.target

[Service]
Type=simple
User=loki
ExecStart=/usr/local/bin/loki -config.file /etc/loki/loki-local-config.yaml
Restart=on-failure
RestartSec=20
StandardOutput=append:/etc/loki/logs/loki.log
StandardError=append:/etc/loki/logs/loki.log

[Install]
WantedBy=multi-user.target
```

4. Jalankan service yang telah dibuat

```
$ sudo systemctl start loki.service
```

Install Grafana

1. Download Grafana

```
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
```

2. Add Grafana repository to your APT sources

```
sudo add-apt-repository "deb https://packages.grafana.com/oss/deb stable main"
```

3. Refresh APT cache

```
sudo apt update
```

4. Install Grafana

```
sudo apt install Grafana
```

5. Setelah Grafana terinstall, gunakan systemctl untuk menjalankan Grafana server

```
sudo systemctl start grafana-server
```

6. Cek status Grafana apakah sudah berjalan

```
sudo systemctl status grafana-server
```

pastikan outputnya active (running) seperti di bawah ini

```
Output
● grafana-server.service - Grafana instance
    Loaded: loaded (/lib/systemd/system/grafana-server.service; disabled; vendor preset: enabled)
   Active: active (running) since Thu 2020-05-21 08:08:10 UTC; 4s ago
     Docs: http://docs.grafana.org
 Main PID: 15982 (grafana-server)
    Tasks: 7 (limit: 1137)
```

7. Enable service grafana di boot

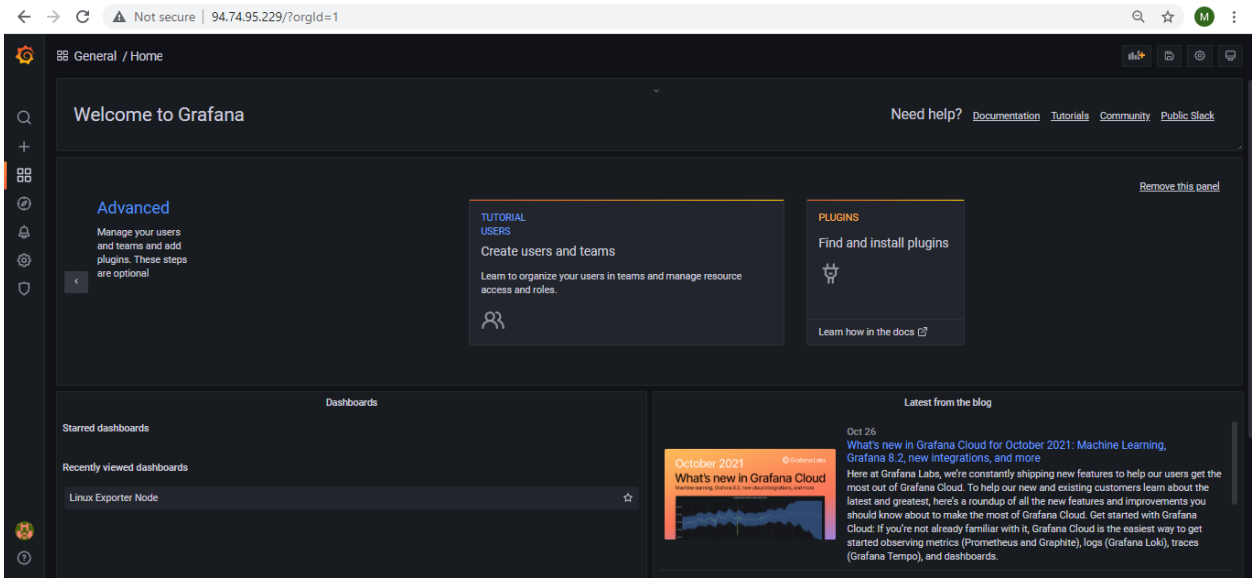
```
sudo systemctl enable grafana-server
```

pastikan outputnyya seperti di bawah ini

```
Output
Synchronizing state of grafana-server.service with SysV service script with /lib/systemd/systemd-sysv-install.
```

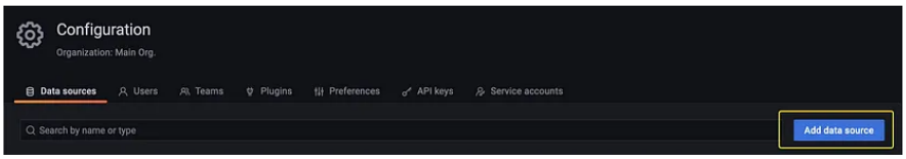
```
Executing: /lib/systemd/systemd-sysv-install enable grafana-server
Created symlink /etc/systemd/system/multi-user.target.wants/grafana-server.service →
/usr/lib/systemd/system/grafana-server.service.
```

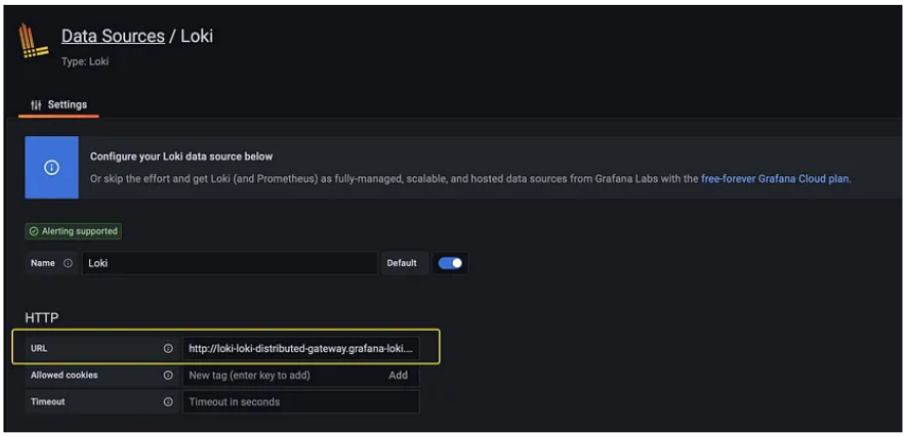
8. Kemudian untuk akses Grafana gunakan browser dan masukan http://ip-server:3000



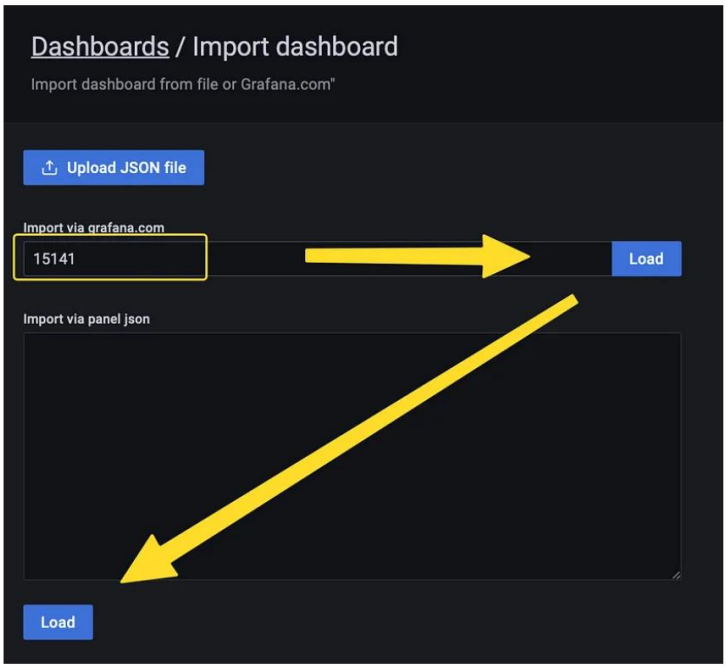
Setting loki Grafana

1. Buat koneksi ke server loki

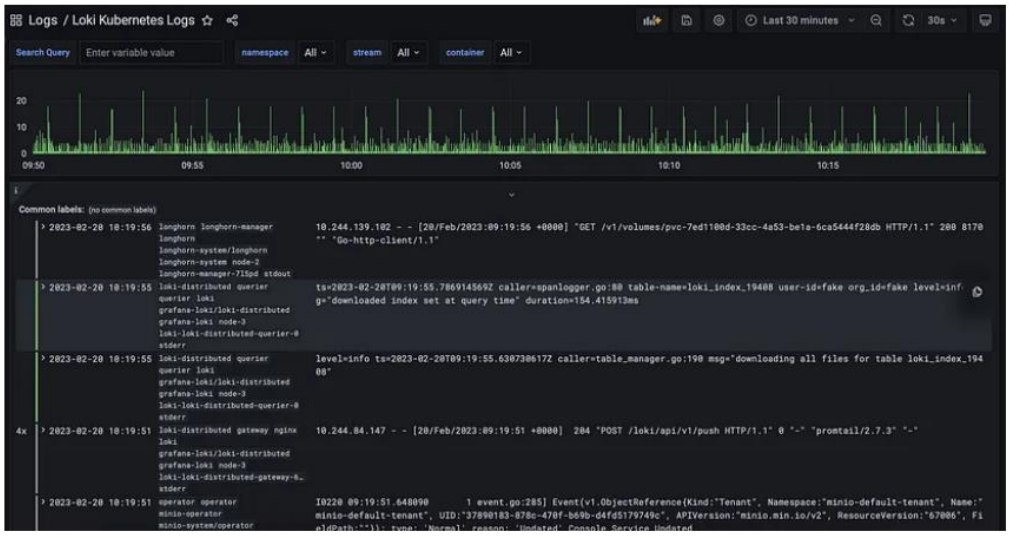




2. Lakukan import dashboard dengan id 15141



3. Berikut contoh tampilan log di Grafana yang telah di import



Install alert manager

1. Masuk ke dalam folder alert-manager-kubernetes

```
$ cd alert-manager-kubernetes
```

2. Lakukan deploy service, deployment dan config map

```
$ kubectl apply -f alertmanagerconfigmap.yaml
$ kubectl apply -f alerttemplateconfigmap.yaml
$ kubectl apply -f deployment.yaml
$ kubectl apply -f service.yaml
```

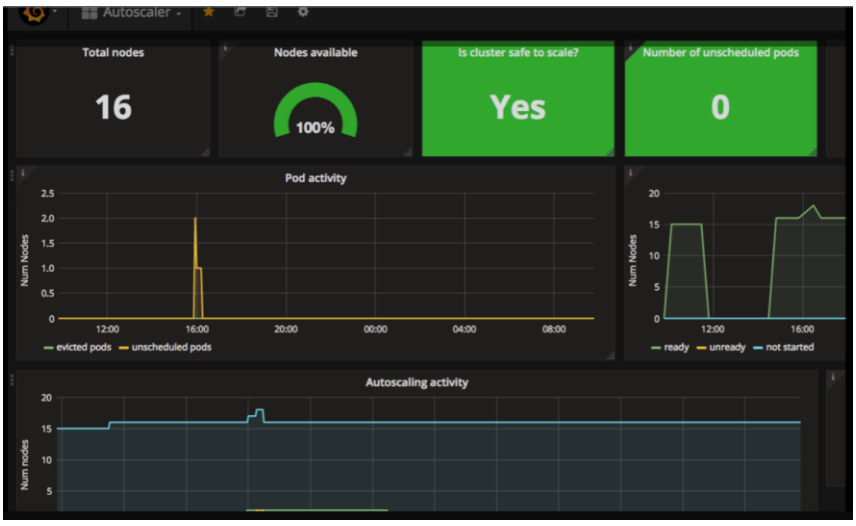
3. Untuk endpointnya sudah di setting di ingress dan sudah di setting di file configurasi Prometheus

```
- host: alert-exporter.com
  http:
    paths:
    - path: /
      pathType: Prefix
      backend:
        service:
          name: alertmanager
          port:
            number: 9300
```

```
# Alertmanager configuration
alerting:
 alertmanagers:
 - static_configs:
   - targets:
     - "alert-exporter.com"
```

Import Grafana dashboard untuk Prometheus

1. Buat koneksi ke server Prometheus seperti contoh di setting loki Grafana
2. Lakukan import id dengan 3831



Import Grafana dashboard untuk alert

1. Buat koneksi ke server Prometheus seperti contoh di setting loki Grafana
2. Lakukan import id dengan 9578