

## **BAB IV**

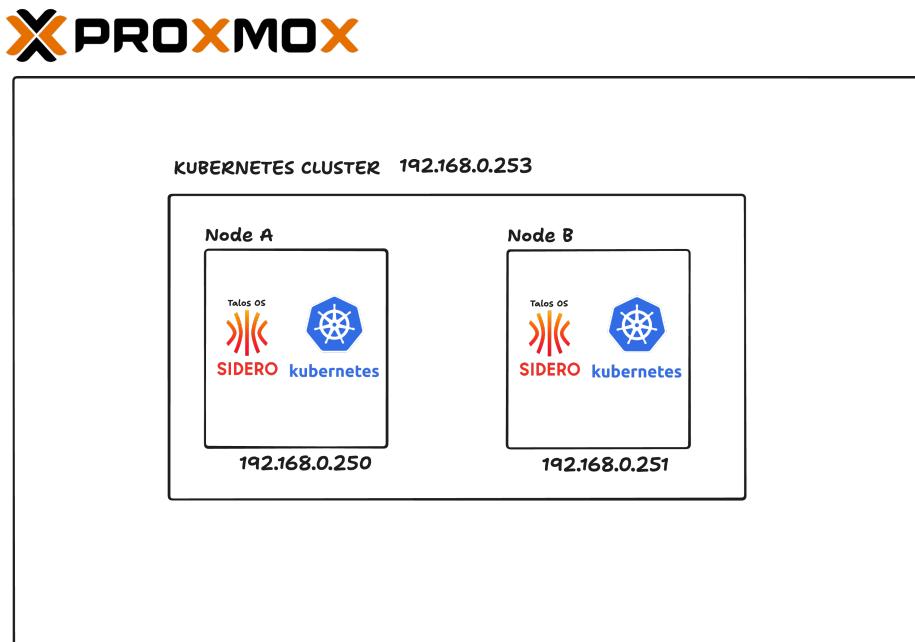
### **HASIL DAN PEMBAHASAN**

#### **4.1 Perancangan (Design)**

Pada tahap pertama ini peneliti akan melakukan perancangan sebuah infrastruktur dimana sistem Kubernetes dan ArgoCD akan dijalankan lalu microservice untuk dilakukan simulasi implementasi pada ArgoCD. Semua rancangan akan menggunakan visualisasi diagram secara garis besar (high-level) agar mudah dipahami.

##### **4.1.1 Sistem Arsitektur Infrastruktur**

Peneliti menggunakan Proxmox VE (Virtual Environment) yang didalamnya terdapat Virtual Machine berupa Talos OS Linux yang siap digunakan untuk menjalankan Kubernetes cluster

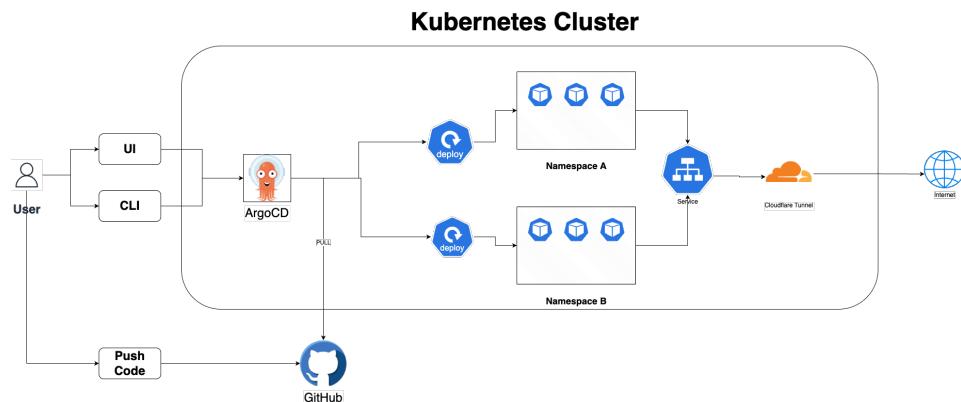


**Gambar 4.1** Arsitektur Infrastruktur High Level

##### **4.1.2 Sistem Arsitektur Kubernetes Cluster**

Pada bagian sebelumnya peneliti sudah memaparkan arsitektur infrastruktur dimana Kubernetes cluster akan berjalan. Pada tahap ini peneliti akan memaparkan

arsitektur pada Kubernetes cluster nya itu sendiri dimana ArgoCD akan berjalan.



**Gambar 4.2** Arsitektur Kubernetes High Level

Didalam sistem Kubernetes cluster yang dibuat terdapat komponen ArgoCD dan Cloudflare Tunnel (optional) yang bertujuan agar server microservice bisa diakses pada world wide web (internet).

## 4.2 Pengkodean (Coding) / Implementasi

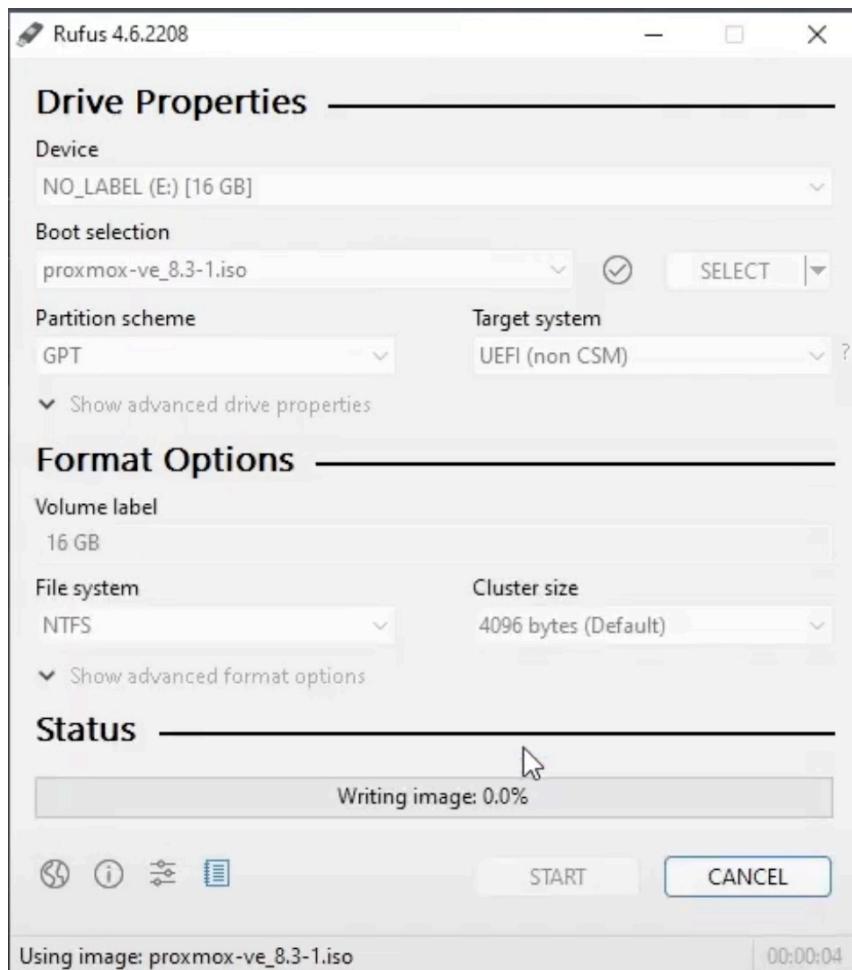
Tahap ini peneliti akan menjabarkan secara rinci pengkodean/implementasi rancangan sistem yang sudah dirancang. Peneliti akan melakukan implementasi rancangan sistem menggunakan beberapa komponen yaitu

1. Proxmox VE (versi 8.4)
2. Talos OS (versi 1.9.5)
3. Kubernetes
4. ArgoCD
5. Cloudflare Tunnel
6. Git repository (GitHub)

### 4.2.1 Implementasi Proxmox VE

Untuk implementasi Proxmox VE sendiri peneliti melakukan instalasi proxmox pada 2 mesin dengan arsitektur CPU X86. Pertama kita akan download ISO file atau installer proxmox yang terdapat di link ini [https://enterprise.proxmox.com/iso/proxmox-ve\\_8.4-1.iso](https://enterprise.proxmox.com/iso/proxmox-ve_8.4-1.iso). Setelah

itu kita memerlukan sebuah Flashdisk atau media untuk bootable ISO tersebut disini peneliti menggunakan tools yang bernama Rufus



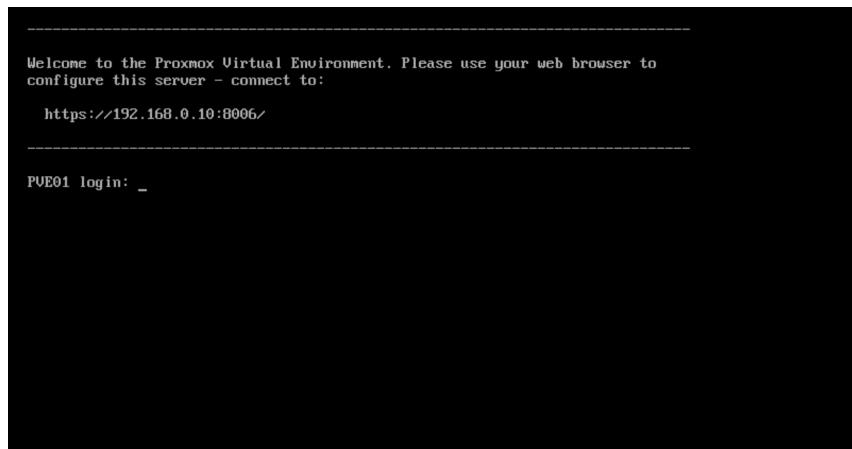
Gambar 4.3 Proses Pembuatan Bootable Proxmox Menggunakan Rufus

Setelah itu kita perlu mengganti bootable menu yang mengarah pada flashdisk atau media yang kita gunakan untuk instalasi proxmox ketika booting BIOS pada mesin yang digunakan. Akan terdapat layar instalasi Rufus seperti ini yang akan muncul pada mesin yang kita gunakan.



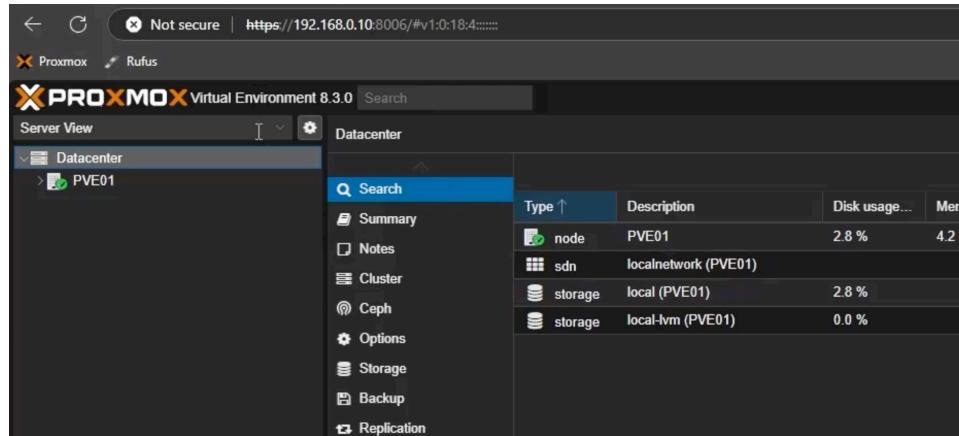
**Gambar 4.4** Tampilan Awal Instalasi Proxmox VE

Selanjutnya kita tinggal mengikuti apa yang diarahkan secara default oleh user interface yang ditampilkan hingga mesin akan restart dan berada pada tampilan seperti ini. Pada tampilan layar tersebut terdapat server yang bisa kita gunakan untuk akses user interface melalui browser



**Gambar 4.5** Tampilan Terminal Setelah Instalasi Proxmox Selesai

Akses UI pada komputer yang ada pada network yang sama dengan Proxmox melalui web user interface. Peneliti akan mengulang implementasi ini untuk mesin kedua.

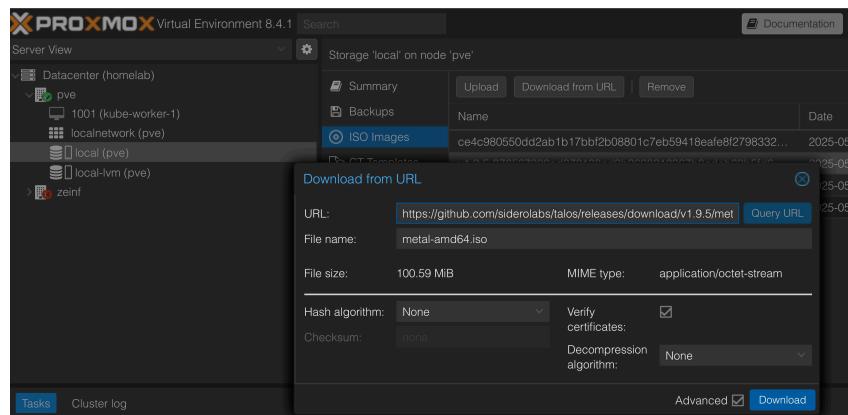


**Gambar 4.6** Tampilan Web Interface Proxmox VE

#### 4.2.2 Implementasi Talos OS

Pada tahap selanjutnya akan dilakukan instalasi Talos OS yang akan di-install menggunakan VM yang ada pada Proxmox VE. Pertama yang dilakukan adalah mengunduh ISO file Talos OS di sini.

<https://github.com/siderolabs/talos/releases/download/v1.9.5/metal-amd64.iso>



**Gambar 4.7** Proses Download ISO Talos OS

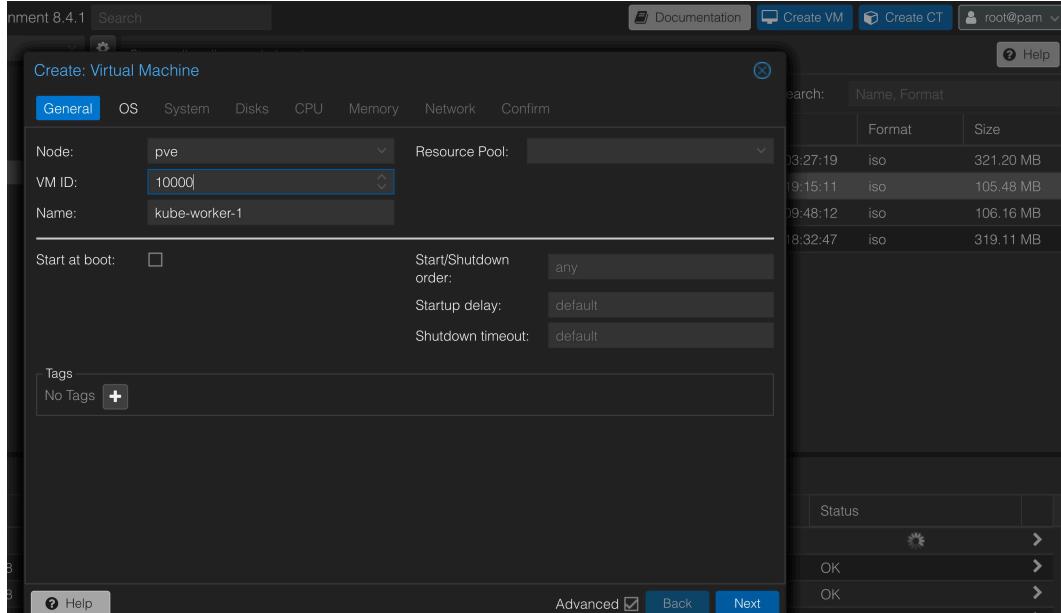
Url tersebut lalu didownload melalui proxmox agar tersimpan didalam proxmox. Lalu tahap selanjutnya adalah melakukan instalasi VM Talos OS pada proxmox.

##### 4.2.2.1 Instalasi Talos OS

Tahap ini adalah bagian instalasi VM Talos OS. Peneliti melakukan instalasi pada proxmox VE melalui interface web. Tahap ini akan dilakukan pada 2 mesin

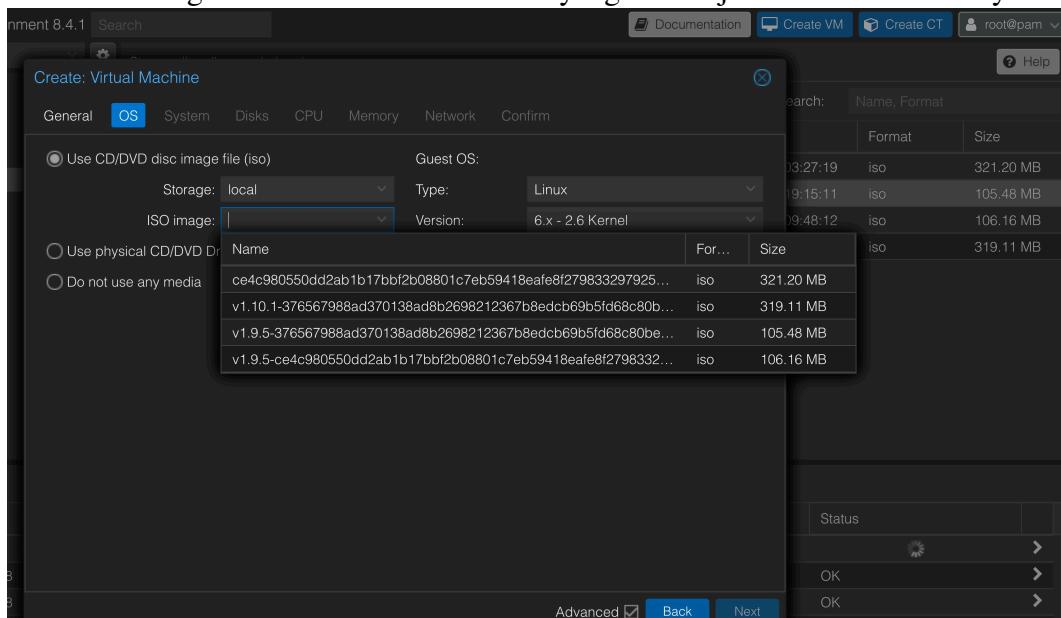
berbeda pada proxmox.

### 1. Klik Create VM lalu isikan VM ID dan Name untuk VM Talos OS



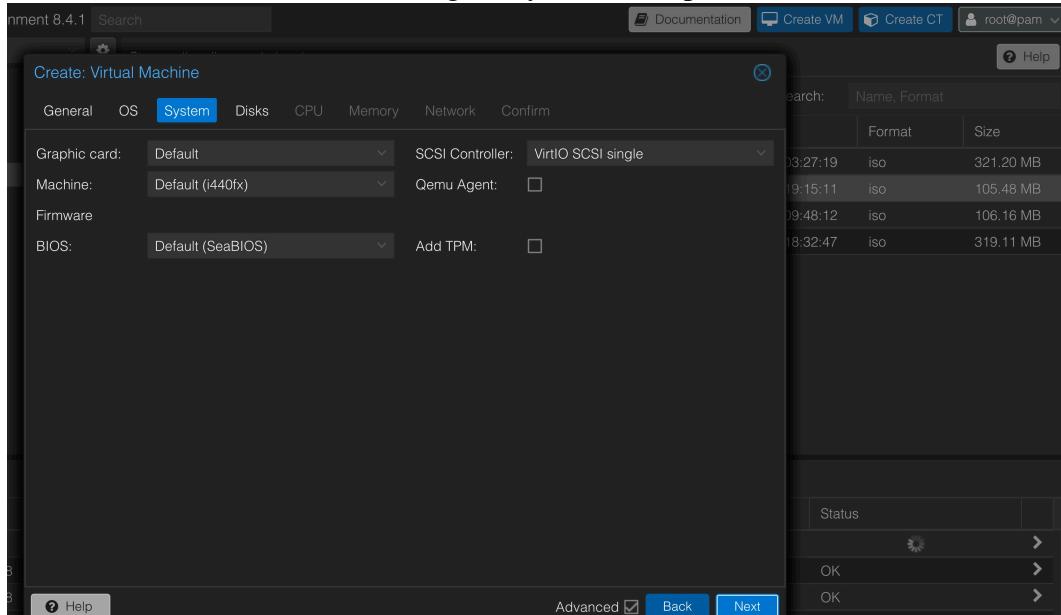
**Gambar 4.8** Instalasi Talos OS 1

### 2. Pada bagian OS. Pilih ISO Talos OS yang baru saja diunduh sebelum nya



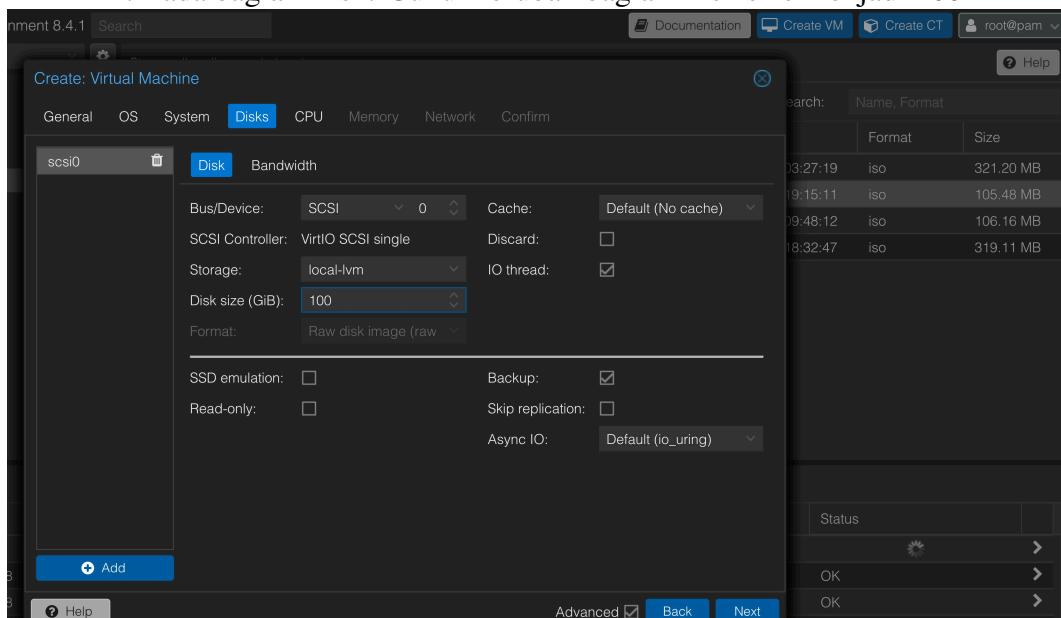
**Gambar 4.9** Instalasi Talos OS 2

### 3. Pada bagian System. cukup next



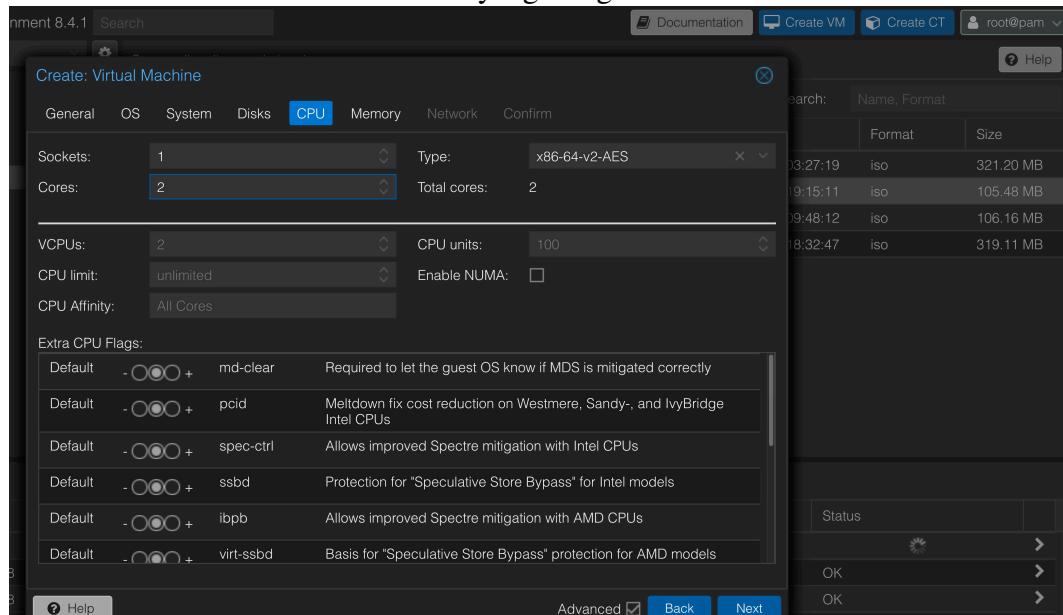
Gambar 4.10 Instalasi Talos OS 3

### 4. Pada bagian Disk. Cukup merubah bagian Disk size menjadi 100



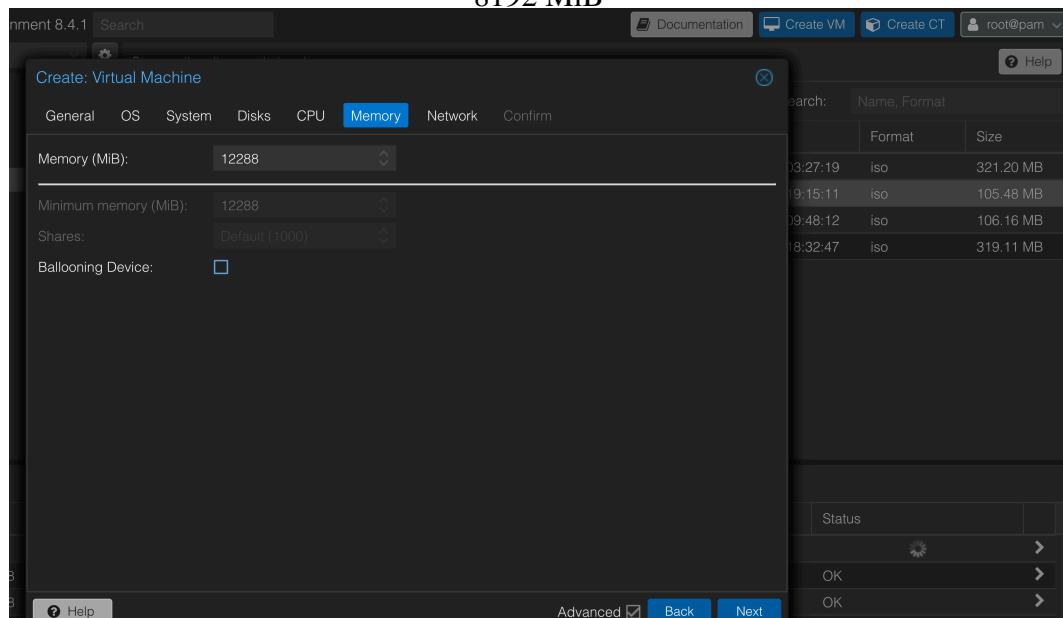
Gambar 4.11 Instalasi Talos OS 4

5. Pada bagian CPU. Cukup merubah cores menjadi 2 atau 4 sesuai spesifikasi mesin yang diinginkan



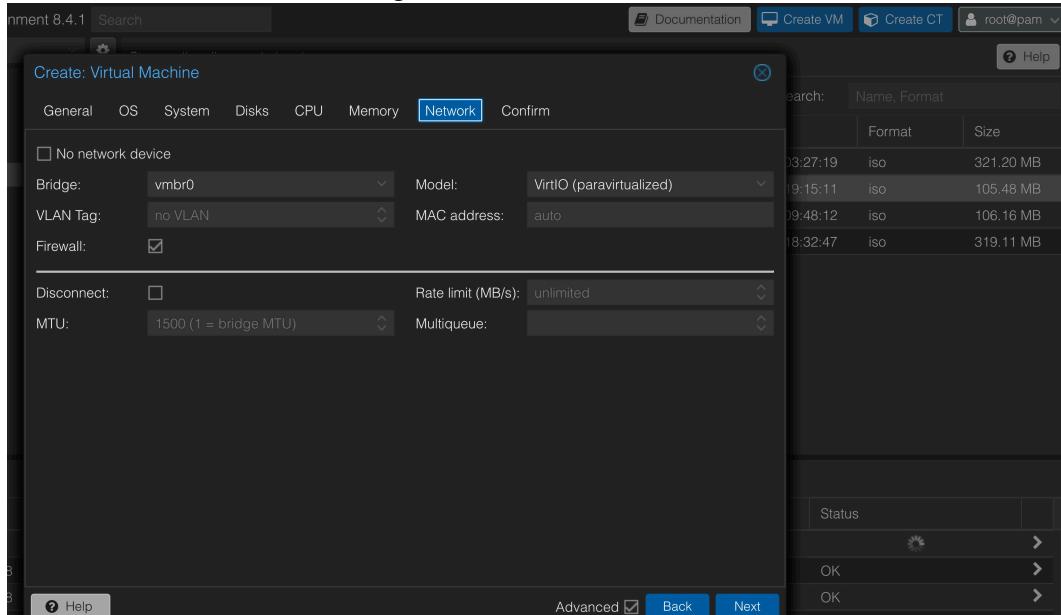
Gambar 4.12 Instalasi Talos OS 5

6. Pada bagian memory. Isikan memory sesuai yang dinginkan dengan minimal 8192 MiB



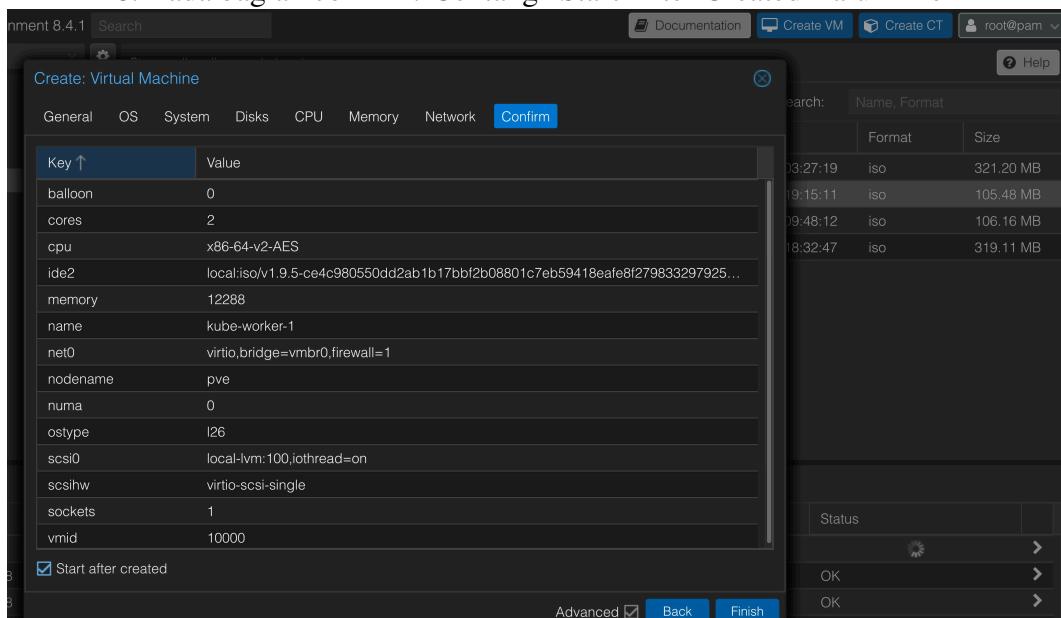
Gambar 4.13 Instalasi Talos OS 6

## 7. Pada bagian Network. Silahkan klik next



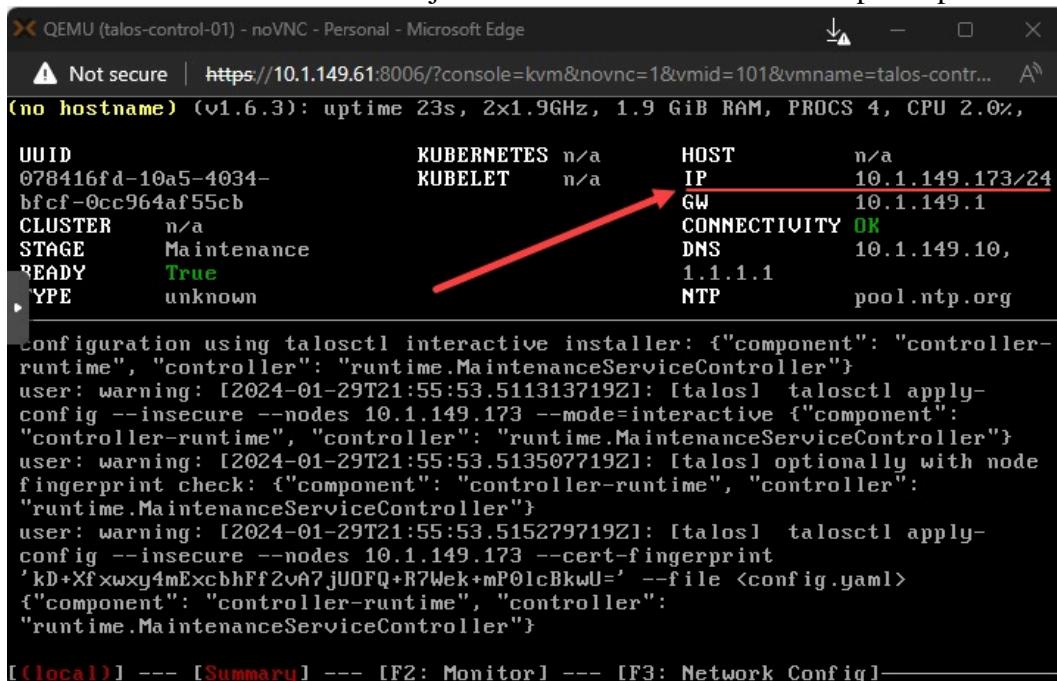
Gambar 4.14 Instalasi Talos OS 7

## 8. Pada bagian confirm. Centang "Start After Created" lalu Finish



Gambar 4.15 Instalasi Talos OS 8

10. Ketika Talos OS Sudah berjalan terminal Talos OS akan tampak seperti ini



The screenshot shows a terminal window with the following output:

KUBERNETES	HOST
n/a	n/a
KUBELET	IP 10.1.149.173/24
n/a	GW 10.1.149.1
Maintenance	CONNECTIVITY OK
True	DNS 10.1.149.10,
unknown	1.1.1.1
	NTP pool.ntp.org

Configuration using talosctl interactive installer: {"component": "controller-runtime", "controller": "runtime.MaintenanceServiceController"}  
user: warning: [2024-01-29T21:55:53.511313719Z]: [talos] talosctl apply-config --insecure --nodes 10.1.149.173 --mode=interactive {"component": "controller-runtime", "controller": "runtime.MaintenanceServiceController"}  
user: warning: [2024-01-29T21:55:53.513507719Z]: [talos] optionally with node fingerprint check: {"component": "controller-runtime", "controller": "runtime.MaintenanceServiceController"}  
user: warning: [2024-01-29T21:55:53.515279719Z]: [talos] talosctl apply-config --insecure --nodes 10.1.149.173 --cert-fingerprint 'kD+Xfxwxy4mExcjhFf2uA?jUOFQ+R?Wek+mP0lcBkwU=' --file <config.yaml>  
{"component": "controller-runtime", "controller": "runtime.MaintenanceServiceController"}  
[ (local) --- [Summary] --- [F2: Monitor] --- [F3: Network Config]

Gambar 4.16 Instalasi Talos OS 9

#### 4.2.3 Pengkodean Instalasi Kubernetes Script

Tahap ini adalah tahap untuk instalasi Kubernetes pada Talos OS. Disini peneliti akan menggunakan deklaratif yaml yang akan mengautomatisasi instalasi kubernetes pada Talos OS. Instalasi Kubernetes pada Talos OS sendiri mengikuti panduan dari pedoman yang terdapat pada website Talos OS <https://www.talos.dev/v1.9.5/introduction/getting-started/>

```

1 version: '3'
2 tasks:
3   talos:
4     desc: Bootstrap the Talos cluster
5     dir: '{{.TALOS_DIR}}'
6     cmd:
7       - '[ -f talsecret.sops.yaml ] || talhelper gensecret | sops
8         --filename-override talos/talsecret.sops.yaml --encrypt
9           /dev/stdin > talsecret.sops.yaml'
10      - talhelper genconfig
11      - talhelper gencommand apply --extra-flags="--insecure" |
12        bash
13      - until talhelper gencommand bootstrap | bash; do sleep 10;
14        done
15      - until talhelper gencommand kubeconfig
16        --extra-flags="{{.ROOT_DIR}} --force" | bash; do sleep
17          10; done

```

**Table 4.1** Konfigurasi script instalasi Kubernetes cluster pada Talos OS

Setelah dilakukan instalasi Kubernetes cluster pada Talos OS tampilan terminal akan menampilkan nama cluster dan juga sudah tidak menampilkan status Stage: Maintenance seperti Gambar 4.16.

```

kube-control-1 (v1.9.5): uptime 41m25s, 4x3.49GHz, 14 GiB RAM, PROCS 63, CPU
  UUID                                     TYPE      HOST
8aac4351-4534-49e3-
ab03-4042883e506b                         controlplane IP
CLUSTER          kubernetes (2
machines)                                KUBERNETES 192.168.0.200/24,
SIDEROLINK n/a                            KUBELET    192.168.0.250/32
                                         ✓ Healthy   GW          192.168.0.1
                                         ✓ Healthy   CONNECTIVITY ✓ OK
  Logs
"runtime.MachineStatusController"
kern:  info: [2025-05-15T19:58:23.891235046Z]: eth0: renamed from tmpf194c
kern:  info: [2025-05-15T19:58:23.935322046Z]: eth0: renamed from tmpcbea6
kern:  info: [2025-05-15T19:58:23.936710046Z]: eth0: renamed from tmpd7271
kern:  info: [2025-05-15T19:58:23.963162046Z]: eth0: renamed from tmpd3bd
kern:  info: [2025-05-15T19:58:24.023889046Z]: eth0: renamed from tmpbaa39
kern:  info: [2025-05-15T19:58:24.091156046Z]: eth0: renamed from tmp6fd81
kern:  info: [2025-05-15T19:58:24.096220046Z]: eth0: renamed from tmpa93f1
kern:  info: [2025-05-15T19:58:24.119990046Z]: eth0: renamed from tmp5e2b7
kern:  info: [2025-05-15T19:58:24.120383046Z]: eth0: renamed from tmp17793
kern:  info: [2025-05-15T19:58:24.151169046Z]: eth0: renamed from tmp77f0e
kern:  info: [2025-05-15T19:58:24.153420046Z]: eth0: renamed from tmp76b39
kern:  info: [2025-05-15T19:58:49.178916046Z]: eth0: renamed from tmp65ac9
kern:  info: [2025-05-15T20:01:42.044436046Z]: eth0: renamed from tmpbc3b8

```

**Gambar 4.17** Tampilan Talos OS setelah instalasi kubernetes cluster

```

1  clusterName: kubernetes
2  endpoint: https://192.168.0.250:6443
3  ...
4  nodes:
5    - hostname: "kube-cp-1"
6      ipAddress: "192.168.0.200"
7      installDisk: "/dev/sda"
8      machineSpec:
9        controlPlane: true
10       networkInterfaces:
11         - deviceSelector:
12             hardwareAddr: "de:ad:be:ef:00:01"
13             addresses:
14               - "192.168.0.200/24"
15             routes:
16               - network: "0.0.0.0/0"
17                 gateway: "192.168.0.1"
18             mtu: 1500
19             vip:
20               ip: "192.168.0.250"
21    - hostname: "kube-worker-1"
22      ipAddress: "192.168.0.201"
23      installDisk: "/dev/sda"
24      machineSpec:
25        controlPlane: false
26      networkInterfaces:
27        - deviceSelector:
28            hardwareAddr: "de:ad:be:ef:00:02"
29            dhcp: false
30            addresses:
31              - "192.168.0.201/24"
32            routes:
33              - network: "0.0.0.0/0"
34                gateway: "192.168.0.1"
35             mtu: 1500

```

**Table 4.2** Konfigurasi script instalasi Kubernetes cluster pada Talos OS

Setelah instalasi kubernetes pada Talos OS dilakukan maka kita dapat mengakses kubernetes cluster tersebut menggunakan cli kubectl. Sebagai contoh saya mempunyai aplikasi yang menampilkan informasi header pada browser pada echo.zeinfahrozi.my.id

```
{
  "path": "/",
  "headers": {
    "host": "echo.zeinfahrozi.my.id",
    "x-request-id": "35226cc0237d8725c530dde2ebd3edf2",
    "x-real-ip": "2a09:bac1:3480:50::da:ea",
    "x-forwarded-for": "2a09:bac1:3480:50::da:ea",
    "x-forwarded-host": "echo.zeinfahrozi.my.id",
    "x-forwarded-port": "443",
    "x-forwarded-proto": "https",
    "x-scheme": "https",
    "x-scheme": "https",
    "x-original-forwarded-for": "2a09:bac1:3480:50::da:ea",
    "user-agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36",
    "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7",
    "accept-encoding": "gzip, br",
    "accept-language": "en-US,en-GB;q=0.9,en;q=0.8,id-ID;q=0.7,id;q=0.6",
    "cdn-loop": "cloudflare; loops=1",
    "cf-connecting-ip": "2a09:bac1:3480:50::da:ea",
    "cf-ipcountry": "ID",
    "cf-ray": "9405a3071edce782-CGK",
    "cf-visitor": "{\"scheme\":\"https\"}",
    "cf-warp-tag-id": "b3524fd1-ce55-48f0-a8b3-1811cfef6a7",
    "if-none-match": "W/\"732-XUT4cXocJ3eOnLDMkzmIkzg6y6k\"",
    "priority": "u0,i",
    "sec-ch-ua": "\"Chromium\";v=\"136\", \"Google Chrome\";v=\"136\", \"Not.A/Brand\";v=\"99\"",
    "sec-ch-ua-mobile": "?0",
    "sec-ch-ua-platform": "\"macOS\"",
    "sec-fetch-dst": "document",
    "sec-fetch-mode": "navigate",
    "sec-fetch-site": "none",
    "sec-fetch-user": "?1",
    "upgrade-insecure-requests": "1"
  },
  "method": "GET",
  "body": "",
  "fresh": false,
  "hostname": "echo.zeinfahrozi.my.id",
  "ip": "2a09:bac1:3480:50::da:ea",
  "ips": [
    "2a09:bac1:3480:50::da:ea"
  ]
}
```

**Gambar 4.18** Tampilan echo.zeinfahrozi.my.id pada browser

### 4.3 Implementasi ArgoCD pada cluster kubernetes

Tahap ini merupakan instalasi instance ArgoCD itu sendiri pada kubernetes cluster yang sudah dibuat sebelumnya. ArgoCD dipasang menggunakan Helm chart dengan konfigurasi kustom yang disesuaikan dengan kebutuhan lingkungan produksi.

Berikut adalah contoh perintah untuk menginstal ArgoCD menggunakan Helm:

```

1 # Menambahkan repo ArgoCD
2 helm repo add argo https://argoproj.github.io/argo-helm
3 helm repo update

4

5 # Membuat namespace untuk ArgoCD
6 kubectl create namespace argocd

7

8 # Menginstal ArgoCD dengan Helm
9 helm upgrade --install argocd argo/argo-cd \
10   --namespace argocd \
11   --values values.sops.yaml \

```

```
12 --wait
```

#### 4.3.1 Instalasi ArgoCD

ArgoCD diinstal menggunakan Helm dengan nilai-nilai kustom yang didefinisikan dalam file konfigurasi. Berikut adalah contoh konfigurasi `values.sops.yaml` yang digunakan:

```
1 # values.sops.yaml
2 crds:
3   install: true
4
5 global:
6   domain: argo.zeinfahrozi.my.id
7
8 configs:
9   params:
10     server.insecure: true
11 cm:
12   statusbadge.enabled: true
13   kustomize.buildOptions: --enable-alpha-plugins --enable-exec
14   helm.valuesFileSchemes: >-
15     secrets+gpg-import,secrets+gpg-import-kubernetes,
16     secrets+age-import,secrets+age-import-kubernetes,
17     secrets,secrets+literal,https
18   resource.exclusions: |
19     - apiGroups:
20       - cilium.io
21     kinds:
22       - CiliumIdentity
23   clusters:
24     - "*"
```

Beberapa konfigurasi penting yang diterapkan:

- ArgoCD diakses melalui domain `argo.zeinfahrozi.my.id`
- Mode `insecure` diaktifkan untuk pengembangan
- Fitur status badge diaktifkan untuk memantau status aplikasi
- Dukungan untuk multiple value files dengan skema yang berbeda
- Eksklusi sumber daya tertentu seperti `CiliumIdentity` dari manajemen ArgoCD

#### **4.3.2 Konfigurasi High Availability**

Untuk memastikan ketersediaan tinggi, komponen-komponen kritis ArgoCD dikonfigurasi dengan multiple replica:

- ArgoCD Server: 2 replica
- ArgoCD Controller: 2 replica
- Dex Server: 2 replica

#### **4.3.3 Integrasi dengan Monitoring**

ArgoCD terintegrasi dengan stack monitoring yang sudah ada di cluster melalui ServiceMonitor untuk memantau metrik dari komponen-komponennya:

- ArgoCD Server metrics
- ArgoCD Controller metrics
- Dex Server metrics
- Redis metrics

#### **4.3.4 Manajemen Aplikasi**

Setelah ArgoCD terinstal, aplikasi-aplikasi dapat dikelola menggunakan GitOps. Setiap aplikasi didefinisikan sebagai kustom resource Kubernetes yang mereferensikan repositori Git yang berisi manifest Kubernetes. ArgoCD akan secara otomatis melakukan sinkronisasi antara status yang diinginkan (yang didefinisikan di Git) dengan status aktual di cluster.

#### **4.3.5 Keamanan**

Beberapa aspek keamanan yang diterapkan pada instalasi ArgoCD ini antara lain:

- Penggunaan HTTPS untuk akses web UI
- Integrasi dengan Dex untuk autentikasi
- Pembatasan akses berbasis peran (RBAC)
- Penyimpanan rahasia yang aman menggunakan SOPS

Dengan konfigurasi ini, ArgoCD siap digunakan untuk mengelola aplikasi secara deklaratif menggunakan prinsip GitOps, di mana semua perubahan konfigurasi dilakukan melalui pull request dan version control system.

#### 4.4 Implementasi Cloudflare Tunnel

Cloudflare Tunnel digunakan untuk mengekspos layanan dalam cluster ke internet dengan aman tanpa perlu membuka port firewall. Berikut adalah langkah-langkah implementasinya:

1. \*\*Persiapan\*\* - Memiliki akun Cloudflare - Mendaftarkan domain yang akan digunakan - Mengatur DNS di Cloudflare
2. \*\*Instalasi Cloudflare Tunnel\*\* Cloudflare Tunnel diinstal menggunakan ArgoCD dengan konfigurasi sebagai berikut:

```
1 # cloudflaered.yaml
2 apiVersion: argoproj.io/v1alpha1
3 kind: Application
4 metadata:
5   name: cloudflaered
6   namespace: argo-system
7 spec:
8   project: kubernetes
9   sources:
10    - repoURL: "https://github.com/mozarik/zein-home-lab.git"
11      path: kubernetes/apps/network/cloudflaered
12      targetRevision: main
13   destination:
14     name: in-cluster
15     namespace: network
16   syncPolicy:
17     automated:
18       prune: true
19       selfHeal: true
```

3. \*\*Konfigurasi Tunnel\*\* Setelah terinstal, Cloudflare Tunnel perlu dikonfigurasi untuk meneruskan lalu lintas ke layanan dalam cluster. Berikut contoh konfigurasi untuk mengekspos ArgoCD:

```
1 # config.yaml
2 tunnel: <tunnel-id>
3 credentials-file: /etc/cloudflaered/credentials.json
4 ingress:
```

```
5     - hostname: argo.zeinfahrozi.my.id
6       service: http://argocd-server.argocd.svc.cluster.local:80
7     - service: http_status:404
```

#### 4.4.1 Konfigurasi Git Repository

Git repository digunakan sebagai sumber kebenaran (source of truth) untuk konfigurasi infrastruktur. Repository yang digunakan adalah <https://github.com/mozarik/zein-home-lab> dengan struktur sebagai berikut:

Alur kerja GitOps yang diterapkan: 1. Perubahan konfigurasi dilakukan melalui pull request 2. Setelah pull request disetujui dan digabungkan ke branch main 3. ArgoCD secara otomatis mendeteksi perubahan dan melakukan sinkronisasi dengan cluster

#### 4.4.2 Integrasi dengan GitHub Actions

Untuk memastikan kualitas kode dan keamanan, diterapkan GitHub Actions workflow yang akan: 1. Melakukan linting pada file konfigurasi Kubernetes 2. Melakukan validasi dengan kubeval 3. Melakukan pengecekan keamanan dengan kubesec

Contoh workflow GitHub Actions:

```
1 name: Lint and Validate
2
3 on:
4   push:
5     branches: [ main ]
6   pull_request:
7     branches: [ main ]
8
9 jobs:
10  lint-validate:
11    runs-on: ubuntu-latest
12    steps:
13      - uses: actions/checkout@v3
14
15      - name: Lint Kubernetes files
16        uses: azure/k8s-manifests-base@v1
17        with:
18          action: lint
19          files: '**/*.yaml'
20
```

```
21      - name: Validate Kubernetes files
22        uses: azure/k8s-manifests-base@v1
23        with:
24          action: validate
25          files: '**/*.yaml'
```

Dengan konfigurasi ini, seluruh perubahan infrastruktur dapat dilacak melalui riwayat Git, dan proses deployment menjadi lebih terotomatisasi dan konsisten.

## 4.5 Testing

Setelah menyelesaikan tahap instalasi dan konfigurasi infrastruktur, langkah selanjutnya adalah melakukan pengujian untuk memastikan seluruh komponen berfungsi seperti yang diharapkan. Pengujian ini mencakup beberapa aspek penting termasuk fungsionalitas Kubernetes cluster, integrasi ArgoCD, serta alur kerja GitOps yang telah diterapkan. Melalui pengujian menyeluruh ini, diharapkan dapat dievaluasi sejauh mana solusi yang dibangun mampu memenuhi kebutuhan pengembangan dan operasional aplikasi secara efisien.

### 4.5.1 Pengujian (Testing)

Pengujian black-box testing yang dilakukan terhadap sistem pada tahap ini menggunakan metode validasi (validation). Metode validasi yang digunakan dalam melakukan pengujian ini berfungsi untuk mengetahui valid atau tidaknya sebuah fungsi dari sistem yang dibangun. Melakukan pengujian black-box dengan metode validasi ini juga menentukan apakah sistem telah sesuai seperti apa yang diinginkan oleh stakeholder pada tahap perencanaan. Pengujian black-box ini dilakukan dengan jumlah test case sebanyak 15 (lima belas) yang mencakup berbagai aspek fungsionalitas ArgoCD.

### 4.5.2 Metodologi Pengujian

Pengujian dilakukan dengan pendekatan black-box testing yang berfokus pada fungsionalitas sistem tanpa memperhatikan struktur internal kode. Setiap test case dirancang untuk memverifikasi fitur-fitur kunci dari ArgoCD dalam mendukung alur kerja GitOps.

### 4.5.3 Hasil Pengujian

Berikut adalah daftar test case yang telah dilakukan beserta hasilnya:

**Table 4.3** Daftar Test Case Black-Box Testing

Kode Uji	Nama Uji	Kasus Uji	Hasil Yang Diharapkan	Status
BT-001	Login ke Dashboard ArgoCD	<ol style="list-style-type: none"> <li>Buka halaman login ArgoCD</li> <li>Masukkan kredensial admin</li> <li>Klik tombol login</li> </ol>	Pengguna berhasil login dan diarahkan ke dashboard utama	Valid
BT-002	Tambah Aplikasi Baru	<ol style="list-style-type: none"> <li>Klik "New App"</li> <li>Isi form dengan detail aplikasi</li> <li>Klik "Create"</li> </ol>	Aplikasi baru berhasil dibuat dan muncul di daftar aplikasi	Valid
BT-003	Sinkronisasi Otomatis	<ol style="list-style-type: none"> <li>Buat perubahan pada file konfigurasi di repo Git</li> <li>Push perubahan ke branch yang dimonitor</li> </ol>	ArgoCD mendeteksi perubahan dan melakukan sinkronisasi otomatis	Valid
BT-004	Rollback Aplikasi	<ol style="list-style-type: none"> <li>Pilih aplikasi</li> <li>Klik "History and Rollback"</li> <li>Pilih versi sebelumnya</li> <li>Klik "Sync"</li> </ol>	Aplikasi berhasil di-rollback ke versi sebelumnya	Valid
BT-005	Validasi Status Kesehatan	<ol style="list-style-type: none"> <li>Deploy aplikasi dengan konfigurasi salah</li> <li>Periksa status di dashboard</li> </ol>	Menampilkan status "Degraded" atau "Error" dengan pesan yang jelas	Valid
BT-006	Pencarian Aplikasi	<ol style="list-style-type: none"> <li>Gunakan fitur search di dashboard</li> <li>Masukkan nama aplikasi</li> </ol>	Menampilkan aplikasi yang sesuai dengan kata kunci pencarian	Valid
BT-007	Filter Aplikasi	<ol style="list-style-type: none"> <li>Gunakan filter berdasarkan status/kategori</li> <li>Pilih filter tertentu</li> </ol>	Menampilkan aplikasi yang sesuai dengan filter yang dipilih	Valid

**Table 4.4** Daftar Test Case Black-Box Testing (Lanjutan)

Kode Uji	Nama Uji	Kasus Uji	Hasil Yang Diharapkan	Status
BT-008	Manajemen Kluster	1. Tambah kluster baru 2. Verifikasi koneksi	Kluster baru terdaftar dan terhubung dengan status "Healthy"	Valid
BT-009	Logout	1. Klik profil pengguna 2. Pilih "Logout"	Pengguna berhasil logout dan diarahkan ke halaman login	Valid
BT-010	Responsivitas UI	1. Akses dashboard dari berbagai perangkat (desktop, tablet, mobile)	Tampilan UI menyesuaikan dengan ukuran layar	Valid
BT-011	Notifikasi Sinkronisasi	1. Lakukan sinkronisasi manual 2. Periksa notifikasi	Muncul notifikasi yang menampilkan status sinkronisasi	Valid
BT-012	Error Handling	1. Masukkan URL repo Git yang tidak valid 2. Coba buat aplikasi	Menampilkan pesan error yang jelas tentang URL yang tidak valid	Valid
BT-013	Eksport Konfigurasi	1. Pilih aplikasi 2. Eksport konfigurasi	File konfigurasi berhasil diunduh dalam format YAML	Valid
BT-014	Manajemen Izin	1. Buat pengguna dengan role terbatas 2. Verifikasi akses	Pengguna hanya dapat mengakses fitur sesuai role yang diberikan	Valid
BT-015	Audit Log	1. Lakukan beberapa aksi di dashboard 2. Periksa halaman audit log	Semua aksi terekam dalam log dengan timestamp dan detail yang jelas	Valid

#### 4.5.4 Analisis Hasil Pengujian

Berdasarkan hasil pengujian yang telah dilakukan, dapat dianalisis bahwa implementasi ArgoCD berhasil memenuhi kebutuhan dalam mendukung alur kerja

GitOps pada infrastruktur Kubernetes. Berikut adalah analisis mendalam dari hasil pengujian:

#### **4.5.4.1 Ketahanan Sistem**

Sistem berhasil melewati semua skenario pengujian yang mencakup berbagai aspek fungsionalitas ArgoCD. Hal ini menunjukkan bahwa arsitektur yang dirancang telah memenuhi kebutuhan dasar dalam implementasi GitOps.

#### **4.5.4.2 Kesesuaian dengan Ekspektasi**

Dari 15 test case yang dilakukan, seluruhnya menunjukkan hasil yang sesuai dengan ekspektasi. Ini menunjukkan bahwa ArgoCD dapat diandalkan untuk mengelola aplikasi pada cluster Kubernetes dengan pendekatan GitOps.

#### **4.5.4.3 Keandalan Fitur Inti**

Fitur-fitur inti seperti sinkronisasi otomatis, rollback, dan manajemen konfigurasi berfungsi dengan baik. Hal ini menjadi bukti bahwa ArgoCD dapat diandalkan untuk keperluan continuous deployment dalam lingkungan produksi.

#### **4.5.4.4 Keterbatasan**

Meskipun semua test case berhasil, terdapat beberapa aspek yang memerlukan perhatian lebih lanjut, seperti manajemen kredensial yang aman dan pengaturan RBAC yang lebih ketat untuk keperluan produksi.

#### **4.5.4.5 Rekomendasi**

Berdasarkan hasil pengujian, berikut beberapa rekomendasi untuk pengembangan selanjutnya:

1. Implementasi mekanisme backup dan recovery yang lebih komprehensif
2. Peningkatan pengujian keamanan dan penetrasi
3. Pengembangan pipeline CI/CD yang lebih matang
4. Implementasi monitoring dan alerting yang lebih baik