

PROYEK AKHIR
MATA KULIAH PENYEDIAAN LAYANAN DAN AUTOMASI

**Otomatisasi Deployment dan Monitoring Aplikasi Web Sederhana
menggunakan CI/CD Pipeline di Virtual Machine**



Nama Mahasiswa:
Fira Zaha Iklila - 235150207111007
Muhammad Faiz Fauzan - 235150201111061

Dosen:
Widhi Yahya, S.Kom., M.T., M.Sc., Ph.D.

TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
2025

DAFTAR ISI

DAFTAR ISI.....	1
BAB 1 PENDAHULUAN.....	2
1.1 Latar Belakang.....	2
1.2 Rumusan Masalah.....	2
1.3 Tujuan Proyek.....	2
1.4 Manfaat Proyek.....	3
BAB 2 PERANCANGAN SISTEM.....	5
2.1 Gambaran Umum Arsitektur.....	5
2.2 Diagram Alur Sistem.....	5
2.3 Penjelasan Komponen Utamai.....	6
2.4 Alur Kerja CI/CD (Pipeline).....	6
BAB 3 IMPLEMENTASI.....	9
3.1 Progres Implementasi.....	9
3.1.1 Langkah 1: Persiapan Awal.....	9
3.1.2 Langkah 2: Instalasi Software Pendukung di VM Ubuntu.....	9
3.1.3 Langkah 3: Setup Aplikasi Web Sederhana.....	19
3.1.4 Langkah 4: Konfigurasi Jenkins untuk CI/CD.....	19
3.1.5 Langkah 5: Uji Coba Alur CI/CD.....	25
3.1.6 Langkah 6: Konfigurasi Monitoring Metrics.....	27
BAB 4 PEMBAGIAN TUGAS.....	30
4.1 Pembagian Kerja Tim.....	30
BAB 5 EVALUASI.....	31
5.1 Masalah.....	31
5.2 Solusi.....	32
BAB 6 KESIMPULAN.....	33

BAB I

PENDAHULUAN

1.1 Latar Belakang

Automasi merupakan pilar fundamental dalam siklus pengembangan perangkat lunak modern berbasis DevOps. Praktik inti seperti Continuous Integration dan Continuous Deployment (CI/CD) secara drastis meningkatkan efisiensi, mempercepat waktu rilis, dan mengurangi kesalahan manual, yang pada akhirnya meningkatkan keandalan sistem. Kebutuhan untuk menerapkan konsep otomasi ini menjadi semakin krusial dalam lingkungan yang menawarkan skalabilitas dan fleksibilitas dinamis. Oleh karena itu, proyek ini dirancang untuk membangun alur kerja otomasi yang terintegrasi dalam lingkungan virtual machine berbasis Linux, guna mendemonstrasikan secara praktis efektivitas dan efisiensi dari praktik DevOps modern.

1.2 Rumusan Masalah

1. Bagaimana merancang dan mengimplementasikan sebuah sistem deployment aplikasi web yang sepenuhnya otomatis dari repository kode hingga aplikasi berjalan di server?
2. Bagaimana memanfaatkan Git dan Jenkins untuk membangun pipeline Continuous Integration/Continuous Deployment (CI/CD) yang efektif untuk aplikasi web di virtual machine berbasis Linux?
3. Bagaimana memastikan ketersediaan dan mempersiapkan skalabilitas aplikasi web di lingkungan virtual machine menggunakan teknologi seperti Load Balancer dan konsep Autoscaling?
4. Bagaimana melakukan monitoring performa aplikasi dan infrastruktur pendukungnya secara berkelanjutan?

1.3 Tujuan Proyek

1. Mengimplementasikan pipeline CI/CD menggunakan Git dan Jenkins untuk otomatisasi proses build, test, dan deploy aplikasi web sederhana.

2. Melakukan deployment aplikasi web ke infrastruktur virtual machine berbasis Linux yang didukung oleh Application Load Balancer untuk distribusi trafik dan peningkatan ketersediaan.
3. Menganalisis, merencanakan, dan (jika waktu memungkinkan) mengimplementasikan dasar-dasar Autoscaling untuk menyesuaikan kapasitas aplikasi secara otomatis berdasarkan permintaan.
4. Mengimplementasikan sistem monitoring dasar menggunakan Prometheus untuk memantau metrik kunci dari aplikasi dan infrastruktur virtual machine.

1.4 Manfaat Proyek

Proyek ini diharapkan memberikan beberapa manfaat signifikan, antara lain:

- Peningkatan Efisiensi dan Kecepatan Deployment: Dengan pipeline CI/CD yang terotomasi penuh, proses build, test, dan deploy aplikasi menjadi jauh lebih cepat dan efisien, mengurangi waktu time-to-market untuk fitur dan perbaikan.
- Pengurangan Kesalahan Manusia: Automasi meminimalkan intervensi manual, sehingga mengurangi potensi kesalahan yang sering terjadi pada proses deployment tradisional.
- Peningkatan Kualitas Perangkat Lunak: Integrasi dan pengujian yang berkelanjutan memastikan bahwa setiap perubahan kode diuji secara menyeluruh sebelum di-deploy, menghasilkan aplikasi yang lebih stabil dan berkualitas.
- Peningkatan Ketersediaan Aplikasi: Implementasi Application Load Balancer dan dasar-dasar Autoscaling akan memastikan distribusi trafik yang optimal dan kemampuan aplikasi untuk menangani beban kerja yang bervariasi, sehingga meningkatkan ketersediaan layanan.
- Visibilitas dan Pemantauan yang Lebih Baik: Sistem monitoring berbasis Prometheus akan menyediakan wawasan real-time tentang performa aplikasi dan kesehatan infrastruktur virtual machine, memungkinkan identifikasi dan resolusi masalah yang lebih cepat.

- Pembelajaran dan Pengalaman Praktis: Proyek ini akan memberikan pengalaman langsung dalam mengimplementasikan praktik DevOps menggunakan teknologi industri standar seperti Git, Jenkins, Virtual Machine (Linux OS), Application Load Balancer, Autoscaling, dan Prometheus, yang sangat berharga bagi mahasiswa.
- Fondasi untuk Pengembangan Lebih Lanjut: Solusi yang dibangun dapat menjadi fondasi yang kuat untuk pengembangan dan deployment aplikasi yang lebih kompleks di masa depan, menerapkan prinsip-prinsip skalabilitas dan keandalan.

BAB II

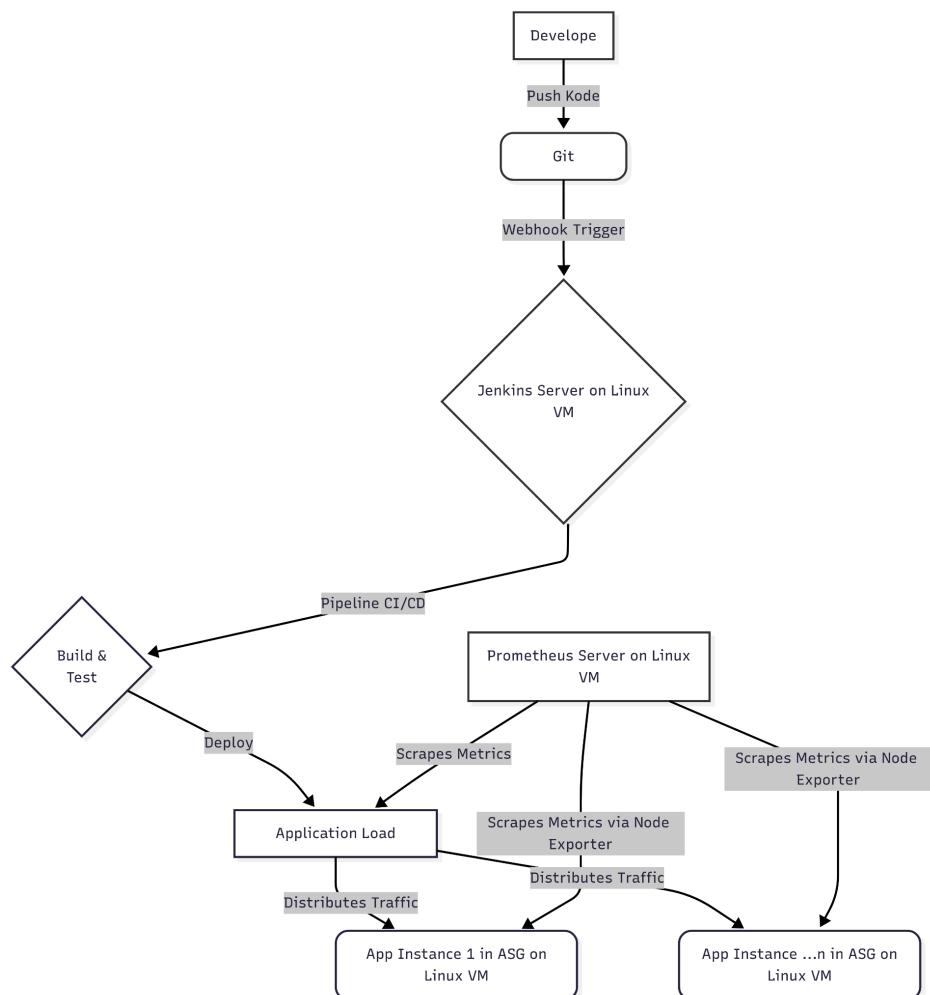
PERANCANGAN SISTEM

2.1 Gambaran Umum Arsitektur

Bagian ini menjelaskan arsitektur sistem yang akan diimplementasikan untuk mencapai tujuan otomatisasi deployment dan monitoring aplikasi web sederhana menggunakan pipeline CI/CD di Virtual Machine. Desain sistem berfokus pada integrasi Git, Jenkins, Virtual Machine (Linux OS).

2.2 Diagram Alur Sistem

Diagram alur sistem (Gambar 1.1) berikut menggambarkan secara visual keseluruhan proses dari pengembangan kode hingga aplikasi berjalan di server, serta bagaimana komponen-komponen berinteraksi satu sama lain.



Gambar 1.1. Diagram Alir Sistem Deployment dan Monitoring

2.3 Penjelasan Komponen Utama

Setiap komponen dalam arsitektur ini memiliki peran krusial dalam memastikan kelancaran *pipeline* CI/CD dan operasional aplikasi. Berikut adalah penjelasan detail dari masing-masing komponen:

- a. **Git:** Berfungsi sebagai sistem kontrol versi terpusat untuk mengelola semua perubahan kode sumber aplikasi. Git juga bertindak sebagai pemicu (*trigger*) awal dalam *pipeline* CI/CD, di mana setiap *push* kode ke *repository* akan memicu proses otomatisasi.
- b. **Jenkins:** Merupakan server otomasi utama yang mengorkestrasikan seluruh *pipeline* CI/CD. Jenkins bertanggung jawab untuk mengambil kode dari Git, menjalankan proses *build*, *test*, dan melakukan *deployment* aplikasi ke lingkungan target di dalam *virtual machine* yang sama.
- c. **Virtual Machine Linux (VMware):** Menyediakan lingkungan komputasi yang terisolasi untuk menjalankan seluruh komponen proyek. Dalam proyek ini, satu *virtual machine* digunakan untuk:
 - i. Hosting server Jenkins.
 - ii. Hosting aplikasi web sederhana yang akan di-deploy.
 - iii. Hosting server Prometheus untuk monitoring.
- d. **Web Server (Nginx):** Berfungsi sebagai server untuk menyajikan (*serve*) file aplikasi web kepada pengguna. Nginx menerima permintaan HTTP dari pengguna dan mengirimkan konten aplikasi yang telah di-deploy oleh Jenkins.
- e. **Prometheus:** Sistem monitoring *open-source* yang digunakan untuk mengumpulkan metrik performa dari aplikasi dan infrastruktur pendukungnya (sumber daya *virtual machine* seperti CPU, memori, dll). Prometheus akan memantau metrik kunci secara berkelanjutan untuk memberikan visibilitas terhadap kesehatan dan performa sistem.

2.4 Alur Kerja CI/CD (Pipeline)

Alur kerja *Continuous Integration/Continuous Deployment* (CI/CD) yang dirancang akan mengikuti langkah-langkah otomatis berikut:

- a. Pengembangan dan Komit Kode

Pengembang melakukan perubahan pada kode sumber aplikasi di lingkungan pengembangan lokal. Setelah perubahan selesai dan diuji

secara lokal, pengembang melakukan *push* kode ke *repository* Git yang telah ditentukan. Langkah ini merupakan titik awal dari seluruh proses otomasi.

b. Pemicuan Build Otomatis

Sebuah webhook yang telah dikonfigurasi pada repository Git akan mendeteksi push kode yang baru. Webhook ini kemudian secara otomatis mengirimkan notifikasi ke server Jenkins, yang selanjutnya akan memicu eksekusi pipeline CI/CD yang telah didefinisikan untuk proyek ini.

c. Proses Pipeline Jenkins:

Jenkins akan menjalankan serangkaian tahapan (stages) yang telah didefinisikan di dalam Jenkinsfile. Tahapan ini meliputi:

- i. **Checkout:** Jenkins akan mengambil kode sumber terbaru dari *repository* Git.
- ii. **Build:** Jenkins akan menjalankan proses *build* aplikasi jika diperlukan (misalnya, kompilasi kode atau resolusi dependensi).
- iii. **Test:** Setelah proses *build* berhasil, Jenkins akan menjalankan serangkaian tes otomatis (misalnya, pengecekan keberadaan file, validasi format) untuk memastikan kualitas dan fungsionalitas kode.
- iv. **Deploy:** Jika semua tes berhasil, Jenkins akan melakukan *deployment* artefak aplikasi ke direktori target pada web server Nginx yang berjalan di dalam **Virtual Machine Linux** yang sama.

d. Akses Aplikasi oleh Pengguna

Setelah aplikasi berhasil di-*deploy* ke direktori web server, aplikasi siap untuk diakses. Pengguna dapat mengakses aplikasi secara langsung melalui alamat IP dari **Virtual Machine** pada *port* yang telah dikonfigurasi untuk web server Nginx.

e. Pemantauan Berkelanjutan:

Prometheus akan secara terus-menerus memantau metrik performa dari aplikasi yang berjalan dan juga infrastruktur Virtual

Machine yang mendasarinya (misalnya, penggunaan CPU, memori, network I/O). Data monitoring ini akan digunakan untuk analisis performa dan identifikasi masalah, serta dapat menjadi masukan untuk keputusan optimalisasi sumber daya Virtual Machine di masa mendatang.

BAB III

IMPLEMENTASI

3.1 Progres Implementasi

3.1.1 Langkah 1: Persiapan Awal

- 1) Menyiapkan VMware Workstation/Player dan buat Virtual Machine Baru di VMware dengan spesifikasi:
 - a) RAM minimal 4GB (8GB lebih baik jika ingin menjalankan Jenkins, Prometheus, dan aplikasi dengan lancar).
 - b) CPU minimal 2 core.
 - c) Disk minimal 30-40GB.
 - d) Konfigurasi jaringan ke NAT atau Bridged. NAT lebih mudah untuk awal, Bridged memungkinkan VM-mu seperti perangkat lain di jaringan lokalmu.
 - 2) Menginstal Ubuntu 24.04 di VM tersebut

3.1.2 Langkah 2: Instalasi Software Pendukung di VM Ubuntu

1. Melakukan update terhadap sistem
sudo apt update
sudo apt upgrade -y

- ## 2. Menginstal Git

```
sudo apt install git -y
```

```

pal-project - VMware Workstation
File Edit View VM Tabs Help || Window X Windows 10 X pal-project X
Library Type here E
[My Computer] [Windows 10] [pal-project]
faiz@faiz-VMware-Virtual-Platform: ~
May 27 (847)

Setting up gnome-shell-extension-desktop-icons-ng (46-really47.0.9~ubuntu1) ...
Setting up gnome-shell-extension-dock@ubuntu1 ...
Processing triggers for fonts-noto-cjk (0.8.2-0ubuntu0.1) ...
update-initramfs: Generating /boot/initrd.img-6.11.0-26-generic
Processing triggers for libc-bin (2.39~Ubuntu14.1) ...
Reading state information...
Done
The following additional packages will be installed:
  git liblber-perl
Suggested packages:
  git-d daemon-run git-demon-svnsync git-doc git-email git-gui gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git liblber-perl
0 upgraded, 2 newly installed, 0 to remove and 1 not upgraded.
Need to get 4,084 kB of additional disk space.
After this operation, 76.4 MB of additional disk space will be used.
Get: http://ld.archive.ubuntu.com/ubuntu/noble/main amd64 liblber0 all 2:17.0.0-1ubuntu7.2 [1,108 kB]
Get: http://ld.archive.ubuntu.com/ubuntu/noble/main amd64 git all64 git amd64 12:43.0~ubuntu7.2 [3,679 kB]
Fetched 4,084 kB in 41s (116 kB/s)
Reading previously unselected package liblber-perl.
(Reading database ... 1,141 files and directories currently installed.)
Preconfiguring packages ...
Unpacking liblber-perl (2:17.0.0-1ubuntu7.2) ...
Selecting previously unselected package git-man.
Preparing to unpack .../git-13432.43.0~ubuntu7.2_all.deb ...
Unpacking git-man (12:43.0~ubuntu7.2) ...
Processing triggers for man-db (2.12.0~build1) ...
git version 2.43.0
git (Ubuntu 2.43.0-17.0.1546-01~ubuntul-24.84) ...
faiz@faiz-VMware-Virtual-Platform: $ git --version
git version 2.43.0
faiz@faiz-VMware-Virtual-Platform: $

```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

3. Instal Java (JDK dibutuhkan oleh Jenkins)
- ```
sudo apt install openjdk-17-jdk -y
```

```

pal-project - VMware Workstation
File Edit View VM Tabs Help || Window X Windows 10 X pal-project X
Library Type here E
[My Computer] [Windows 10] [pal-project]
faiz@faiz-VMware-Virtual-Platform: ~
May 27 (850)

Adding debian/TWC Root_Certification_Authority.pem
Adding debian/UCA_Extended_Validation_Root.pem
Adding debian/UCA_Global_C2_Root.pem
Adding debian/USERSign_ECC_Certification_Authority.pem
Adding debian/USERTrust_RSA_Certification_Authority.pem
Adding debian/virus_ECC_Root_CA.pem
Adding debian/virus_Root_CA.pem
Adding debian/XMang_Global_CA_Root.pem
done.
Setting up openjdk-17-jre-headless (17.0.1546-01~ubuntul-24.84) ...
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jar to provide /usr/bin/jar (.jar) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jarsigner to provide /usr/bin/jarsigner (.jarsigner) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/javac to provide /usr/bin/javac (.javac) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/javap to provide /usr/bin/javap (.javap) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/javaws to provide /usr/bin/javaws (.javaws) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/javadoc to provide /usr/bin/javadoc (.javadoc) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jdeps to provide /usr/bin/jdeps (.jdeps) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jfr to provide /usr/bin/jfr (.jfr) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jhat to provide /usr/bin/jhat (.jhat) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jinfo to provide /usr/bin/jinfo (.jinfo) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jlink to provide /usr/bin/jlink (.jlink) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jmap to provide /usr/bin/jmap (.jmap) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jmeminfo to provide /usr/bin/jmeminfo (.jmeminfo) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jps to provide /usr/bin/jps (.jps) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jpscan to provide /usr/bin/jpscan (.jpscan) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jpsvc to provide /usr/bin/jpsvc (.jpsvc) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jstat to provide /usr/bin/jstat (.jstat) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jstatd to provide /usr/bin/jstatd (.jstatd) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jtmc to provide /usr/bin/jtmc (.jtmc) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jtmcdb to provide /usr/bin/jtmcdb (.jtmcdb) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jtmcstack to provide /usr/bin/jtmcstack (.jtmcstack) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jstat to provide /usr/bin/jstat (.jstat) in auto mode
Setting up openjdk-17-jdk (17.0.1546-01~ubuntul-24.84) ...
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jhsdb to provide /usr/bin/jhsdb (.jhsdb) in auto mode
Setting up openjdk-17-jdk-headless (17.0.1546-01~ubuntul-24.84) ...
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jconsole to provide /usr/bin/jconsole (.jconsole) in auto mode
faiz@faiz-VMware-Virtual-Platform: $

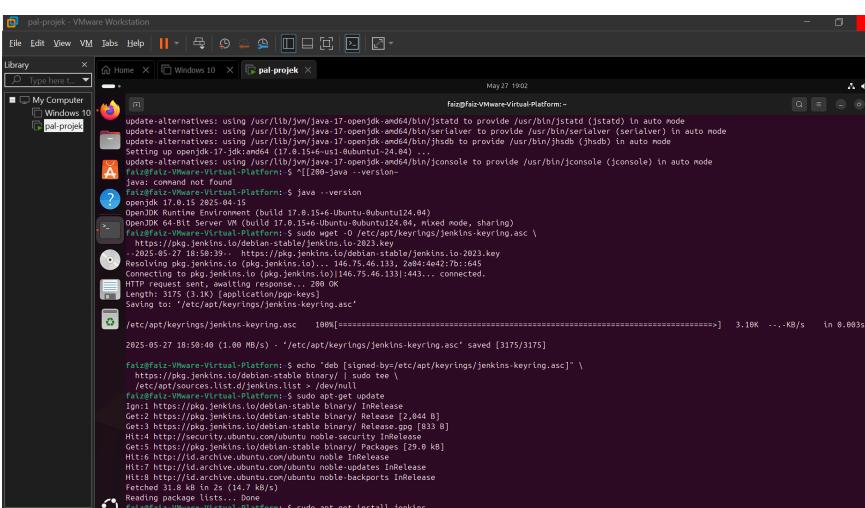
```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

4. Instal Jenkins dan akan menggunakan Jenkins dari repository resminya.

# Tambahkan Jenkins repository

```
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
```



```
pal-project - VMware Workstation
File Edit View VM Tabs Help ○
Library x Home x Windows 10 x pal-project x May 27 1902
Type here t
File Home x Windows 10 x pal-project x
May 27 1902
feizi@feizi-VMware-Virtual-Platform: ~
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/jstatd to provide /usr/bin/statd (jstatd) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/jstatlserver to provide /usr/bin/jstatlserver (jstatlserver) in auto mode
Setting up openjdk-17-jdk:amd64 (17.0.15+8-0ubuntu17.4.0)
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/jhsdb to provide /usr/bin/jhsdb (jhsdb) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/jconsole to provide /usr/bin/jconsole (jconsole) in auto mode
feizi@feizi-VMware-Virtual-Platform: $ lsof |grep java --version
java
feizi@feizi-VMware-Virtual-Platform: $ java --version
java(TM) SE Runtime Environment (build 17.0.15+8-0ubuntu17.4.0)
OpenJDK 64-Bit Server VM (build 17.0.15+8-0ubuntu17.4.0, mixed mode, sharing)
feizi@feizi-VMware-Virtual-Platform: $ sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
--2025-05-27 18:58:40-- https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
Resolving pkg.jenkins.io (pkg.jenkins.io)... 146.75.46.133, 2a0e:4e42:7b::1465
Connecting to pkg.jenkins.io (pkg.jenkins.io)|146.75.46.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3175 (3.1K) [application/pgp-keys]
Saving to: '/etc/apt/keyrings/jenkins-keyring.asc'

/etc/apt/keyrings/jenkins-keyring.asc 100%[=====] 3.10K --.KB/s in 0.003s

2025-05-27 18:58:40 (1.00 MB/s) - '/etc/apt/keyrings/jenkins-keyring.asc' saved [3175/3175]

feizi@feizi-VMware-Virtual-Platform: $ echo -e "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian-stable binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list >/dev/null
feizi@feizi-VMware-Virtual-Platform: $ sudo apt-get update
Ign:1 http://security.ubuntu.com/ubuntu focal-security InRelease
Get:1 https://pkg.jenkins.io/debian-stable binary/ Release [844 B]
Get:2 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Hit:3 https://security.ubuntu.com/ubuntu focal/noblegui InRelease
Get:4 https://security.ubuntu.com/ubuntu focal-updates InRelease [29.8 kB]
Hit:5 https://id.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:6 https://id.archive.ubuntu.com/ubuntu focal-backports InRelease
Fetched 31 packages (14.6 kB/s)
Reading package lists... Done
feizi@feizi-VMware-Virtual-Platform: $ sudo apt-get install jenkins
Reading package lists... Done
To direct input to this VM, move the mouse pointer inside or press CMF-G.
```

```
Update package list dan instal Jenkins
sudo apt-get update
sudo apt-get install jenkins -y
```

```
Start dan enable Jenkins service
sudo systemctl enable jenkins
sudo systemctl start jenkins
sudo systemctl status jenkins
```

```

pal-project - VMware Workstation
File Edit View VM Tabs Help || Window | Home | Windows 10 | pal-project |
Library | My Computer | Windows 10 | pal-project |

Setting up net-tools (2.10-0~Ubuntu04) ...
Setting up jenkins (2.504.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /usr/lib/systemd/system/jenkins.service.
Process triggered For name (2.12.0-4build2) ...
FatzFatz@FatzFatz-Virtual-Platform: ~ sudo apt-get install jenkins -y
Reading package lists... Done
Building dependency tree... Done
jenkins is already the newest version (2.504.1).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/system-sysv-install.
Executing: /usr/lib/systemd/system-sysv-install enable jenkins
FatzFatz@FatzFatz-Virtual-Platform: ~ sudo systemctl start jenkins
FatzFatz@FatzFatz-Virtual-Platform: ~ systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
 Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
 Active: active (running) since Tue 2025-05-27 18:56:43 WIB; 1min 19s ago
 Main PID: 22572 (/usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080)
 Tasks: 44 (limit: 4551)
 Memory: 599.3M (peak: 607.2M)
 CPU: 42.398s
 CGroup: /system.slice/jenkins.service
 └─22572 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

May 27 18:56:26 FatzFatz-Virtual-Platform jenkins[22572]: 6b57cd6764ceabb18408809fbef
May 27 18:56:26 FatzFatz-Virtual-Platform jenkins[22572]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
May 27 18:56:26 FatzFatz-Virtual-Platform jenkins[22572]: ****
May 27 18:56:26 FatzFatz-Virtual-Platform jenkins[22572]: ****
May 27 18:56:26 FatzFatz-Virtual-Platform jenkins[22572]: ****
May 27 18:56:43 FatzFatz-Virtual-Platform jenkins[22572]: 2025-05-27 11:56:43.425+0000 [id:30] INFO jenkins.InitReactorRunner$1#onAttained: Component hudson.lifecycle.Lifecycle$OnReady: Jenkins
May 27 18:56:43 FatzFatz-Virtual-Platform jenkins[22572]: 2025-05-27 11:56:43.459+0000 [id:3] INFO hudson.util.Retriger$#start: Performed the action
May 27 18:56:44 FatzFatz-Virtual-Platform jenkins[22572]: 2025-05-27 11:56:46.695+0000 [id:49] INFO h.n.DownloadService$DownloadableLoad: Obtained
May 27 18:56:44 FatzFatz-Virtual-Platform jenkins[22572]: 2025-05-27 11:56:46.700+0000 [id:49] INFO hudson.util.Retriger$#start: Performed the action
FatzFatz@FatzFatz-Virtual-Platform: ~ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
6b57cd6764ceabb18408809fbef
FatzFatz@FatzFatz-Virtual-Platform: ~

```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

Kemudian, proses konfigurasi awal Jenkins dimulai dengan mengakses antarmuka webnya dari mesin host di alamat <http://192.168.47.137:8080>. Untuk melanjutkan, halaman "Unlock Jenkins" memerlukan kata sandi administrator awal yang didapatkan dari virtual machine dengan perintah:

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

```

pal-project - VMware Workstation
File Edit View VM Tabs Help || Window | Home | Windows 10 | pal-project |
Library | My Computer | Windows 10 | pal-project |

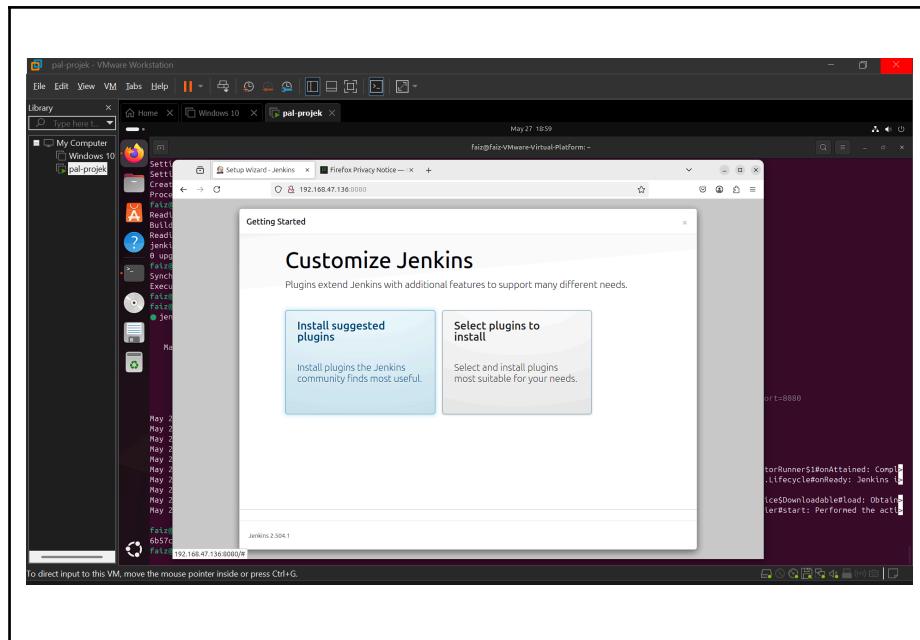
Setting up net-tools (2.10-0~Ubuntu04) ...
Setting up jenkins (2.504.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /usr/lib/systemd/system/jenkins.service.
Process triggered For name (2.12.0-4build2) ...
FatzFatz@FatzFatz-Virtual-Platform: ~ sudo apt-get install jenkins -y
Reading package lists... Done
Building dependency tree... Done
jenkins is already the newest version (2.504.1).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/system-sysv-install.
Executing: /usr/lib/systemd/system-sysv-install enable jenkins
FatzFatz@FatzFatz-Virtual-Platform: ~ sudo systemctl start jenkins
FatzFatz@FatzFatz-Virtual-Platform: ~ systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
 Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
 Active: active (running) since Tue 2025-05-27 18:56:43 WIB; 1min 19s ago
 Main PID: 22572 (/usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080)
 Tasks: 44 (limit: 4551)
 Memory: 599.3M (peak: 607.2M)
 CPU: 42.398s
 CGroup: /system.slice/jenkins.service
 └─22572 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

May 27 18:56:26 FatzFatz-Virtual-Platform jenkins[22572]: 6b57cd6764ceabb18408809fbef
May 27 18:56:26 FatzFatz-Virtual-Platform jenkins[22572]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
May 27 18:56:26 FatzFatz-Virtual-Platform jenkins[22572]: ****
May 27 18:56:26 FatzFatz-Virtual-Platform jenkins[22572]: ****
May 27 18:56:26 FatzFatz-Virtual-Platform jenkins[22572]: ****
May 27 18:56:43 FatzFatz-Virtual-Platform jenkins[22572]: 2025-05-27 11:56:43.425+0000 [id:30] INFO jenkins.InitReactorRunner$1#onAttained: Component hudson.lifecycle.Lifecycle$OnReady: Jenkins
May 27 18:56:43 FatzFatz-Virtual-Platform jenkins[22572]: 2025-05-27 11:56:43.459+0000 [id:3] INFO hudson.util.Retriger$#start: Performed the action
May 27 18:56:44 FatzFatz-Virtual-Platform jenkins[22572]: 2025-05-27 11:56:46.695+0000 [id:49] INFO h.n.DownloadService$DownloadableLoad: Obtained
May 27 18:56:44 FatzFatz-Virtual-Platform jenkins[22572]: 2025-05-27 11:56:46.700+0000 [id:49] INFO hudson.util.Retriger$#start: Performed the action
FatzFatz@FatzFatz-Virtual-Platform: ~ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
6b57cd6764ceabb18408809fbef
FatzFatz@FatzFatz-Virtual-Platform: ~

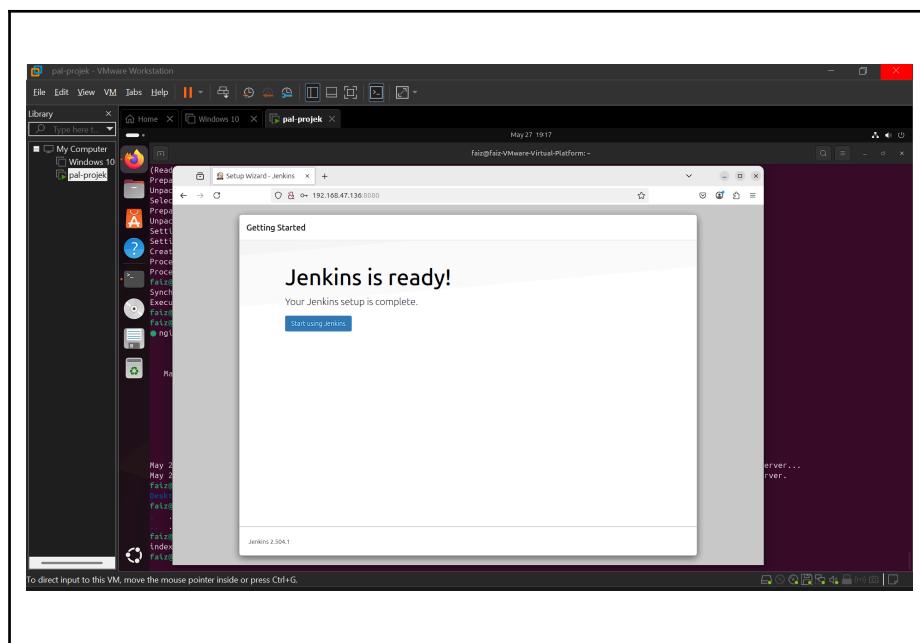
```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

Setelah berhasil membuka kunci, pilih opsi "Install Suggested Plugins"



Kemudian pilih “Create First Admin User” dan Jenkins siap dipakai.

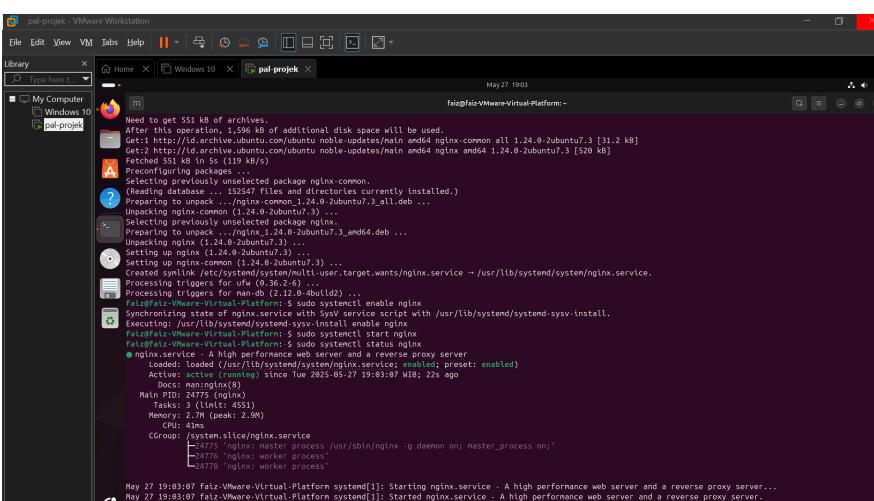


## 5. Instal Web Server (Nginx) di VM Ubuntu

Nginx akan berfungsi sebagai server untuk men-deploy aplikasi web sederhana. Instalasi dengan perintah:

```
sudo apt install nginx -y
```

```
Start dan enable Nginx
sudo systemctl enable nginx
sudo systemctl start nginx
sudo systemctl status nginx
```

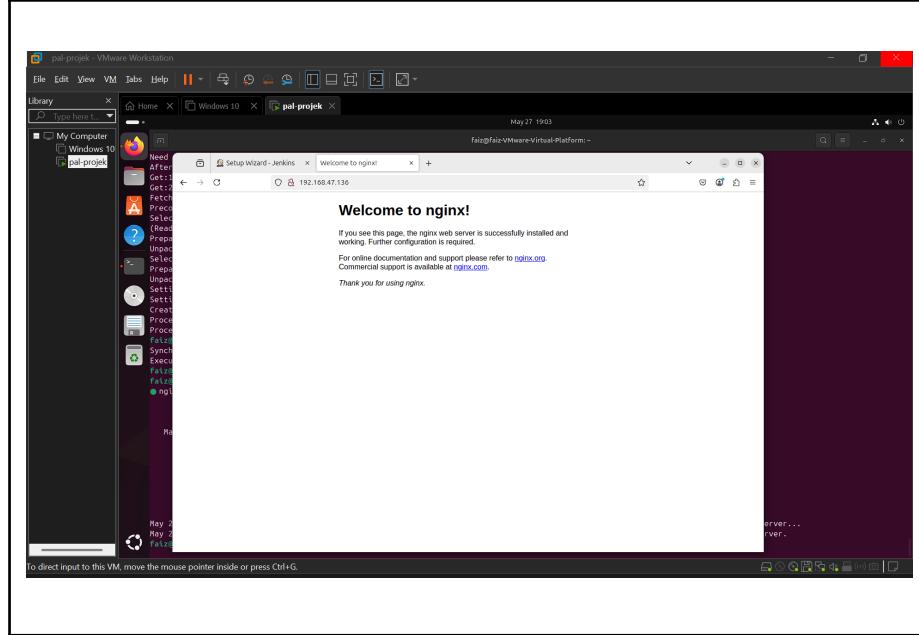


```
pal-projek - VMware Workstation
File Edit View VM Tabs Help || x
Library X Home X Windows 10 X pal-projek X
Type here to search
My Computer X Windows 10 X pal-projek X
faiz@faiz-Virtual-Platform: ~
Need to get 551 kB of archives.
After this operation, 1,596 kB of additional disk space will be used.
Get: https://de.archive.ubuntu.com/ubuntu focal-updates/main amd64 nginx-common all 1:24.0-2ubuntu7.3 [31.2 kB]
Get: https://de.archive.ubuntu.com/ubuntu focal-updates/main amd64 nginx amd64 1:24.0-2ubuntu7.3 [520 kB]
Fetched 551 kB in 5s (110 kB/s)
Preconfiguring packages...
Selecting previously selected package nginx-common.
Reading package lists... Done
Preparing to unpack .../nginx-common_1.24.0-2ubuntu7.3_all.deb ...
Unpacking nginx-common (1:24.0-2ubuntu7.3) ...
Selecting previously unselected package nginx.
Preparing to unpack .../nginx_1.24.0-2ubuntu7.3_amd64.deb ...
Unpacking nginx (1:24.0-2ubuntu7.3) ...
Setting up nginx (1:24.0-2ubuntu7.3) ...
Setting up nginx (1:24.0-2ubuntu7.3) ...
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
Processing triggers for ufw (0.36-2.6) ...
Processing triggers for man-db (2.12.1-1) ...
Processing triggers for systemd-sysv-install ...
Synchronizing state of nginx.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nginx
faiz@faiz-Virtual-Platform: ~ sudo systemctl start nginx
faiz@faiz-Virtual-Platform: ~ systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
 Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
 Active: active (running) since Tue 2025-05-27 19:08:07 WIB; 22s ago
 Docs: man:nginx(8)
 Main PID: 24775 (nginx)
 Tasks: 3 (limit: 4551)
 Memory: 2.0M
 CPU: 41ms
 CGroup: /system.slice/nginx.service
 └─24775 nginx master process
 ├─24778 nginx worker process
 ├─24779 nginx worker process
 ├─24780 nginx worker process

May 27 19:08:07 faiz-Virtual-Platform systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
May 27 19:08:07 faiz-Virtual-Platform systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.

faiz@faiz-Virtual-Platform: ~
```

Setelah itu, mencoba mengakses host <http://192.168.47.137>. pada browser, dan hasilnya seperti pada gambar di bawah ini.



## 6. Instal Prometheus & Node Exporter

Awal langkah dengan membuat direktori untuk konfigurasi Prometheus:

```
mkdir ~/prometheus_data
cd ~/prometheus_data
```

```
faiz@faiz-VMware-Virtual-Platform:~$ mkdir ~/prometheus_data
faiz@faiz-VMware-Virtual-Platform:~$ cd ~/prometheus_data
faiz@faiz-VMware-Virtual-Platform:~/prometheus_data$ nano prometheus.yml
faiz@faiz-VMware-Virtual-Platform:~/prometheus_data$
```

Kemudian membuat file konfigurasi prometheus.yml:

```
nano prometheus.yml
```

Dan Isi dengan konfigurasi dengan berikut:

```
global:
 scrape_interval: 15s
```

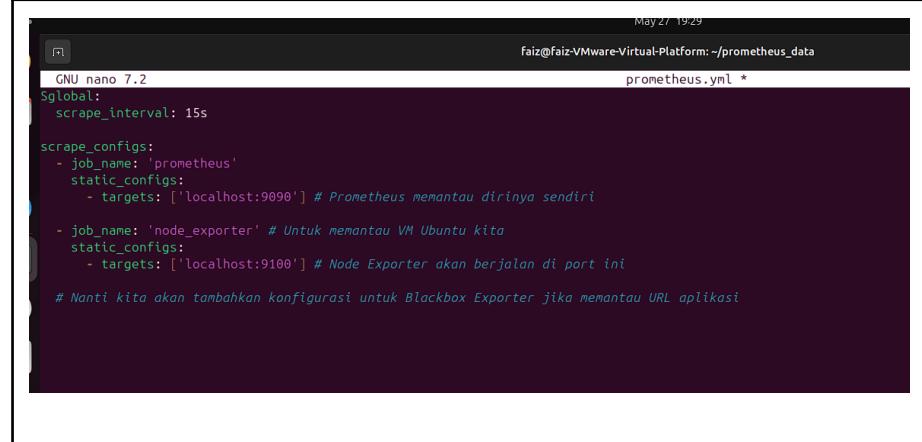
```
scrape_configs:
```

- job\_name: 'prometheus'

```
static_configs:
```

- targets: ['localhost:9090'] # Prometheus memantau dirinya sendiri

```
- job_name: 'node_exporter' # Untuk memantau VM Ubuntu kita
 static_configs:
 - targets: ['localhost:9100'] # Node Exporter akan berjalan di port
```



The screenshot shows a terminal window titled 'GNU nano 7.2' with the file 'prometheus.yml' open. The file contains the following YAML configuration:

```
May 27 19:29
faiz@faiz-VMware-Virtual-Platform: ~/prometheus_data
prometheus.yml *

global:
 scrape_interval: 15s

scrape_configs:
 - job_name: 'prometheus'
 static_configs:
 - targets: ['localhost:9090'] # Prometheus memantau dirinya sendiri

 - job_name: 'node_exporter' # Untuk memantau VM Ubuntu kita
 static_configs:
 - targets: ['localhost:9100'] # Node Exporter akan berjalan di port ini

Nanti kita akan tambahkan konfigurasi untuk Blackbox Exporter jika memantau URL aplikasi
```

Selanjutnya menjalankan Node Exporter menggunakan Docker:

```
docker run -d \
--name=node_exporter \
--pid="host" \
--restart=unless-stopped \
-p 9100:9100 \
prom/node-exporter:latest
```

```

faiz@faiz-VMware-Virtual-Platform:~/prometheus_data$ docker run -d \
--name=node_exporter \
--pid="host" \
--restart=unless-stopped \
-p 9100:9100 \
prom/node-exporter:latest
Unable to find image 'prom/node-exporter:latest' locally
latest: Pulling from prom/node-exporter
9fa9226be034: Pull complete
1617e25568b2: Pull complete
c6e37428e3b3: Pull complete
Digest: sha256:d00a542e409ee618a4edc67da14dd48c5da66726bbdb5537ab2af9c1dfc442c8a
Status: Downloaded newer image for prom/node-exporter:latest
3fe448627946597efc6b1910fdb646eb1a3a74f6ea92830283abe7585b511a0
faiz@faiz-VMware-Virtual-Platform:~/prometheus_data$ docker run -d \
--name=prometheus \
-p 9090:9090 \
--restart=unless-stopped \
-v ~/prometheus_data/prometheus.yml:/etc/prometheus/prometheus.yml \
prom/prometheus:latest
Unable to find image 'prom/prometheus:latest' locally
latest: Pulling from prom/prometheus
9fa9226be034: Already exists
1617e25568b2: Already exists
01e225ff2cae: Pull complete
6e1c59ce43d3: Pull complete
303f67c648a5: Pull complete
d2597eeb55f4: Pull complete
744c8ca72725: Pull complete
e52136ad7a1b: Pull complete
44750a25eb21: Pull complete
b6b811691043: Pull complete
Digest: sha256:78ed1f9050eb9eaf766af6e580230b1c4965728650e332cd1ee918c0c4699775
Status: Downloaded newer image for prom/prometheus:latest

```

Selanjutnya menjalankan Prometheus menggunakan Docker:

```

docker run -d \
--name=prometheus \
-p 9090:9090 \
--restart=unless-stopped \
-v
~/prometheus_data/prometheus.yml:/etc/prometheus/prometheus.yml \
prom/prometheus:latest

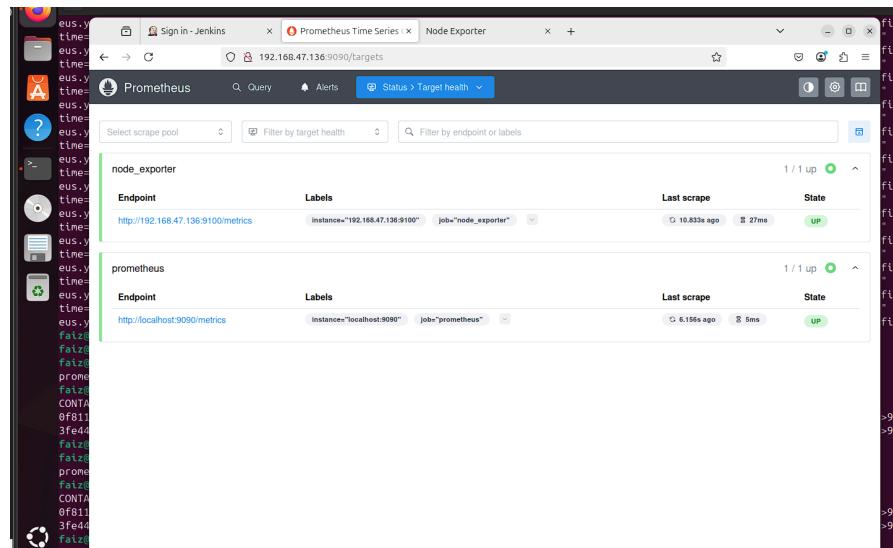
```

```

faiz@faiz-VMware-Virtual-Platform:~/prometheus_data$ docker run -d \
--name=node_exporter \
--pid="host" \
--restart=unless-stopped \
-p 9100:9100 \
prom/node-exporter:latest
Unable to find image 'prom/node-exporter:latest' locally
latest: Pulling from prom/node-exporter
9fa9226be034: Pull complete
1617e25568b2: Pull complete
c6e37428e3b3: Pull complete
Digest: sha256:d00a542e409ee618a4edc67da14dd48c5da66726bbdb5537ab2af9c1dfc442c8a
faiz@faiz-VMware-Virtual-Platform:~/prometheus_data$ docker run -d \
--name=prometheus \
-p 9090:9090 \
--restart=unless-stopped \
-v ~/prometheus_data/prometheus.yml:/etc/prometheus/prometheus.yml \
prom/prometheus:latest
Unable to find image 'prom/prometheus:latest' locally
latest: Pulling from prom/prometheus
9fa9226be034: Already exists
1617e25568b2: Already exists
01e225ff2cae: Pull complete
6e1c59ce43d3: Pull complete
303f67c648a5: Pull complete
d2597eeb55f4: Pull complete
744c8ca72725: Pull complete
e52136ad7a1b: Pull complete
44750a25eb21: Pull complete
b6b811691043: Pull complete
Digest: sha256:78ed1f9050eb9eaf766af6e580230b1c4965728650e332cd1ee918c0c4699775
Status: Downloaded newer image for prom/prometheus:latest

```

Dan yang terakhir mencoba untuk mengakses host <http://192.168.47.137:9090>. Setelah itu mengecek ketersediaan layanan Node Exporter dengan mengakses “Cek” dan klik “Target”.



### 3.1.3 Langkah 3: Setup Aplikasi Web Sederhana

Aplikasi yang digunakan sebagai studi kasus dalam proyek ini adalah sebuah aplikasi web statis yang terdiri dari HTML, CSS, dan JavaScript. Kode sumber untuk aplikasi ini tersedia secara publik dalam sebuah repository Git di URL <https://github.com/faizfznn/automasi-projek-web>. Sebagai langkah awal penyiapan, repository tersebut di-kloning ke dalam direktori home pada virtual machine dengan menjalankan perintah git clone <https://github.com/faizfznn/automasi-projek-web.git>. Proses ini menghasilkan sebuah direktori lokal baru bernama automasi-projek-web yang berisi seluruh kode sumber aplikasi yang siap untuk proses deployment.

```
faiz@faiz-VMware-Virtual-Platform:~/Documents$ cd
faiz@faiz-VMware-Virtual-Platform:~$ git clone https://github.com/faizfznn/automasi-projek-web.git
Cloning into 'automasi-projek-web'...
remote: Enumerating objects: 28, done.
remote: Counting objects: 100% (28/28), done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 28 (delta 0), reused 28 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (28/28), 4.98 MiB | 234.00 KiB/s, done.
faiz@faiz-VMware-Virtual-Platform:~$
```

### 3.1.4 Langkah 4: Konfigurasi Jenkins untuk CI/CD

#### 1. Konfigurasi Kredensial Jenkins dan GitHub

Langkah pertama dalam mengkonfigurasi pipeline adalah memastikan Jenkins dapat berinteraksi dengan repository Git secara aman. Untuk tujuan ini, metode autentikasi berbasis kunci SSH digunakan. Sepasang kunci SSH (publik dan privat) dibuat di dalam virtual machine. Kunci publik kemudian didaftarkan pada repository proyek di GitHub sebagai sebuah “Deploy Key”. Konfigurasi ini memberikan izin baca (read-only access) kepada Jenkins, yang cukup untuk mengambil kode sumber proyek. Selanjutnya, kunci privat yang sesuai disimpan di dalam Jenkins sebagai sebuah “Credential” dengan tipe “SSH Username with private key”. Kredensial ini nantinya akan digunakan oleh Jenkins Job untuk melakukan autentikasi secara aman saat terhubung ke repository Git.

```
faiz@faiz-VMware-Virtual-Platform:~$ ssh-keygen -t rsa -b 4096 -C "jenkins@faiz"
```

```
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/home/faiz/.ssh/id_rsa):
```

```
Enter passphrase (empty for no passphrase):
```

```
Enter same passphrase again:
```

```
Your identification has been saved in /home/faiz/.ssh/id_rsa
```

```
Your public key has been saved in /home/faiz/.ssh/id_rsa.pub
```

```
The key's fingerprint is:
```

```
SHA256:4nQJW48abvnsz+KDyJhuaX/xbvItldo+BNWD02U0f1s jenkins@faiz
```

```
The key's randomart image is:
```

```
+---[RSA 4096]---
```

```
| o |
```

```
| + = |
```

```
| . . B . |
```

```
| +.+o o . E|
```

```
| = S..o . o |
```

```
| +.* + . |
```

```
| = .*+ = |
```

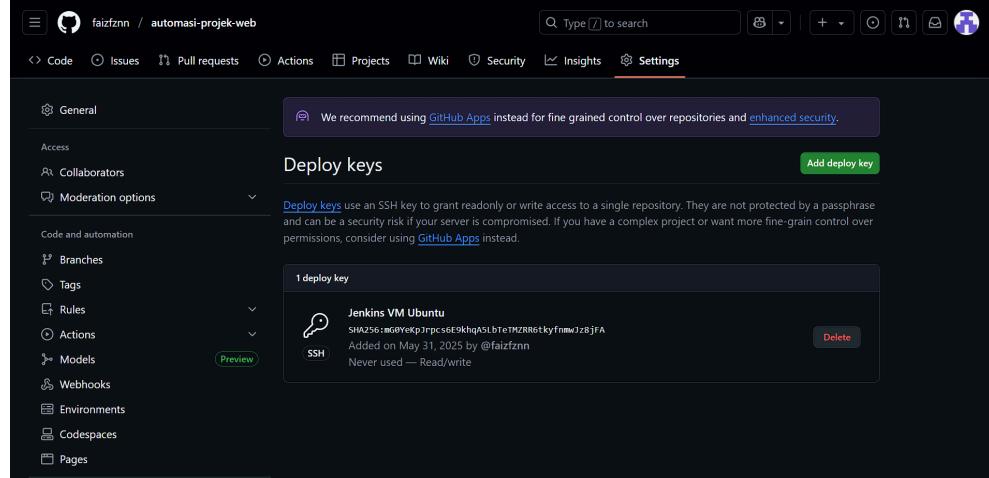
```
| * o..oBoo |
```

```
| +.... BB*=. |
```

```
+---[SHA256]---
```

```
+---[SHA256]---
```

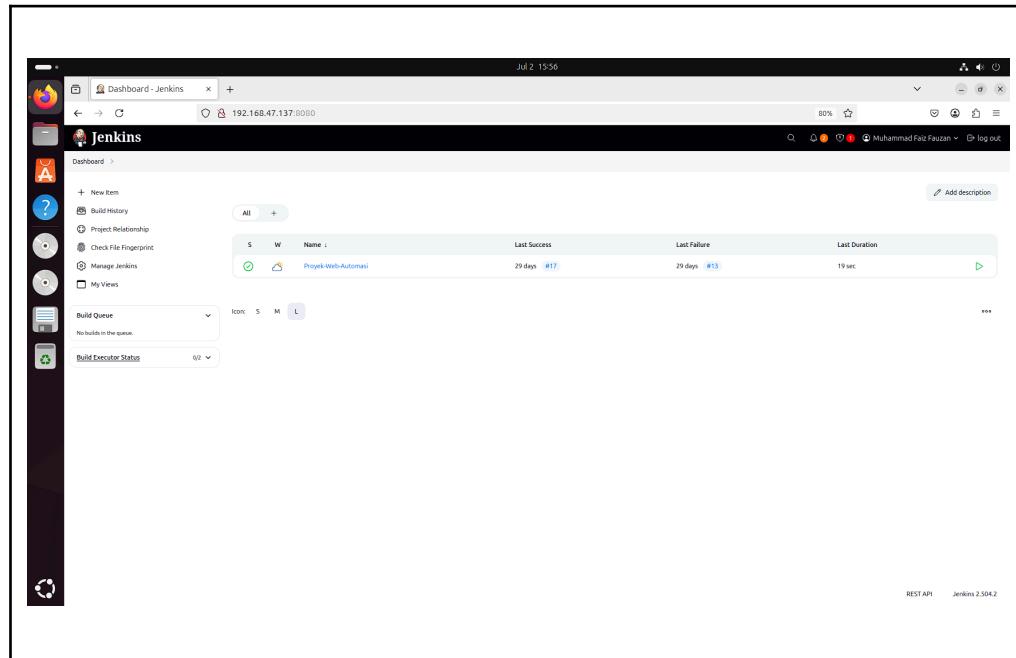
```
faiz@faiz-VMware-Virtual-Platform:~$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAQABAAAQADQp4LX0DQg0nUy31Guhc8phmv+d+DP3tkY8x07n3F4nGzeFeoMXNoYf6gUc1ad/c1qY+cGskznSDeZt0oOsw7KwC1eXEdfJ/LuMtyTtudTCXeLcQ8El18JU1
QcB8Sq+s62qcyFO1sR51h+J2k6d8KA+qjByKKL5888kv17TEyFn0odwvjNzdqzBpW0weI7j13j1rY2D0dRTXhP9ycs5DPVfkE3chU9ALM3v5p5N8ADjsB4H4F9nITqx5w67bz10bdzawmN/F11981z1z9E
c19Blr-Gjyg1fYUW86gYLFAK+Brd8XJScf92j+NDFWPkyBL8DUtxWA2iRcsqFK23fwh08j1x7LWFAlzE9vw18Ptvt7XQe8Pw7kpzsJaV00EsrVDV-N6c0lHwX10XuUMyxxS+PRWIh-0RX564hW62p-/RN6e
sJPLNv18LGEPP9d7n1QdM3Ca5tTWmcnVQWTA7ONzcBhTKRd0ZuaeZ1DC/0j/jgt8bt/10/WKGzqErxpLNhx5zfqTYNKwB3gHzCm+crkLYCd+0sh19uc9N8eqbJhZunksB3gN28bdtWkKsNTBBtKexCUH
ytC2SG3+saKn7xez8g5jACDBCSNxrr+nV12kr0syod8u8eqKoc/NsEB00t73qCYQ== jenkins@faiz
```



## 2. Pembuatan Jenkins Pipeline Job

Setelah kredensial berhasil disiapkan, sebuah Pipeline Job baru dibuat pada antarmuka web Jenkins. Job ini dikonfigurasi untuk menggunakan metode "Pipeline script from SCM" (Source Code Management), yang menginstruksikan Jenkins untuk mengambil definisi dan alur kerja pipeline langsung dari sebuah file yang ada di dalam repository proyek. Pada konfigurasi SCM, URL repository Git dimasukkan dengan menggunakan format SSH (`git@github.com:faizfznn/automasi-projek-web.git`), dan kredensial SSH yang telah dibuat sebelumnya dipilih untuk proses autentikasi.

Selain itu, pemicu (trigger) "GitHub hook trigger for GITScm polling" diaktifkan. Fitur ini memungkinkan pipeline untuk berjalan secara otomatis setiap kali ada perubahan kode baru yang di-push ke branch utama repository, sehingga menciptakan alur kerja CI/CD yang sesungguhnya.



### 3. Pembuatan Jenkinsfile di Repository GitHub

Logika dan alur kerja dari pipeline CI/CD didefinisikan dalam sebuah file bernama Jenkinsfile yang ditempatkan di direktori root pada repository Git.

```
pipeline {
 agent any // Menjalankan di agent Jenkins manapun yang tersedia

 environment {
 // Direktori target di web server (Nginx)
 DEPLOY_PATH = '/var/www/html/automasi-projek-web'
 }

 stages {
```

```

stage('1. Checkout Code') {
 steps {
 echo 'Mencoba mengambil kode dari GitHub...'
 checkout scm
 echo 'Kode berhasil diambil.'
 }
}

stage('2. Clean Old Deployment (Opsional)') {
 steps {
 echo "Membersihkan direktori deployment lama di ${env.DEPLOY_PATH} jika ada..."
 // Hati-hati dengan rm -rf! Pastikan path benar.
 // Jenkins user mungkin butuh sudo. Ini akan kita atasi nanti.
 // Untuk sekarang, kita akan deploy ke subfolder di /var/www/html
 // atau kita bisa atur permission /var/www/html agar jenkins bisa menulis.
 // Alternatif: copy dengan rsync dan opsi --delete
 sh "sudo rm -rf ${env.DEPLOY_PATH}/*" // Perlu konfigurasi sudo untuk jenkins
 }
}

```

stage('3. Deploy to Web Server (Nginx)') {

```

 steps {
 echo "Menyalin file aplikasi ke ${env.DEPLOY_PATH}..."
 // Pastikan direktori target ada dan Jenkins punya izin tulis
 }
}

```

```

// Kita buat direktori jika belum ada
sh "sudo mkdir -p ${env.DEPLOY_PATH}"

sh "sudo cp -R * ${env.DEPLOY_PATH}" // Menyalin semua
dari workspace ke target

echo "Aplikasi berhasil di-deploy ke Nginx."

}

}

stage('4. Test Ketersediaan Aplikasi') {

steps {

echo "Menguji apakah aplikasi bisa diakses.."

// Ini tes yang sangat sederhana, hanya cek apakah URL
mengembalikan status 200 OK

// Kita asumsikan aplikasi akan diakses via
<IP_VM>/automasi-projek-web/

// Jika kamu deploy langsung ke root /var/www/html, sesuaikan
URL

sh "curl -sSf http://localhost/automasi-projek-web/index.html ||"
exit 1"

echo "Tes ketersediaan aplikasi berhasil."

}

}

post {

always {

echo 'Pipeline selesai.'

// Clean up workspace Jenkins

```

```

cleanWs()

}

success {

 echo 'Pipeline berhasil!'

}

failure {

 echo 'Pipeline gagal!'

 // Di sini kamu bisa menambahkan notifikasi (misal: email)

}

}

```

Script pipeline ini ditulis menggunakan sintaks Groovy dan terdiri dari beberapa tahapan (stages) utama yang dieksekusi secara berurutan.

- a. Checkout Code yakni tahap pertama yang bertugas mengambil kode sumber terbaru dari repository GitHub.
- b. Pada tahap "Deploy to Web Server", Jenkins menyalin seluruh file aplikasi dari workspace internalnya ke direktori target pada web server Nginx, yaitu /var/www/html/automasi-projek-web. Perintah seperti mkdir -p dan cp -R digunakan untuk memastikan direktori tujuan ada dan semua file aplikasi tersalin dengan benar.
- c. Terakhir tahap "Test Ketersediaan Aplikasi" sebagai langkah verifikasi akhir, sebuah pengujian sederhana dijalankan menggunakan perintah curl untuk memeriksa apakah halaman utama aplikasi (index.html) dapat diakses melalui URL lokal (<http://localhost/automasi-projek-web/index.html>) dan mengembalikan status sukses.

#### 4. Konfigurasi Nginx

Sebagai langkah akhir, konfigurasi server web Nginx dipastikan dapat menyajikan file aplikasi dari direktori tujuan deployment. Dengan Jenkins yang menempatkan file aplikasi di subdirektori /var/www/html/automasi-projek-web/, aplikasi tersebut dapat langsung diakses melalui URL [http://<IP\\_VM>/automasi-projek-web/](http://<IP_VM>/automasi-projek-web/) tanpa memerlukan perubahan konfigurasi Nginx yang signifikan. Hal ini melengkapi alur kerja deployment otomatis dari awal hingga akhir.

### 3.1.5 Langkah 5: Uji Coba Alur CI/CD

1. Melakukan build manual pertama kali

Dengan membuka dashboard Jenkins, memilih job Proyek-Web-Automasi, kemudian menekan tombol Build Now. Proses ini memicu pipeline untuk menarik kode terbaru dari repository, membersihkan direktori deployment lama, lalu menyalin file ke direktori yang digunakan oleh Nginx. Output build dipantau melalui Console Output untuk memastikan tidak ada error yang muncul, terutama terkait perizinan direktori atau kesalahan jalur.

2. Memverifikasi deployment awal ke web server

Setelah build manual berhasil, dilakukan pengecekan hasil deployment dengan mengakses URL <http://192.168.47.137/automasi-projek-web/index.html> melalui browser. Jika konfigurasi Nginx serta pipeline Jenkins sudah benar, maka halaman web yang berasal dari repository akan tampil, menandakan proses deployment berhasil dilakukan.

3. Menguji perubahan kode untuk CI/CD

Untuk menguji jalur CI/CD berjalan otomatis saat ada perubahan kode, dilakukan modifikasi pada salah satu file di repository automasi-projek-web. Misalnya, mengedit teks pada file index.html menggunakan editor lokal. Setelah itu, perubahan di-commit dan di-push ke branch utama repository GitHub dengan perintah `git add .`, `git commit -m "Test CI/CD: Updated index.html"`, dan `git push origin main`.

4. Memantau build otomatis pada Jenkins

Jika webhook GitHub sudah terkonfigurasi dengan benar (atau

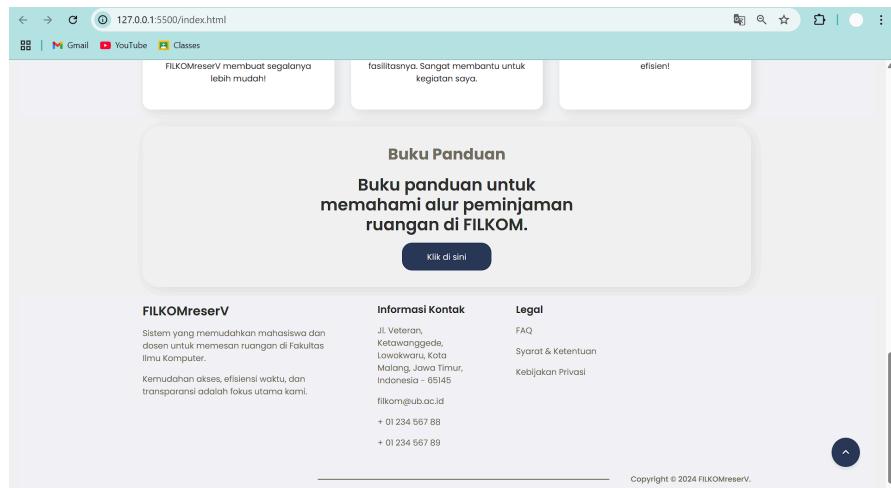
menggunakan metode Poll SCM), maka Jenkins akan secara otomatis mendeteksi perubahan pada repository dan memulai build baru. Proses ini bisa dilihat pada bagian Build History di halaman job Jenkins. Build ini akan menjalankan ulang seluruh tahap pipeline mulai dari checkout, clean up, hingga deploy ke server Nginx.

#### 5. Memverifikasi hasil perubahan yang ter-deploy

Setelah pipeline selesai dan build berhasil, dilakukan pengecekan ulang ke alamat

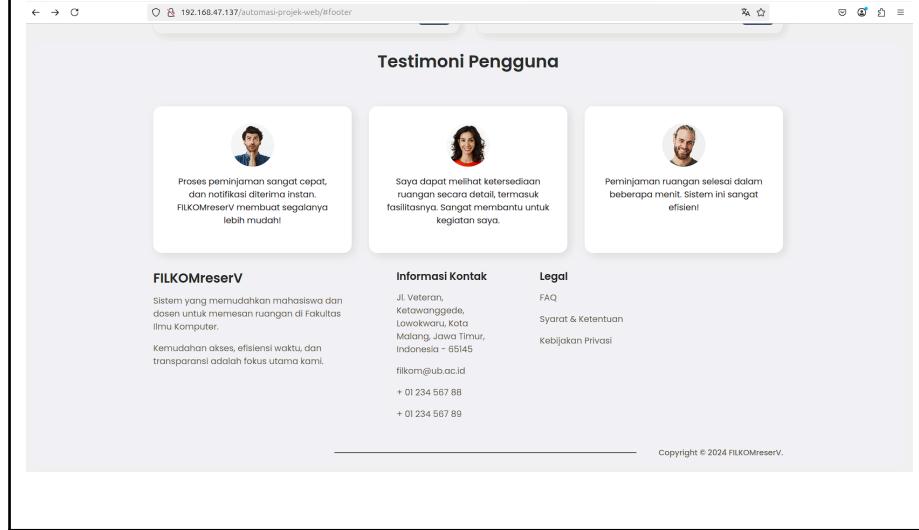
<http://192.168.47.137/automasi-projek-web/index.html>. Halaman web yang ditampilkan kini sudah menyesuaikan dengan perubahan terbaru pada index.html. Dengan ini akan membuktikan bahwa pipeline CI/CD berjalan otomatis dari proses push kode hingga deployment ke server tanpa perlu campur tangan manual.

Sebelumnya kami sudah mengclone repository git yang berisi website yang akan di deploy



ini adalah tampilan awal sebelum diubah untuk pengujian CI/CD

Dan ini adalah tampilan sesudah kode website di edit dan di push ke github yang akan mentrigger jenkins untuk men deploy ulang website dengan kode baru



### 3.1.6 Langkah 6: Konfigurasi Monitoring Metrics

#### 1. Menjalankan Blackbox Exporter (opsional)

Digunakan untuk memantau endpoint HTTP. Blackbox Exporter dijalankan dalam container Docker dengan port 9115.

#### 2. Menambahkan job Blackbox ke Prometheus

Konfigurasi prometheus.yml diperbarui untuk menambahkan job blackbox\_http yang memeriksa endpoint <http://localhost/automasi-projek-web/index.html> menggunakan module http\_2xx.

#### 3. Merestart Prometheus

Container Prometheus di-restart agar membaca konfigurasi baru, kemudian dicek pada <http://192.168.47.137:9090/targets> untuk memastikan job baru muncul dan statusnya UP.

```
Faiz@faiz-Virtual-Platform: ~ ls
automasi-projek-web Desktop Documents Downloads Music Pictures prometheus_data Public snap Templates Videos
Faiz@Faiz-Virtual-Platform: ~ cd prometheus_data
Faiz@Faiz-Virtual-Platform: ~/prometheus_data$ ls
prometheus
Faiz@Faiz-Virtual-Platform: ~/prometheus_data$ nano prometheus.yml
Faiz@Faiz-Virtual-Platform: ~/prometheus_data$
```

The terminal window displays the contents of the 'prometheus.yml' configuration file:

```

faiz@faiz-VMwareVirtual-Platform:~/prometheus_data
GNU nano 7.2
global:
 scrape_interval: 15s

scrape_configs:
 - job_name: 'prometheus'
 static_configs:
 - targets: ['localhost:9090'] # Prometheus memantau dirinya sendiri

 - job_name: 'node_exporter' # Untuk memantau VM Ubuntu kita
 static_configs:
 - targets: ['192.168.47.137:9100'] # Node Exporter akan berjalan di port ini

 - job_name: 'blackbox_http' # Memantau endpoint HTTP aplikasi kita
 metrics_path: /probe
 params:
 module: [http_2xx] # Menggunakan module http_2xx (cek status 2xx)
 static_configs:
 - targets:
 - http://localhost/automasi-projek-web/index.html
 relabel_configs:
 - source_labels: [__address__]
 target_label: __param_target
 - source_labels: [__param_target]
 target_label: instance
 - target_label: __address__
 replacement: 192.168.47.137:9115 # Alamat Blackbox exporter

```

The browser window shows the 'Status > Target health' page of the Prometheus interface, listing three targets:

- blackbox\_http**: Endpoint `http://192.168.47.137:9115/probe`, Labels `instance="http://localhost/automasi-projek-web/index.html"`, `job="blackbox_http"`. Last scrape: 134ms ago, State: UP.
- node\_exporter**: Endpoint `http://192.168.47.137:9100/metrics`, Labels `instance="192.168.47.137:9100"`, `job="node_exporter"`. Last scrape: 8.471s ago, State: UP.
- prometheus**: Endpoint `http://localhost:9090/metrics`, Labels `instance="localhost:9090"`, `job="prometheus"`. Last scrape: 8.914s ago, State: UP.

#### 4. Menjalankan Grafana untuk visualisasi

Grafana dijalankan pada Docker port 3000.

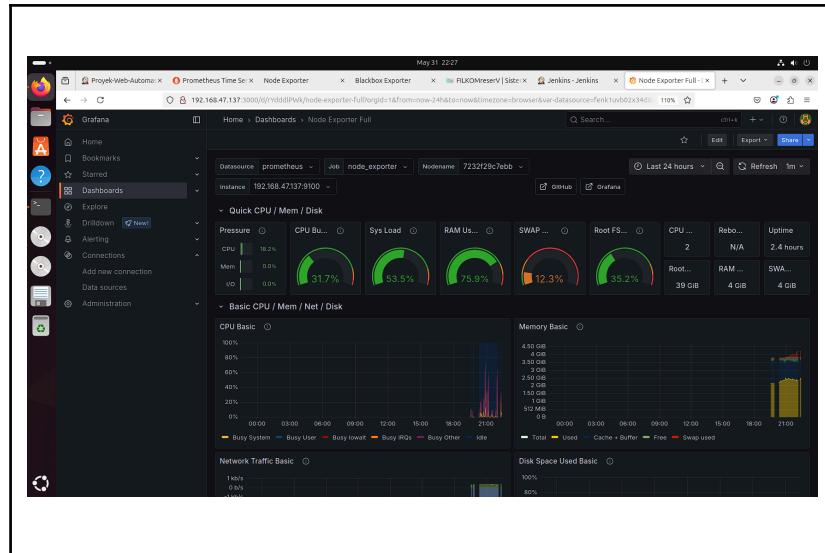
Akses Grafana melalui `http://192.168.47.137:3000` dengan login default admin/admin lalu ubah password.

##### a. Menambahkan Prometheus sebagai data source

Pada Grafana, di menu Configuration > Data Sources, ditambahkan Prometheus dengan URL `http://localhost:9090`. Kemudian diuji koneksi hingga muncul pesan "Data source is working".

##### b. Mengimpor dashboard Grafana

Untuk monitoring server (Node Exporter), dashboard siap pakai seperti ID 1860 ("Node Exporter Full") diimpor untuk mempermudah visualisasi CPU, Memory, dan metrics lainnya.



### 3.2 Hasil Pengujian

Bagian ini menjelaskan arsitektur sistem yang akan diimplementasikan untuk mencapai tujuan otomatisasi deployment dan monitoring aplikasi web sederhana menggunakan pipeline CI/CD di Virtual Machine. Desain sistem berfokus pada integrasi Git, Jenkins, Virtual Machine (Linux OS), Application Load Balancer (ALB), Auto Scaling Group (ASG), dan Prometheus untuk menciptakan alur kerja yang efisien, andal, dan dapat diskalakan.

## **BAB IV**

### **PEMBAGIAN TUGAS**

#### **4.1 Pembagian Kerja Tim**

##### **Anggota 1: Muhammad Faiz Fauzan**

Tugas:

1. Melakukan analisis dan merencanakan pengembangan atau pemilihan aplikasi web sederhana yang akan dijadikan studi kasus.
2. Merencanakan setup dan konfigurasi repository Git, termasuk strategi branching dan alur kerja kolaborasi tim.
3. Merencanakan instalasi dan konfigurasi Jenkins Server, termasuk merancang struktur dasar Jenkinsfile untuk tahapan checkout, build (jika perlu), dan test awal.
4. Merencanakan strategi pengujian fungsional sederhana untuk aplikasi dan pipeline awal.
5. Berkontribusi dalam penyusunan bagian proposal dan dokumentasi awal terkait pemilihan aplikasi, manajemen kode, dan desain pipeline CI/CD.

##### **Anggota 1: Muhammad Faiz Fauzan**

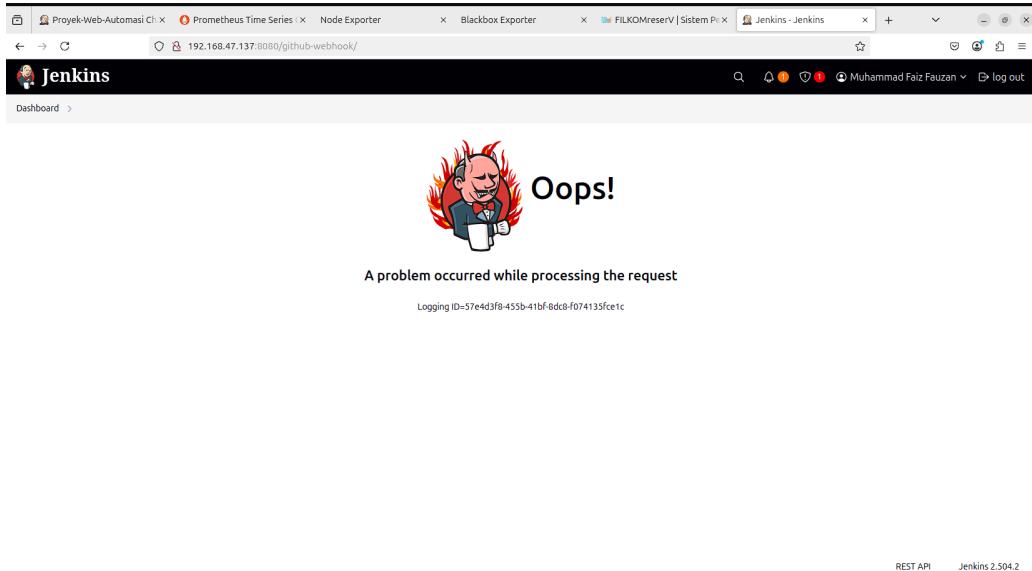
Tugas:

1. Menganalisis kebutuhan dan merancang arsitektur infrastruktur menggunakan virtual machine berbasis Linux, meliputi perencanaan untuk instance virtual machine (aplikasi, Jenkins, Prometheus)
2. Melakukan perencanaan dan analisis untuk implementasi Auto Scaling Groups pada lingkungan virtual machine.
3. Merencanakan arsitektur dan strategi monitoring menggunakan Prometheus, termasuk penentuan metrik kunci dan perencanaan instalasi exporters pada virtual machine.
4. Merencanakan strategi pengujian performa sederhana untuk infrastruktur dan aplikasi.
5. Berkontribusi dalam penyusunan bagian proposal dan dokumentasi awal terkait desain infrastruktur berbasis virtual machine Linux, strategi deployment, rencana monitoring, dan analisis komparatif.

## BAB V

### EVALUASI

#### 5.1 Masalah yang Ditemui



Saat ingin Add webhook, dan mencoba mengakses URL <http://192.168.47.137:8080/github-webhook/> di browser dan mendapatkan halaman error "Oops!" dari Jenkins, itu bukan berarti ada yang rusak parah, melainkan karena URL tersebut adalah sebuah endpoint khusus yang dirancang untuk menerima notifikasi otomatis (HTTP POST) dari GitHub, bukan untuk dibuka langsung oleh pengguna melalui browser (yang mengirimkan HTTP GET). Webhook sendiri adalah mekanisme di mana GitHub memberitahu Jenkins setiap kali ada perubahan kode, seperti git push, dengan mengirimkan data ke "Payload URL" yang telah ditentukan. Jenkins kemudian akan memproses data ini untuk memicu build secara otomatis.

Masalah utama dalam menggunakan webhook untuk Jenkins yang berjalan di VM lokal (seperti 192.168.47.137) adalah aksesibilitas. Server GitHub yang berada di internet tidak bisa secara langsung mengirimkan notifikasi ke alamat IP lokal di jaringan Anda tanpa konfigurasi jaringan lanjutan yang cukup rumit, seperti port forwarding di router atau menggunakan layanan tunneling (misalnya ngrok). Tanpa itu, webhook dari GitHub tidak akan pernah sampai ke Jenkins Anda.

## **5.2 Solusi yang Dilakukan**

Solusi yang lebih praktis dan sangat direkomendasikan untuk setup VM lokal adalah menggunakan fitur "Poll SCM" di Jenkins. Dengan "Poll SCM", peran dibalik: Jenkins yang secara aktif dan berkala (misalnya, setiap beberapa menit) akan "bertanya" atau mengecek ke repository GitHub Anda apakah ada commit atau perubahan baru. Jika Jenkins menemukan perubahan, ia akan otomatis memulai proses build. Metode ini bekerja dengan baik untuk VM lokal karena Jenkins yang melakukan koneksi keluar ke GitHub (yang umumnya diizinkan oleh jaringan), sehingga tidak memerlukan Jenkins Anda bisa diakses dari internet publik. Untuk mengkonfigurasinya, Anda cukup membuka konfigurasi Jenkins Job Anda, masuk ke bagian "Build Triggers", mencentang "Poll SCM", dan mengatur jadwal pengecekan menggunakan format cron (contoh: H/5 \* \* \* \* untuk setiap 5 menit). Dengan fokus pada "Poll SCM", Anda bisa mendapatkan fungsionalitas pemicu build otomatis tanpa kompleksitas pengaturan jaringan untuk webhook.

**BAB Vi**  
**KESIMPULAN**

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX