Otomatisasi Deployment dan Monitoring Aplikasi Web Sederhana menggunakan CI/CD Pipeline di AWS

Penyediaan dan Automasi Layanan

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

Latar Belakang

- Muncul kebutuhan untuk merancang sistem deployment yang sepenuhnya otomatis, terutama dengan alat seperti Git dan Jenkins untuk CI/CD di platform cloud seperti AWS.

- Tanpa otomasi dan monitoring yang memadai, identifikasi dan penanganan masalah menjadi lambat dan bersifat reaktif.

Rumusan Masalah

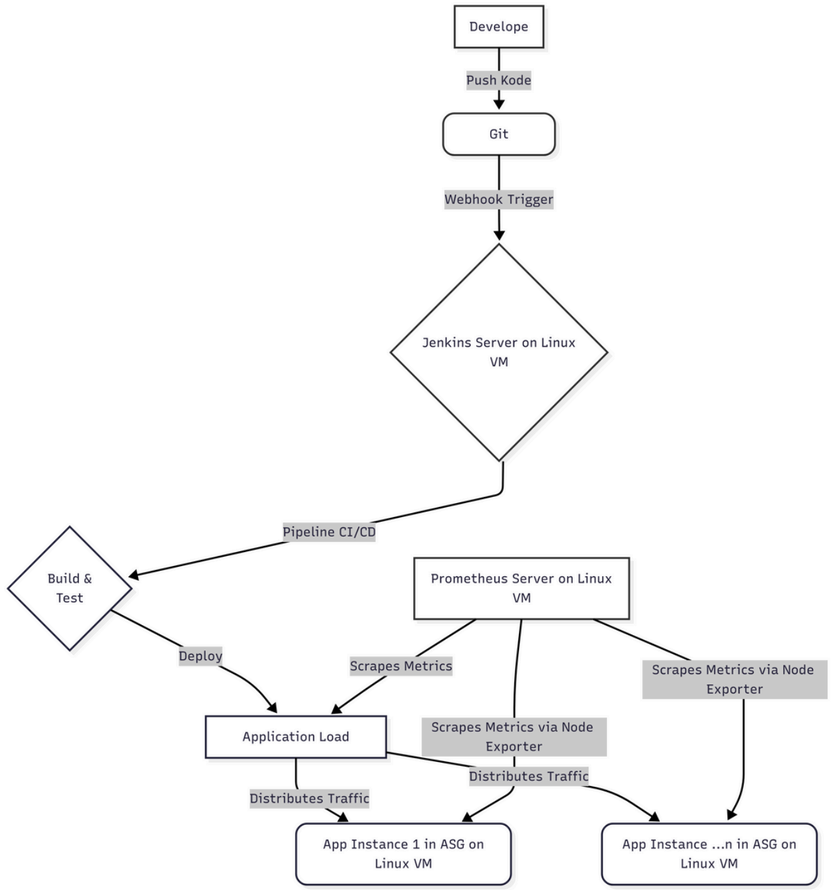
- Bagaimana merancang dan mengimplementasikan sebuah sistem deployment aplikasi web yang sepenuhnya otomatis dari repository kode hingga aplikasi berjalan di server?
- Bagaimana memanfaatkan Git dan Jenkins untuk membangun pipeline Continuous Integration/Continuous Deployment (CI/CD) yang efektif untuk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

Tujuan Proyek

- Mengimplementasikan pipeline CI/CD menggunakan Git dan Jenkins untuk otomatisasi proses build, test, dan deploy aplikasi web sederhana.
- Melakukan deployment aplikasi web ke infrastruktur AWS EC2 yang didukung oleh Application Load Balancer untuk distribusi trafik dan peningkatan ketersediaan.
- Menganalisis, merencanakan, dan (jika waktu memungkinkan) mengimplementasikan dasar-dasar Autoscaling untuk menyesuaikan kapasitas aplikasi secara otomatis berdasarkan permintaan.
- Mengimplementasikan sistem monitoring dasar menggunakan Prometheus untuk memantau metrik kunci dari aplikasi dan infrastruktur AWS.

Desain Solusi

Diagram Alur Sistem



Penjelasan Komponen Utama:

- Git: Kontrol versi dan pemicu.
- Jenkins: Orkestrasi CI/CD.
- AWS EC2: Hosting aplikasi, Jenkins, Prometheus.
- AWS ALB: Distribusi trafik, ketersediaan.
- AWS ASG: Skalabilitas otomatis (direncanakan).
- Prometheus: Monitoring.

Alur Kerja CI/CD:

1. Developer push kode ke Git.
2. Webhook memicu build di Jenkins.
3. Jenkins: Checkout -> (Build) -> Test -> Deploy.
4. Aplikasi di-deploy ke instance EC2 di belakang ALB.
5. Prometheus memantau metrik.

Desain Solusi

Penjelasan Komponen Utama:

- Git: sistem kontrol versi untuk mengelola perubahan kode dan sebagai pemicu (trigger)
- Jenkins: Sebagai server otomasi utama
- Prometheus: Monitoring.

Alur Kerja CI/CD:

1. Developer push kode ke Git.
2. Webhook memicu build di Jenkins.
3. Jenkins: Checkout -> (Build) -> Test -> Deploy.
4. Aplikasi di-deploy ke instance EC2 di belakang ALB.
5. Prometheus memantau metrik.

Rencana Penggunaan

Git (GitHub/GitLab):

akan digunakan untuk manajemen versi kode aplikasi dan Jenkinsfile, dengan strategi branching sederhana yang memicu pipeline CI/CD otomatis melalui webhook.

Jenkins

di-hosting di AWS EC2, akan menjadi engine utama pipeline CI/CD. Pipeline (didefinisikan via Jenkinsfile) akan mengotomatisasi checkout, build, tes sederhana, dan deployment ke AWS.

Rencana Penggunaan

AWS Deployment

Platform Utama (EC2, ALB, ASG): Aplikasi web akan di-deploy ke AWS EC2. Application Load Balancer (ALB) akan mendistribusikan trafik untuk ketersediaan dan sebagai dasar skalabilitas. AWS Auto Scaling Groups (ASG) akan direncanakan dan dianalisis untuk penyesuaian kapasitas otomatis berdasarkan metrik, dengan implementasi awal mungkin berfokus pada penyiapan dasar.

Kubernetes/EKS

AWS EKS akan dieksplorasi dan dianalisis sebagai alternatif deployment canggih untuk aplikasi berbasis container, sebagai studi perbandingan atau pengembangan masa depan. Implementasi penuh EKS di luar cakupan awal.

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.

Pembagian Kerja Tim

Anggota 2: Fira Zaha Iklila

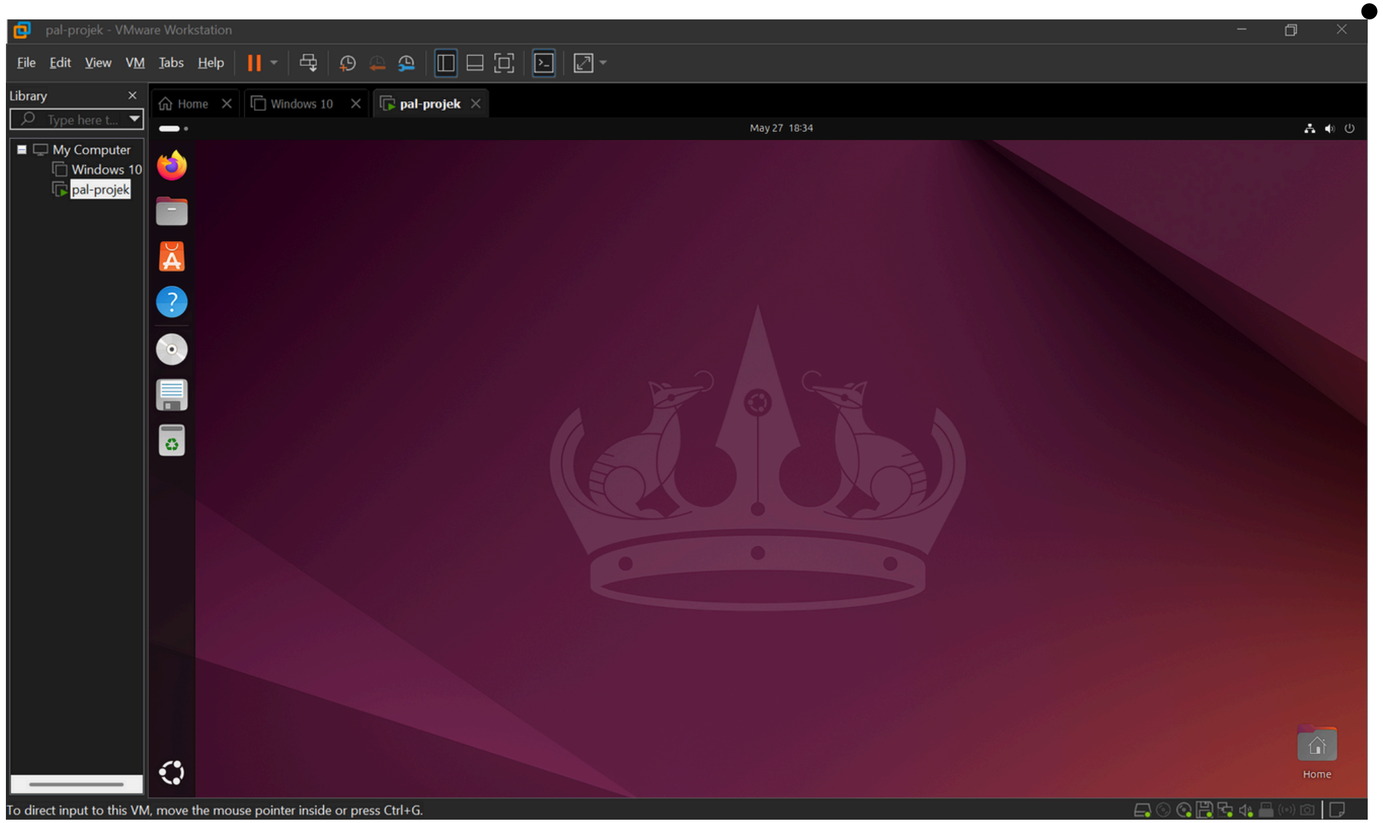
Tugas:

1. Menganalisis kebutuhan dan merancang arsitektur infrastruktur di AWS, meliputi perencanaan untuk instance EC2 (aplikasi, Jenkins, Prometheus), Security Groups, IAM Roles, dan Application Load Balancer.
2. Melakukan perencanaan dan analisis untuk implementasi AWS Auto Scaling Groups.
3. Merencanakan arsitektur dan strategi monitoring menggunakan Prometheus, termasuk penentuan metrik kunci dan perencanaan instalasi exporters.
4. Merencanakan strategi pengujian performa sederhana untuk infrastruktur dan aplikasi.
5. Melakukan analisis komparatif awal mengenai opsi deployment (EC2 dengan ASG vs Kubernetes/EKS).
6. Berkontribusi dalam penyusunan bagian proposal dan dokumentasi awal terkait desain infrastruktur AWS, strategi deployment, rencana monitoring, dan analisis komparatif,

Progess

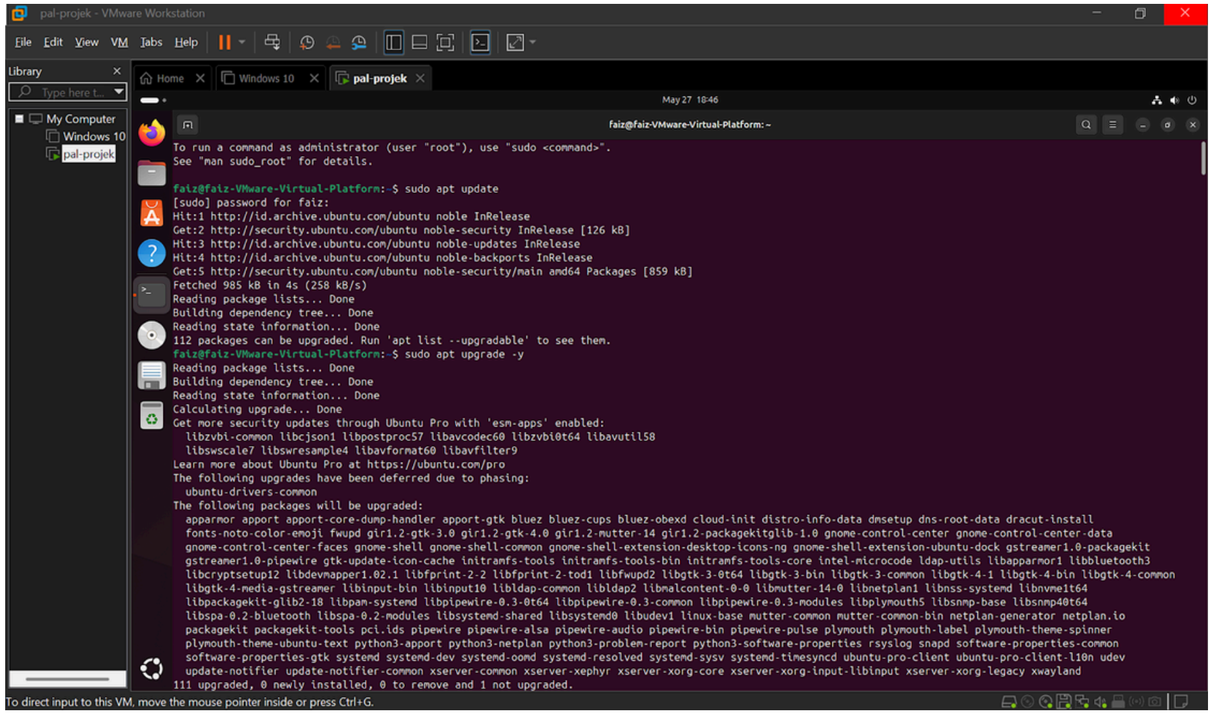
Langkah 0: Persiapan Awal

Instal VMware Workstation/Player di komputermu.



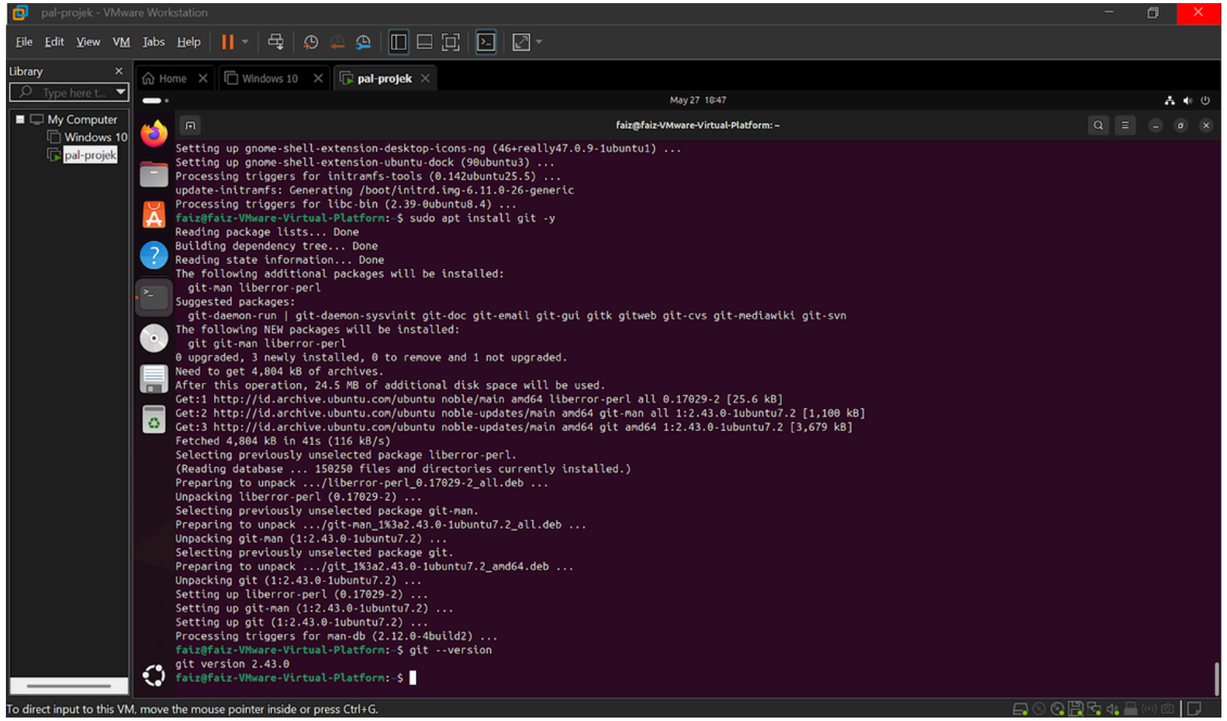
Langkah 1: Instalasi Software Pendukung di VM Ubuntu

Update Sistem



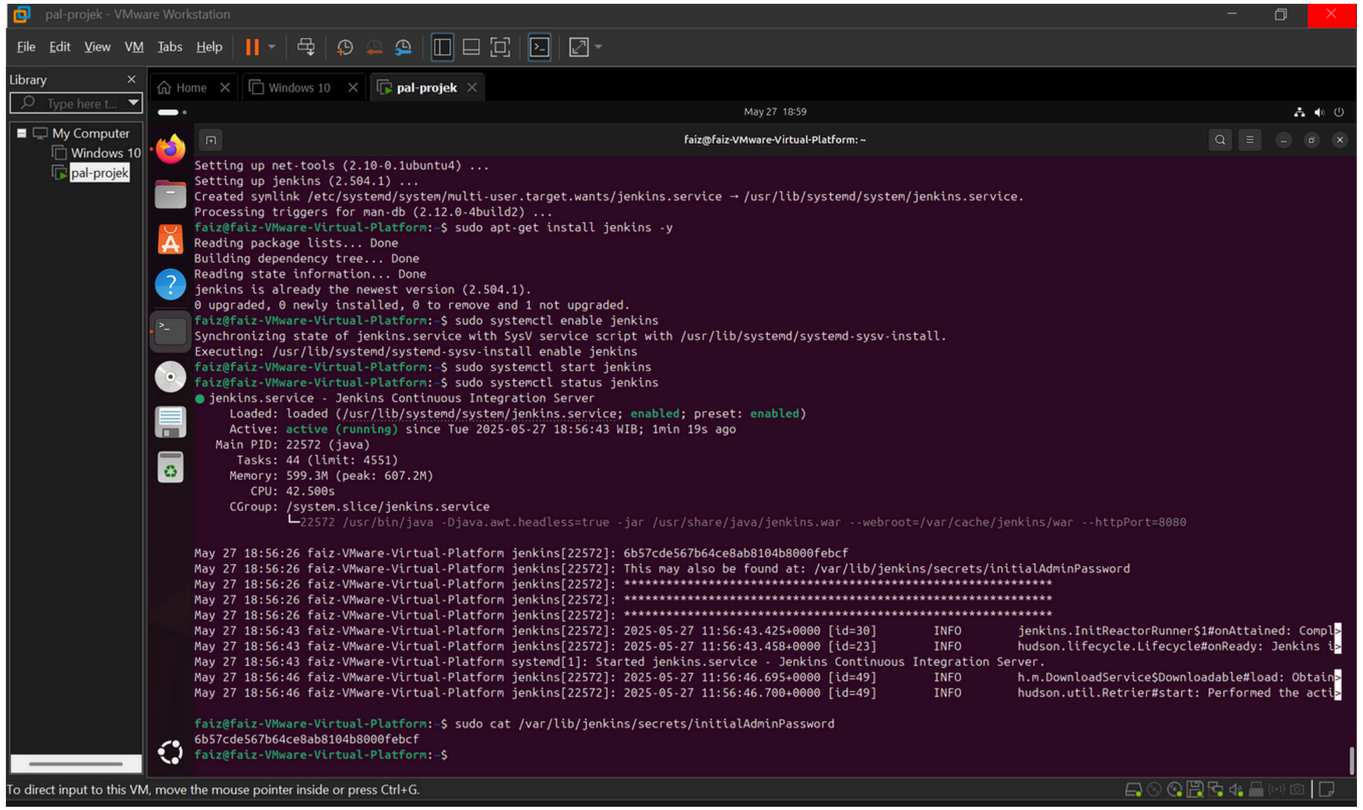
Langkah 1: Instalasi Software Pendukung di VM Ubuntu

Instal Git



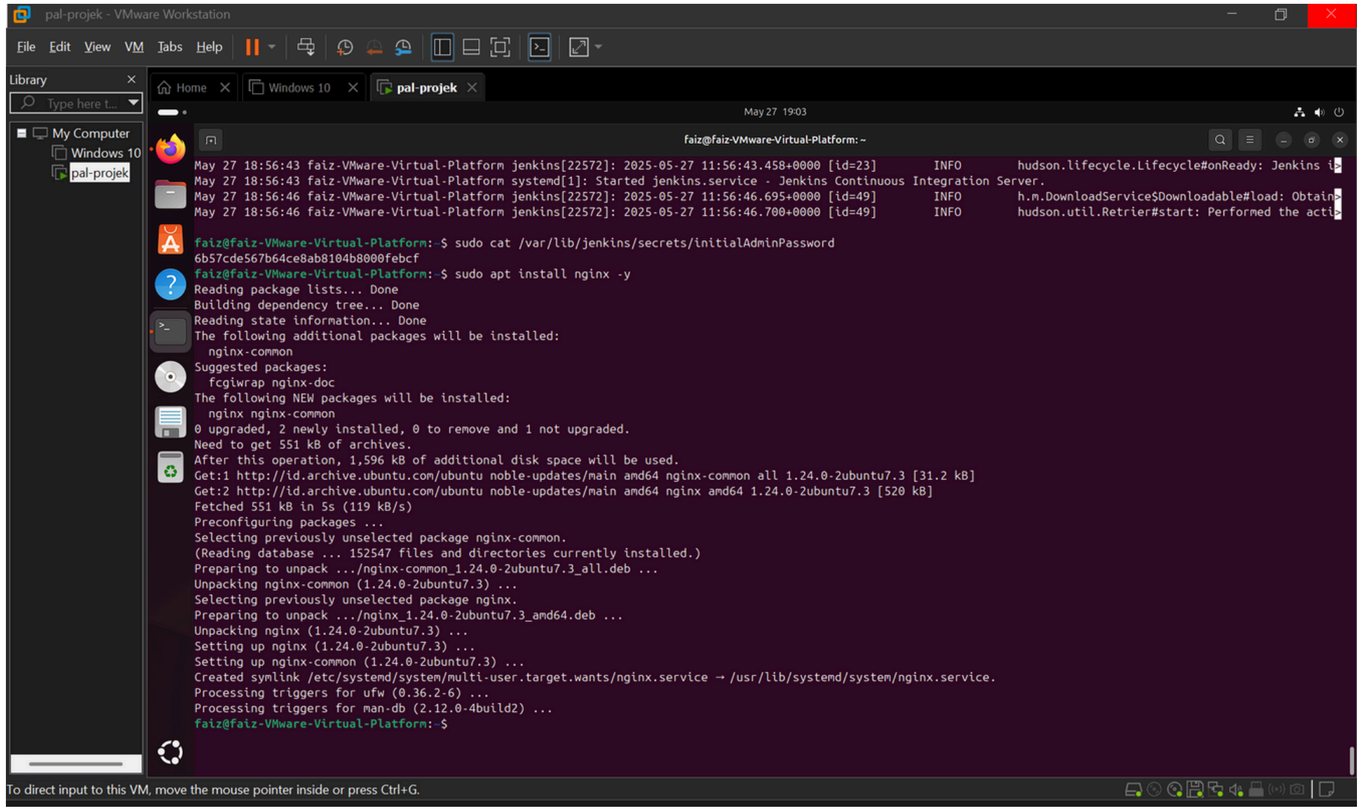
Langkah 1: Instalasi Software Pendukung di VM Ubuntu

Instal Jenkins



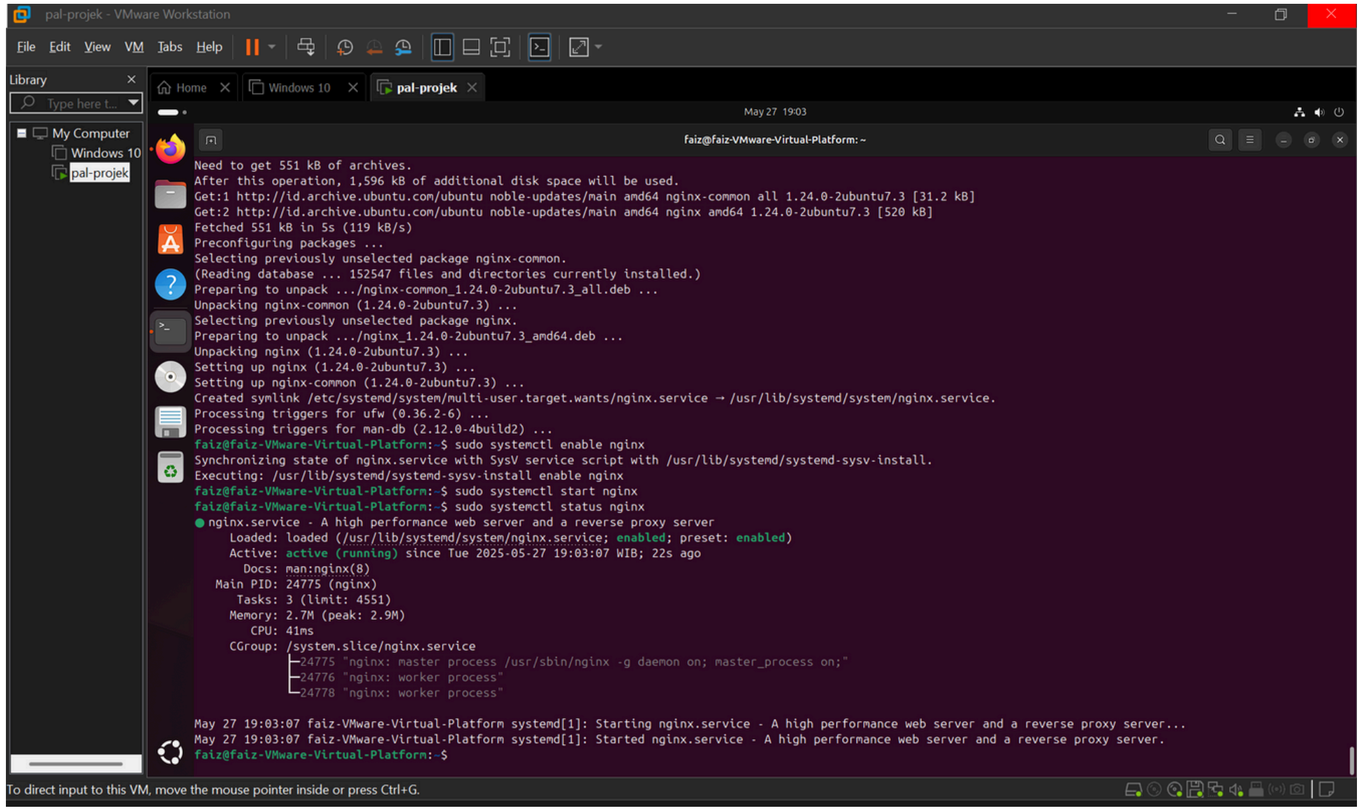
Langkah 1: Instalasi Software Pendukung di VM Ubuntu

Instal Nginx



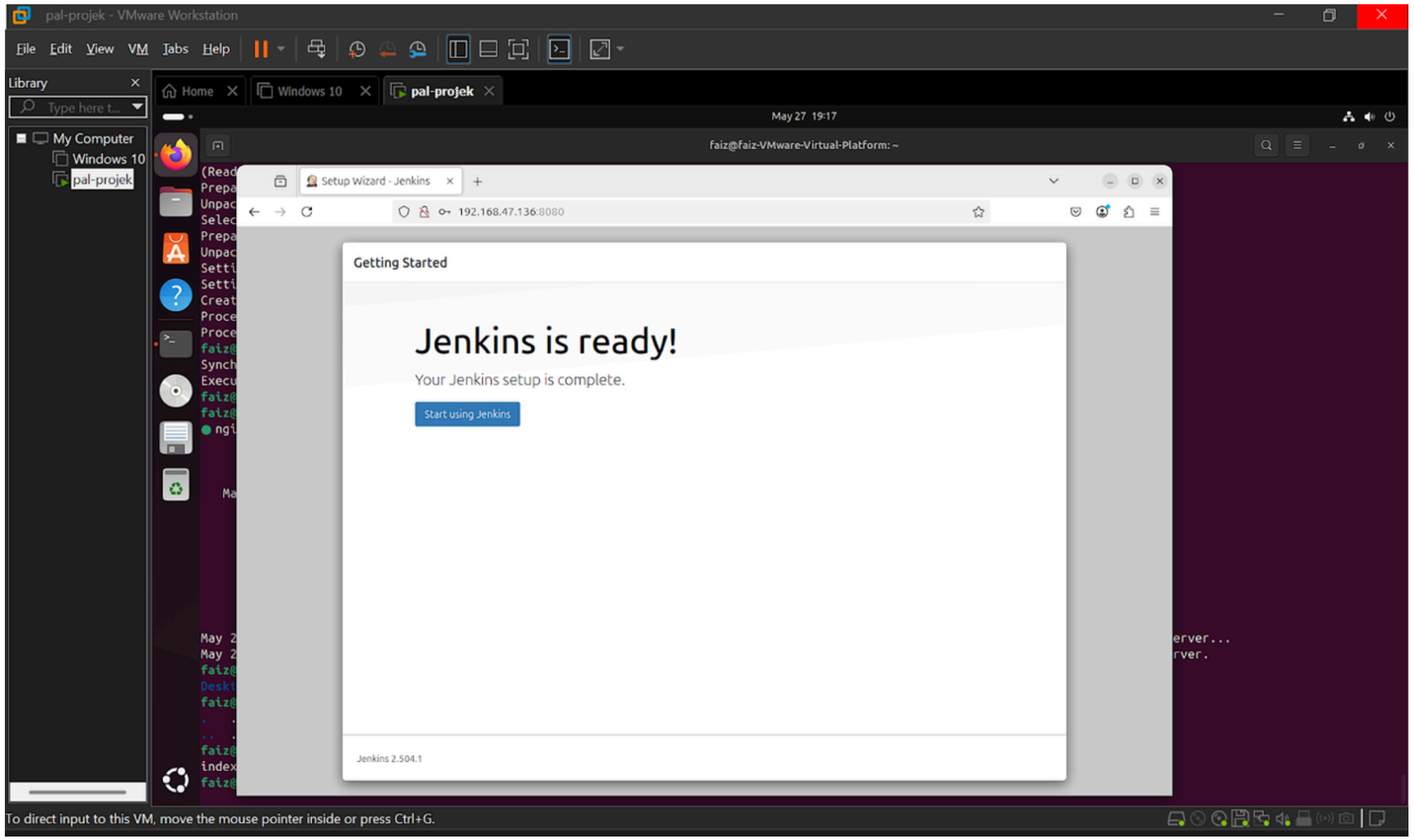
Langkah 1: Instalasi Software Pendukung di VM Ubuntu

Start dan enable semua aplikasi yang sudah kami install sebelumnya (Jenkins, Nginx)



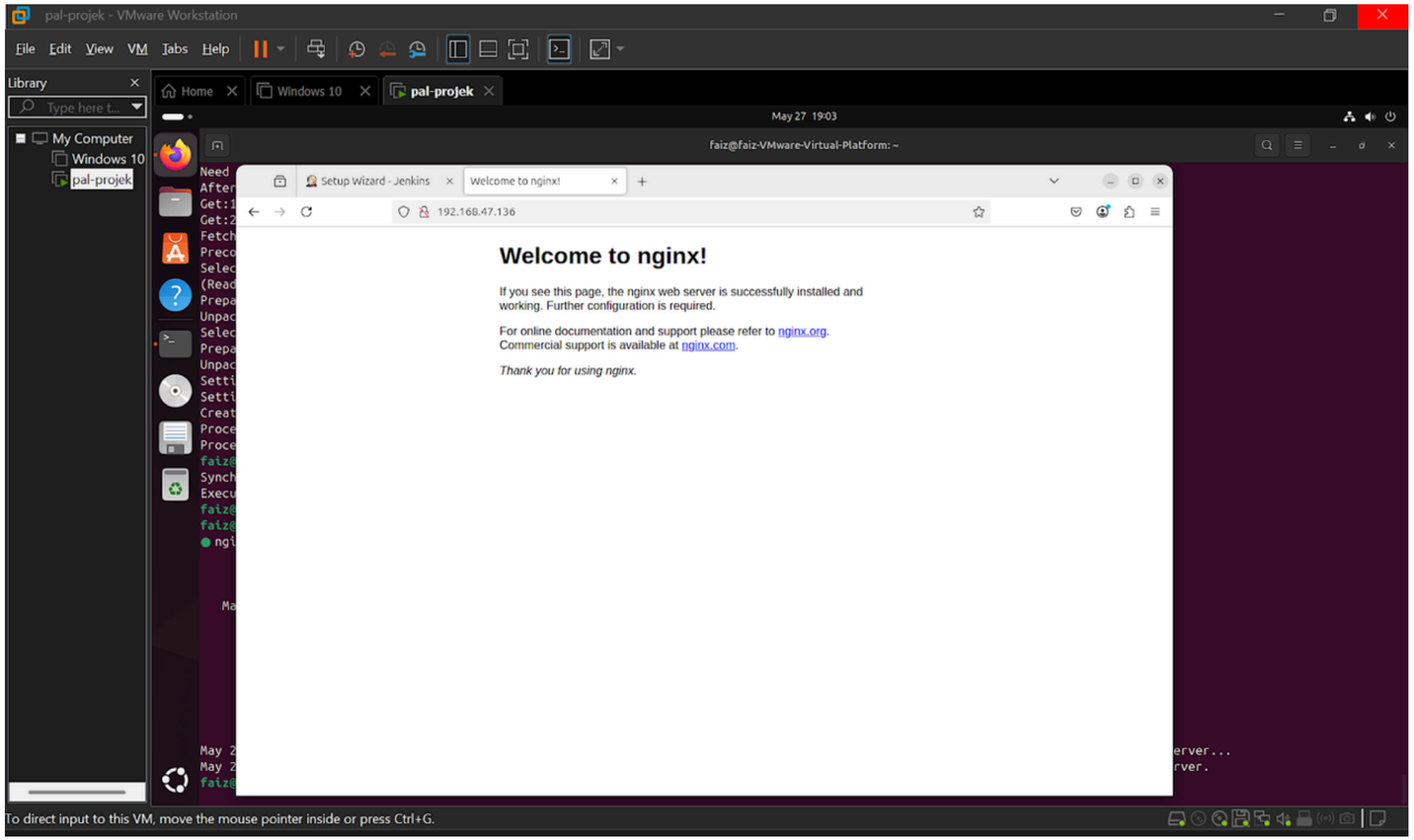
Langkah 1: Instalasi Software Pendukung di VM Ubuntu

Set up Jenkins



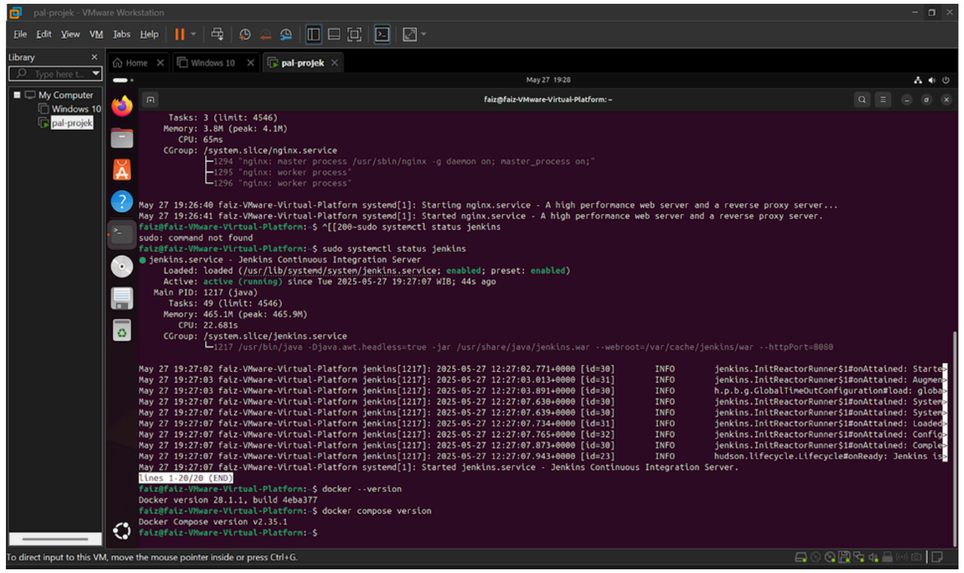
Langkah 1: Instalasi Software Pendukung di VM Ubuntu

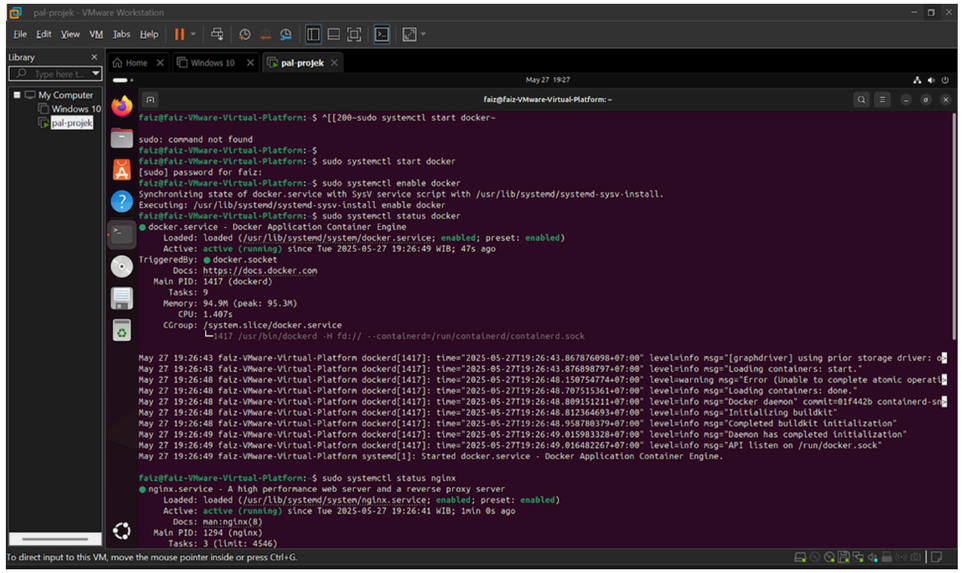
Set up Nginx



Langkah 1: Instalasi Software Pendukung di VM Ubuntu

Instal Docker & Docker Compose





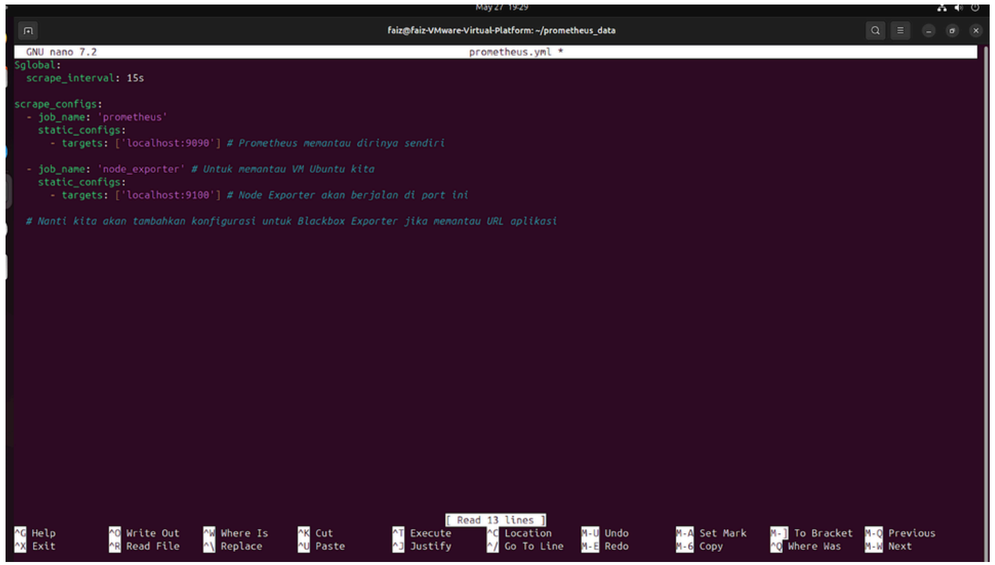
Langkah 1: Instalasi Software Pendukung di VM Ubuntu

Instal Prometheus & Node Exporter

Buat direktori untuk konfigurasi Prometheus:

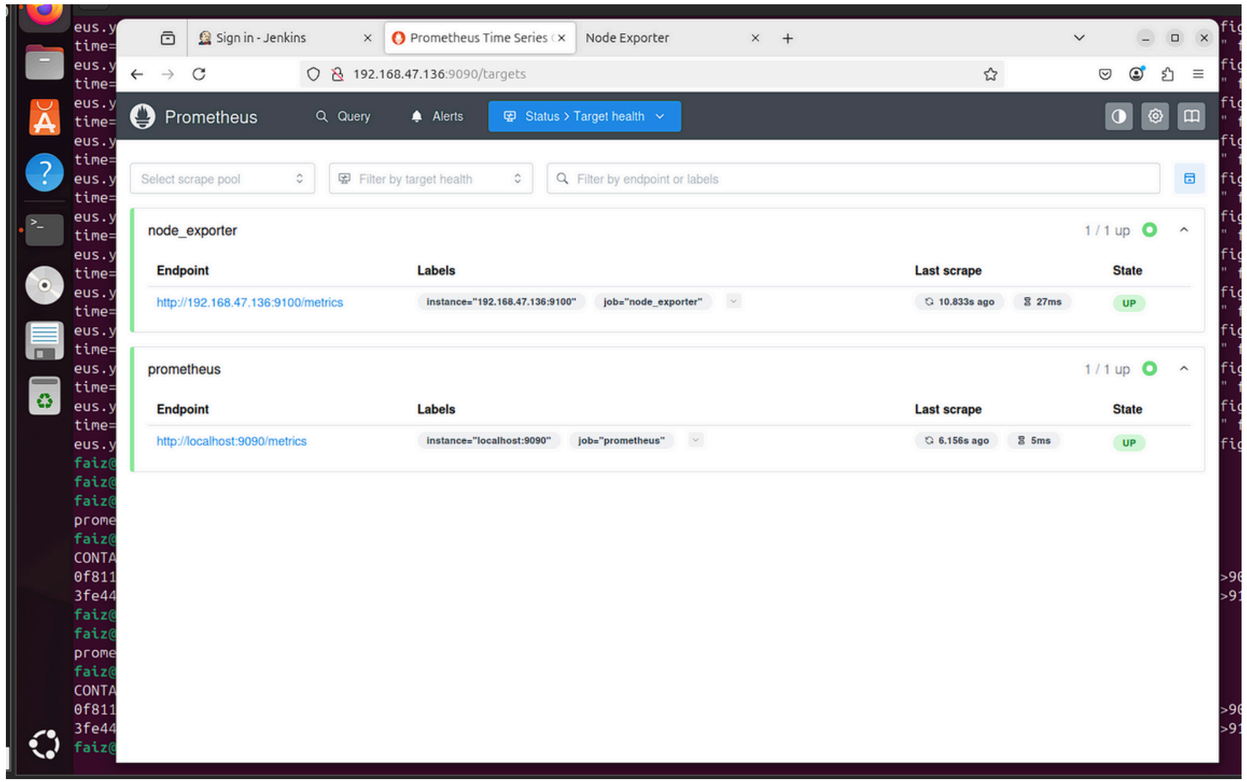
```
faiz@faiz-VMware-Virtual-Platform:~$ mkdir ~/prometheus_data
faiz@faiz-VMware-Virtual-Platform:~$ cd ~/prometheus_data
faiz@faiz-VMware-Virtual-Platform:~/prometheus_data$ nano prometheus.yml
faiz@faiz-VMware-Virtual-Platform:~/prometheus_data$
```

Buat file konfigurasi prometheus.yml:



Langkah 1: Instalasi Software Pendukung di VM Ubuntu

Set up Prometheus



Langkah 2: Setup Aplikasi Web Sederhana

Clone repository ke direktori home di VM (git clone https://github.com/faizfznn/automasi-projek-web.git)

```
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:~$
```

Langkah 3: Konfigurasi Jenkins untuk CI/CD

a. Konfigurasi Credentials di Jenkins

Di VM Ubuntu, buat SSH key baru (jika belum punya):
ssh-keygen -t rsa -b 4096 -C "jenkins@<nama_vm_mu>"

```
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 fingerprint is:
SHA256:4nQJW48abvnsz+KDyJhuaX/xbvItldo+BNWDO2UOf1s jenkins@faiz
The key's randomart image is:
+---[RSA 4096]----+
|           o     |
|          + =    |
|      . .. B .   |
|       +.+o o . E|
|      = S..o . o |
|     +.*  +   .  |
|   = .*+ =       |
|  * o.+oBoo      |
| +.... BB*=.     |
+----[SHA256]-----+
```

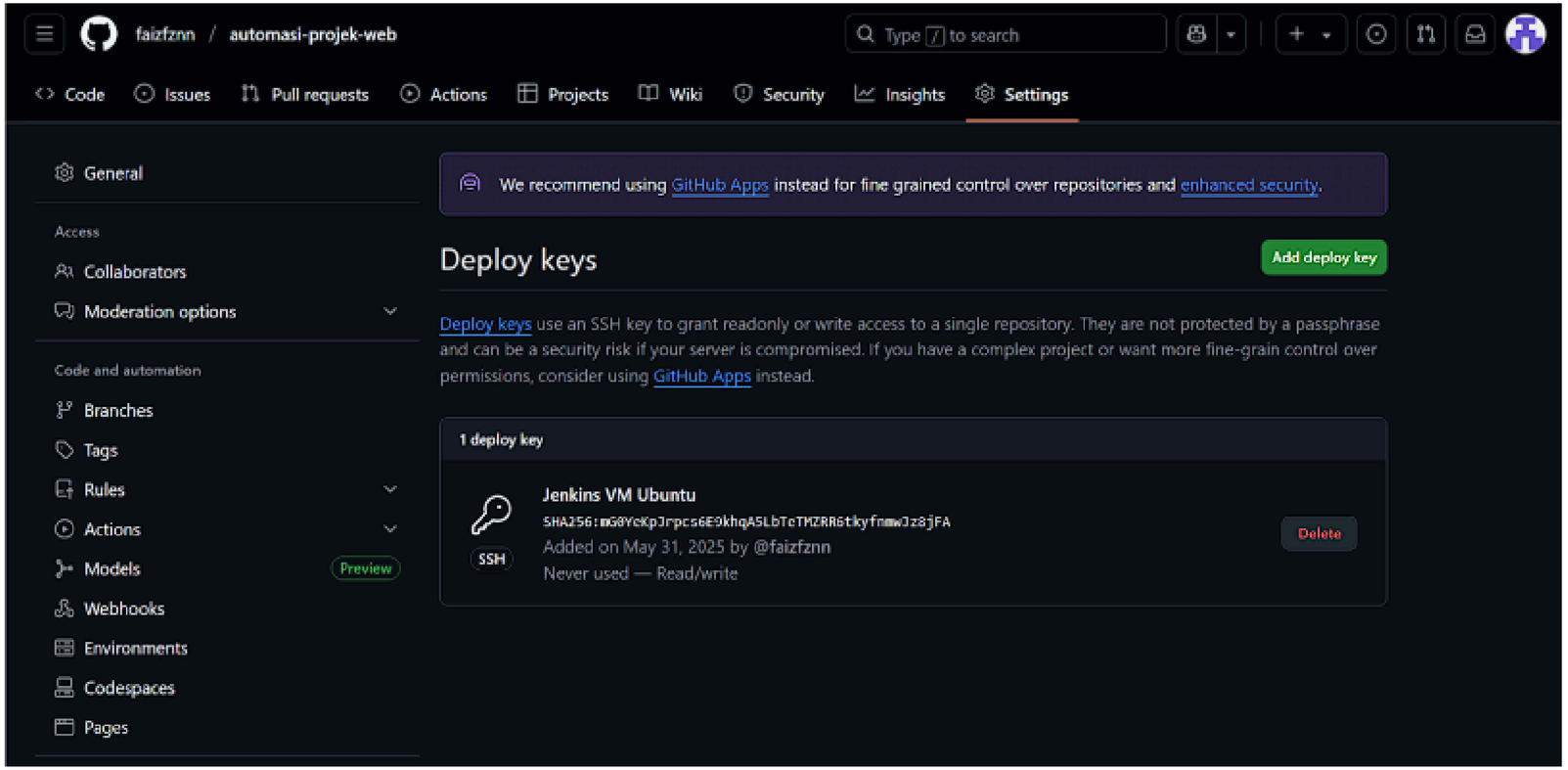
Tampilkan public key

```
+----[SHA256]-----+
faiz@faiz-VMware-Virtual-Platform:~$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDp4LXOtDOgGmUUy31Guhc8phmv+d+DSP3tKY8x07n3F4nGzefeoMXNoYF6gUc1ad/ciqY+cGskznSDreZtOoOswU7WwC1eXEdFJ/LuMtyTtudSTCXeLCq8Eii8JUl
Qc8B5qTs62qcyVFOlsRSlh+nJZk6d8KA+qj8ykKti5808kvi7TEyFn0odVwjNzdgzdBpWOWele7jI3JLrY2DmOdRTXh9FMycaSDPVfkE3ChU9AlM3vSp58N8ADJs8H4F9mITqxx5w647bzl0dbZawmN/lF198yiIz9E
cl9BLr+Gjyg1fYUW8EgYLfAk+BrD8XJSGcF92j+NDfWPky8L8dUXtXWA2irRcsqFK23fwhD8jlx7LWFAlzE9vwL8PtvF7XQe8PW7gkp2sJaVPO0EsrVDV+N0c0LHWXj0XUuvMyxXS/ePRWIHr0RX56H4hW62pC/RN0e
sJPLNVIU8LGEP9d7mlUqdM3CaSt7TWWcmVQPW7A7ONzcD8hTKRoOZUaeZlDC/Ojj/gt8bT/lQ/MXGZqdErxpLNFxh5zfqtYNKWB3gHaZCms+crkLYCd+O5hl9uc9N86qbjHZunkskB3gN28bdtWWKSyNTB8tKex6CUH
ytCZSG3+SaiknY7Xezt8g5JACtDBCSNxr+rnVW1zWroSyod8UaEpKoc/NsE8N0T73gCJYQ== jenkins@faiz
faiz@faiz-VMware-Virtual-Platform:~$ S
```

Langkah 3: Konfigurasi Jenkins untuk CI/CD

a. Konfigurasi Credentials di Jenkins

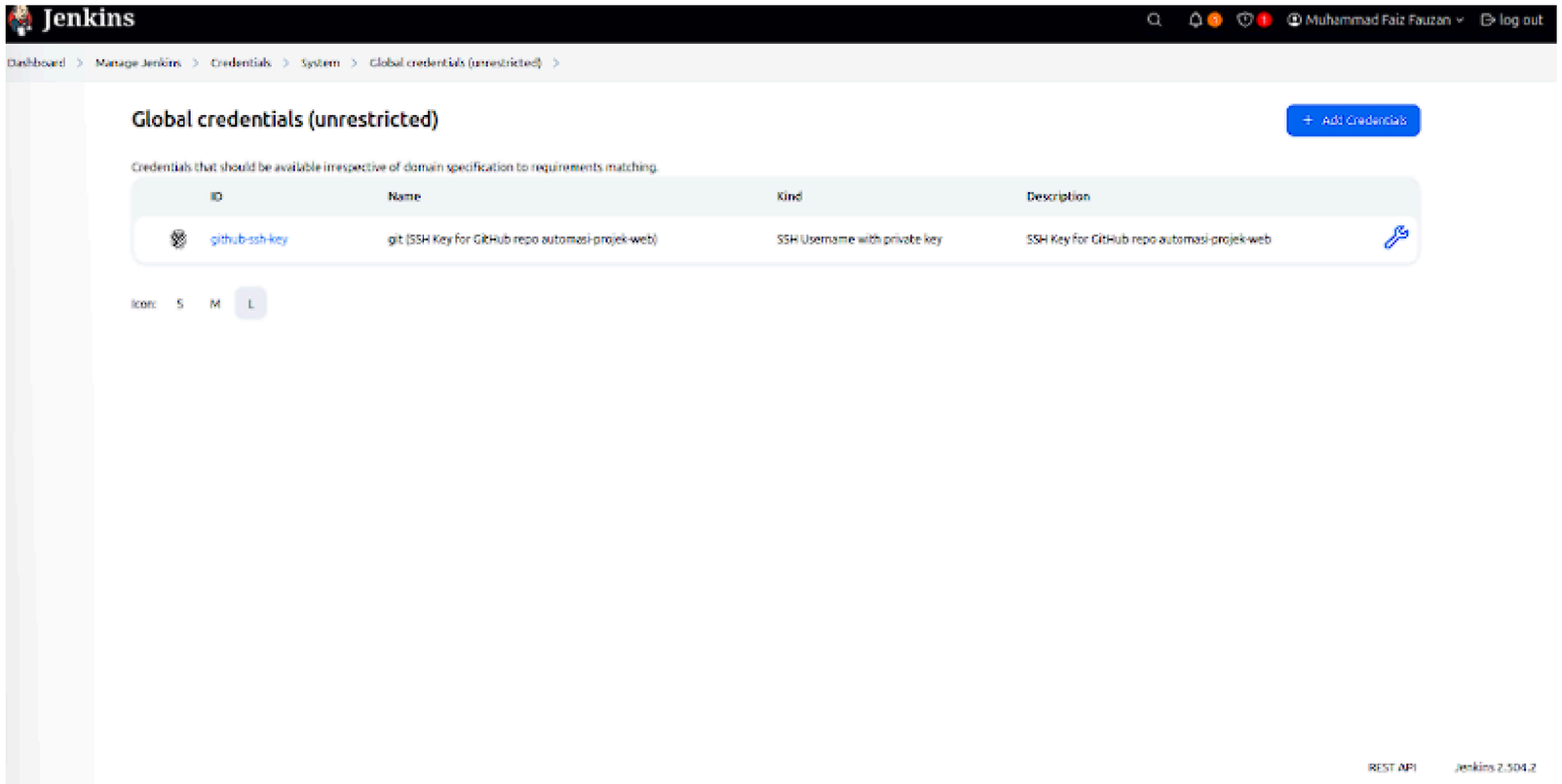
Tambahkan Public Key ke GitHub Repository



Langkah 3: Konfigurasi Jenkins untuk CI/CD

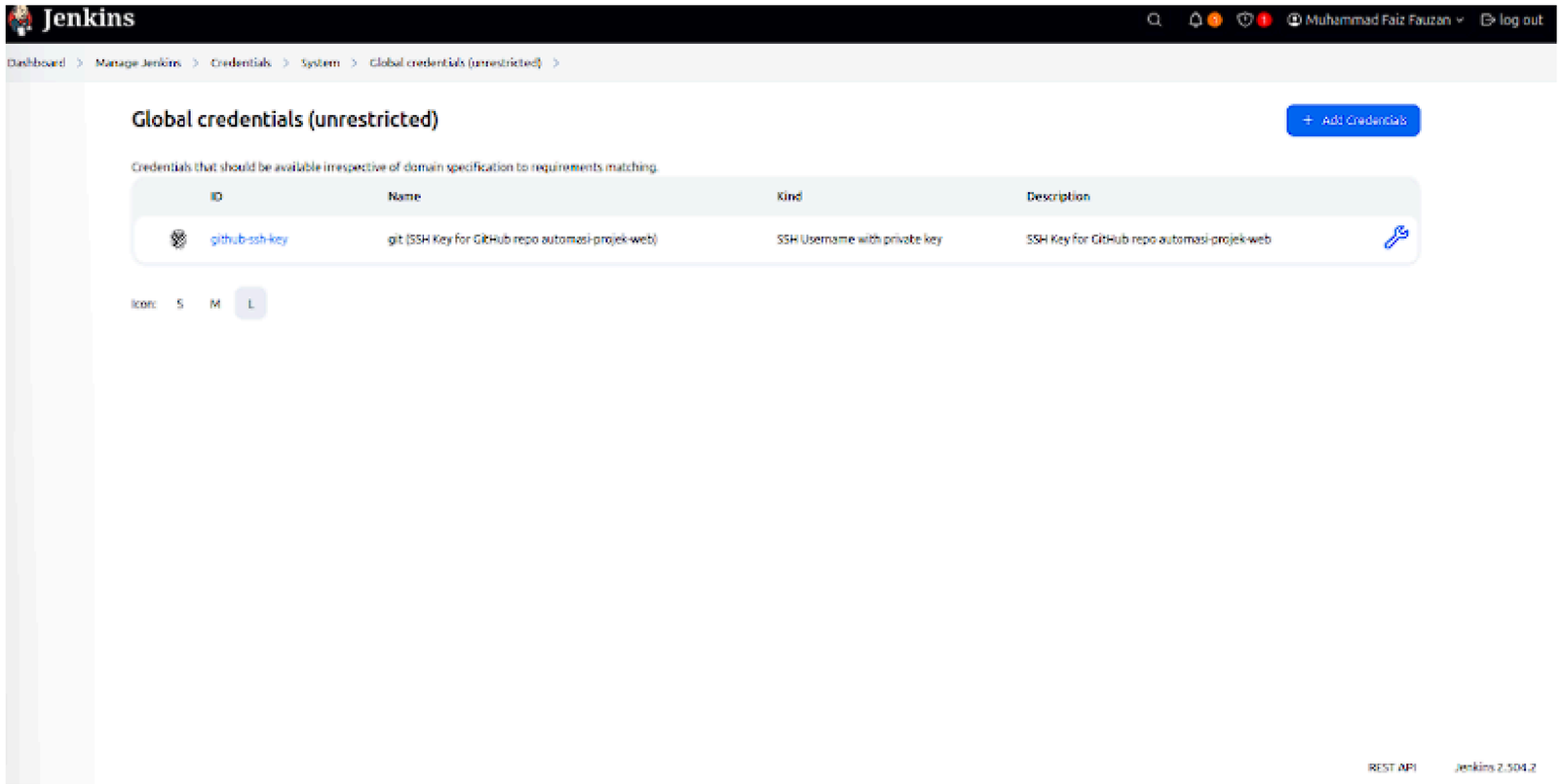
a. Konfigurasi Credentials di Jenkins

Tambahkan Private Key ke Jenkins Credentials



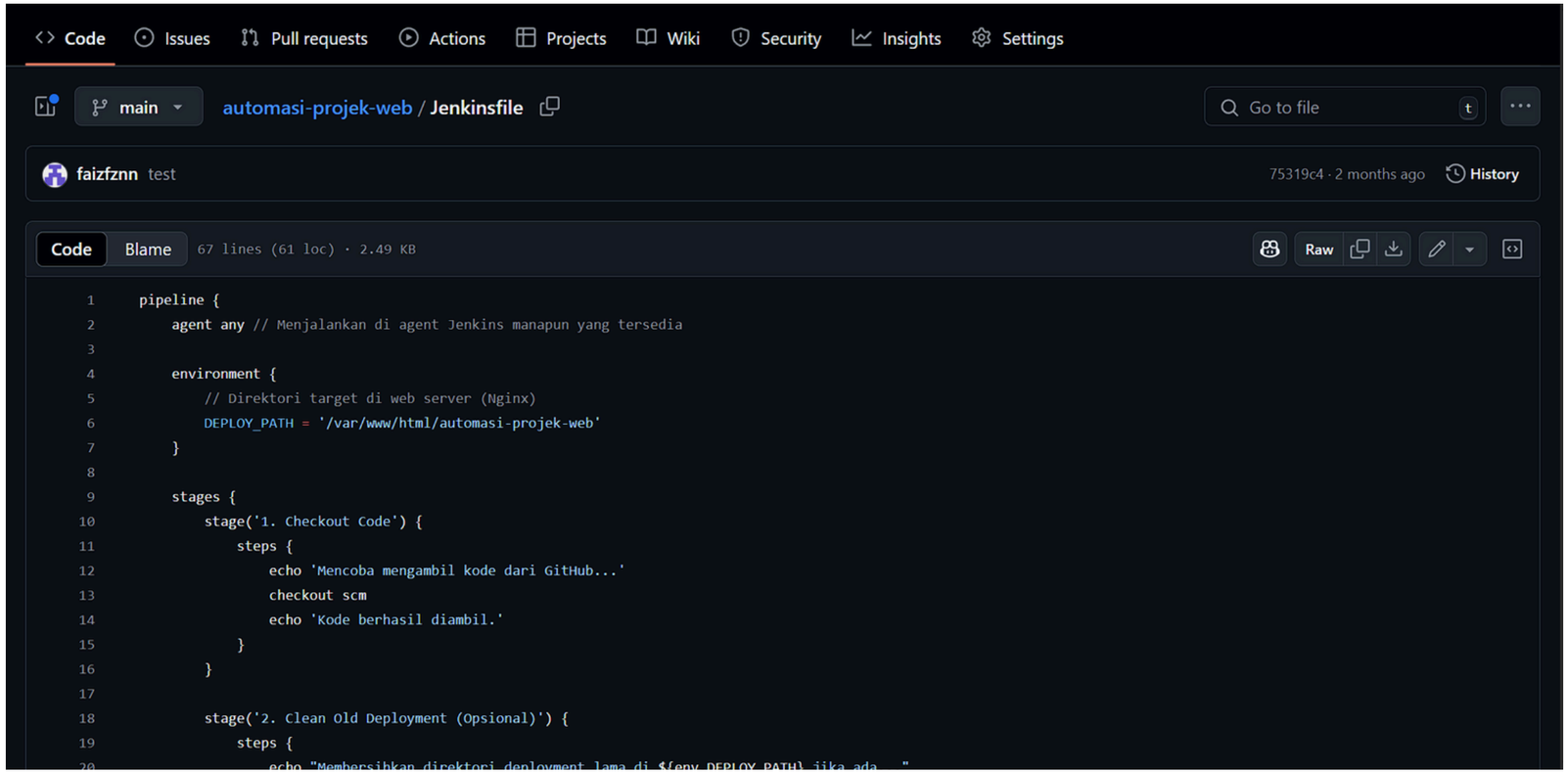
Langkah 3: Konfigurasi Jenkins untuk CI/CD

b. Buat Jenkins Pipeline Job



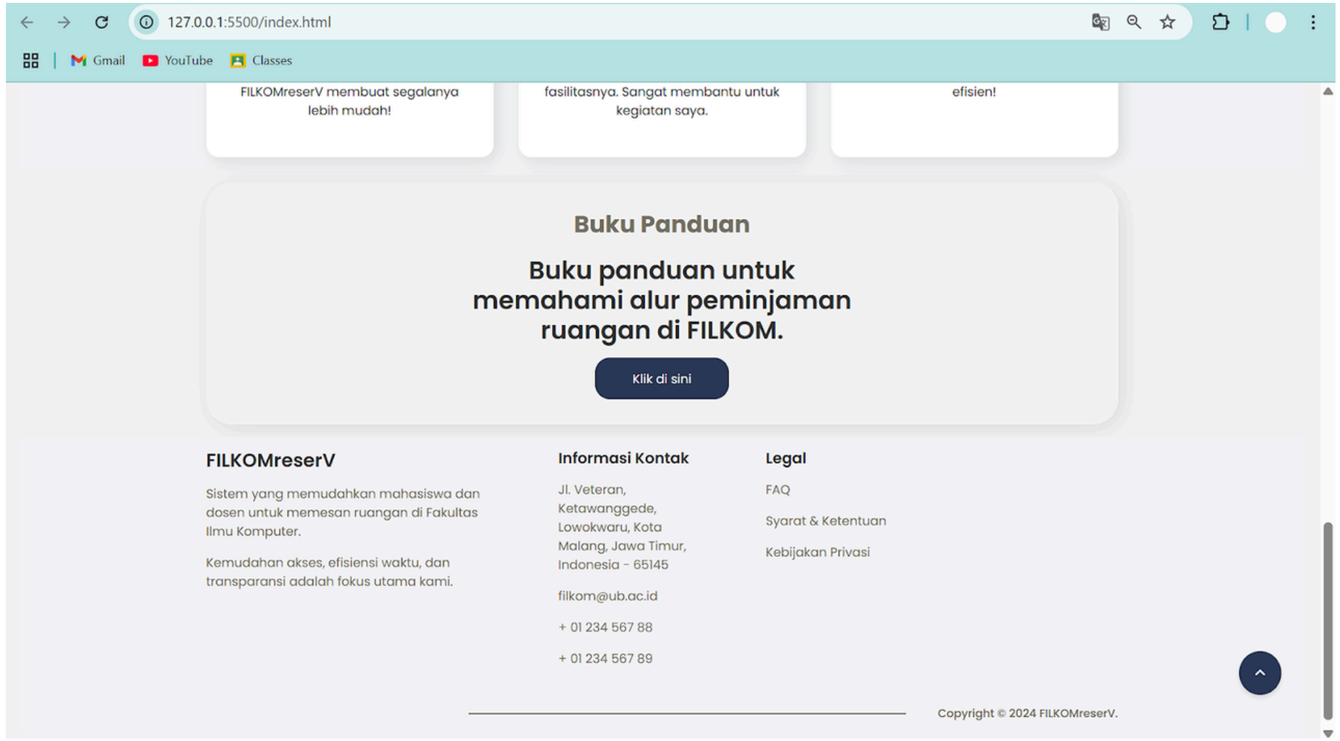
Langkah 3: Konfigurasi Jenkins untuk CI/CD

c. Buat Jenkinsfile di Repository GitHub-mu



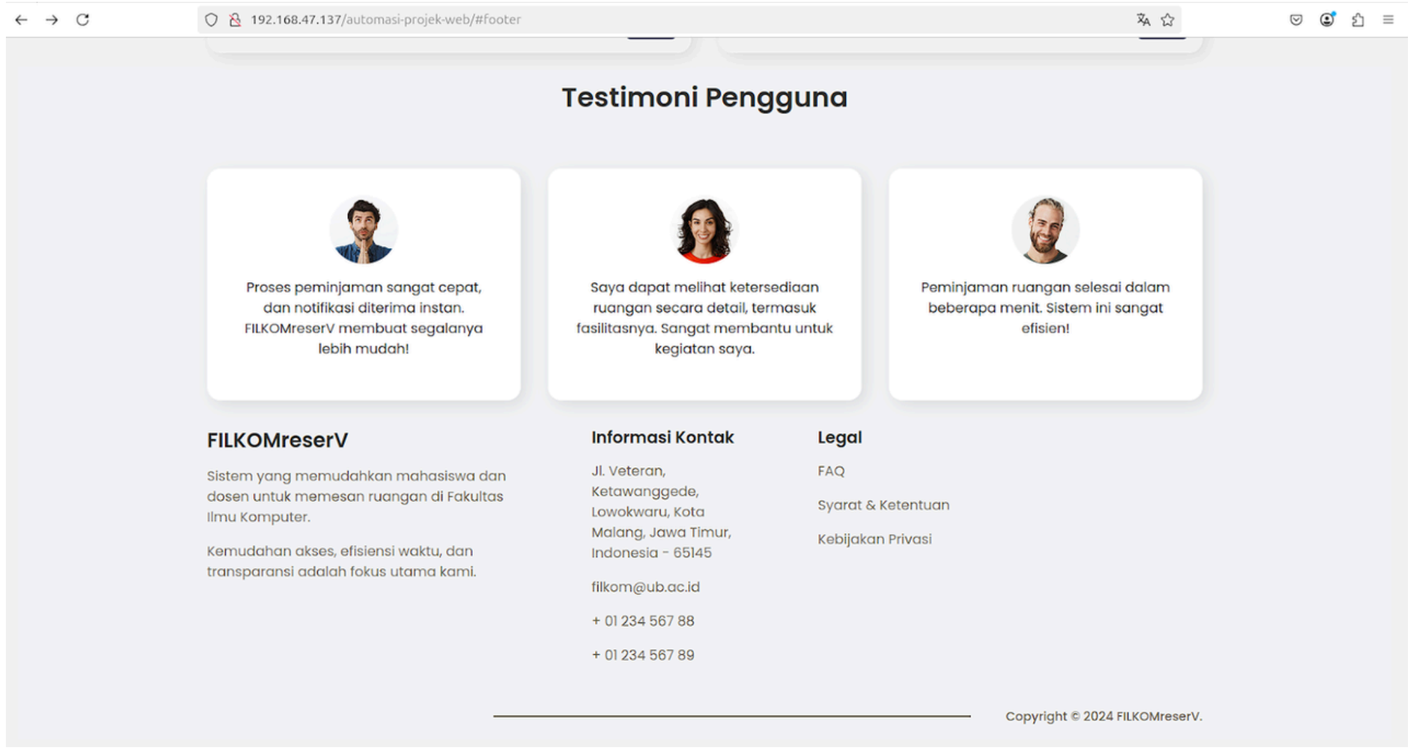
Langkah 4: Uji Coba Alur CI/CD

Sebelum
Kode awal github saat di clone di virtual machine

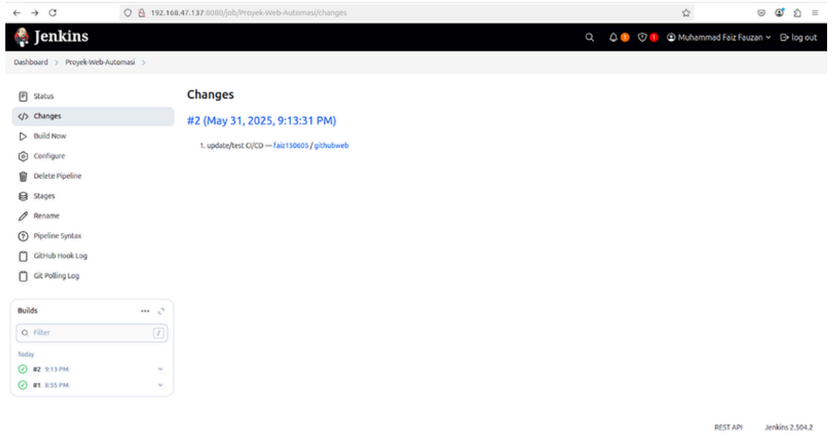


Langkah 4: Uji Coba Alur CI/CD

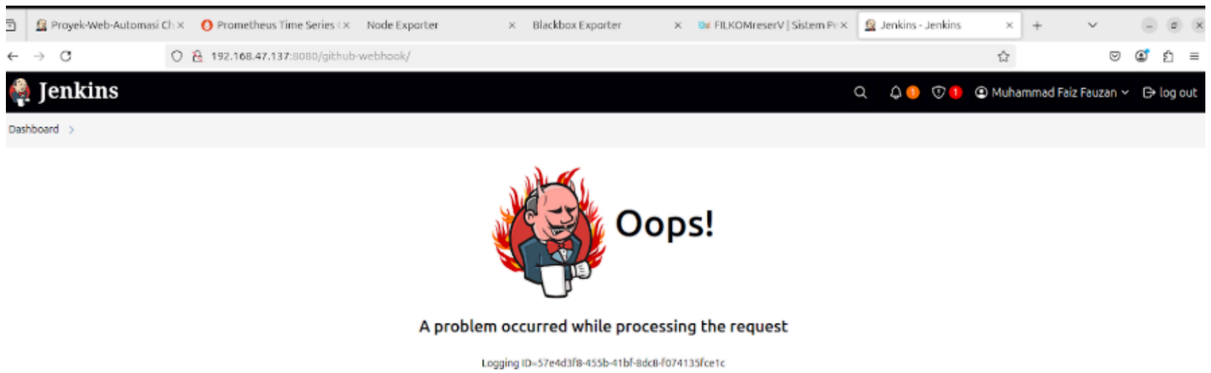
Sesudah
Kode diubah dan di push ke github



Status Jenkins



Masalah yang ditemui dan solusi yang dilakukan



Saya mencoba menambahkan webhook GitHub ke Jenkins dengan mengakses URL http://192.168.47.137:8080/github-webhook/ melalui browser. Namun, halaman memunculkan error “Oops!”. Setelah dicari tahu, Endpoint tersebut memang hanya menerima HTTP POST dari GitHub, bukan untuk akses manual melalui browser (HTTP GET).

Masalah yang ditemui dan solusi yang dilakukan

Namun, tantangan utama muncul karena Jenkins berjalan di VM lokal dengan IP 192.168.x.x, yang tidak dapat diakses langsung oleh server GitHub di internet. Tanpa konfigurasi jaringan tambahan seperti port forwarding atau layanan tunneling (misalnya ngrok), webhook tidak akan bisa mencapai Jenkins.

Untuk mengatasi ini, dengan memakai Poll SCM di Jenkins. Dengan metode ini, Jenkins secara berkala memeriksa repository GitHub apakah ada commit baru. Maka web utama akan mengalami perubahan otomatis, dengan diatur secara berkala 2 menit sekali.

Terima Kasih

kubernetes