Perancangan Dan Implementasi Container Orchestration Cluster Pada Pengembangan Produk Coofis Verse Di Pt Arm Solusi

1st Naufal Azriel Zheinuary
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
naufalzheinuary@student.telkomuniversity.ac.id

2nd Muhammad Iqbal
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
miqbal@telkomuniversity.ac.id

3rd Bramandityo Prabowo
PT Andal Rancang Multi Solusi
Bandung, Indonesia
bram3@armsolusi.com

***Abstrak* — PT Andal Rancang Multi Solusi merupakan sebuah perusahaan IT *Consultant* yang sedang mengembangkan salah satu produknya yaitu Coofis NDE menjadi Coofis Verse dengan menggunakan arsitektur *microservice* dan teknologi *container*. Coofis NDE adalah salah satu produk yang dibuat untuk mengelola tata persuratan secara elektronik. Dalam upaya pengembangan ini PT ARM Solusi menggunakan Kubernetes sebagai platform *container orchestration* untuk mengotomatisasi penyebaran dan pengelolaan *container*, menggunakan Ansible untuk otomatisasi *deployment* dan *operations* serta menggunakan Docker Hub sebagai *image registry*. Pengembangan Coofis Verse ini melibatkan beberapa *tools* pendukung seperti MinIO, MongoDB, dan Keycloak yang akan dikelola dan di*monitoring* dengan menggunakan *software* Lens Kubernetes IDE. *Tools* pendukung yang berada dalam *container* ini dapat diakses melalui *browser* dengan waktu akses rata-rata 3-5 detik. Serta waktu yang dibutuhkan untuk proses *scaling pods* sekitar 10-15 detik.**

Kata Kunci - PT Andal Rancang Multi Solusi, IT *Consultant*, Coofis NDE, *Coofis Verse, Container Orchestration, Kubernetes*

I. PENDAHULUAN

A. Latar Belakang

Pada zaman sekarang telah banyak aspek kehidupan telah berubah karena kemajuan teknologi informasi dan komunikasi, termasuk administrasi dan tata kelola pemerintahan. [1]. Salah satu aplikasi teknologi informasi dalam administrasi publik adalah sistem tata persuratan berbasis web yang terus berkembang untuk meningkatkan efisiensi pengelolaan surat menyurat di berbagai institusi. Tata persuratan biasanya mencakup proses pembuatan, pengelolaan, penyimpanan, dan pengarsipan surat. Salah satu perusahaan yang mengembangkan aplikasi tata persuratan web adalah PT Andal Rancang Multi Solusi.

PT Andal Rancang Multi Solusi adalah salah satu perusahaan yang bergerak di bidang pengembangan teknologi, seperti big data, analitik data, kolaborasi dan otomasi administrasi, serta integrasi aplikasi dan API [2]. Salah satu produk yang dikembangkan yaitu Coofis NDE (*Collaboration Office* Nota Dinas Elektronik) yang dirancang untuk mengotomasikan proses administrasi perusahaan secara *paperless*, mulai dari pembuatan hingga disposisi surat. Meskipun demikian, sistem monolitik yang digunakan pada Coofis NDE dianggap kurang efisien [2], sehingga dilakukan pengembangan menjadi Coofis Verse dengan memanfaatkan arsitektur *microservice* dan teknologi *containerization*.

Containerization adalah metode yang mengisolasi aplikasi dalam lingkungan yang berbeda untuk meningkatkan efisiensi dan skalabilitas, serta memungkinkan aplikasi dijalankan secara konsisten di berbagai lingkungan [3]. Untuk mendukung proses pengembangan, digunakan teknik *container orchestration cluster* dengan menggunakan platform Kubernetes, yang akan mengotomatisasi manajemen siklus hidup *container*, termasuk alokasi sumber daya, penyebaran, penskalaan otomatis, pemantauan kesehatan, migrasi, keseimbangan beban, keamanan, dan konfigurasi jaringan [4].

Dalam proyek ini, fokus utama adalah merancang dan mengimplementasikan *container orchestration cluster* sebagai fondasi infrastruktur yang kuat bagi Coofis Verse. Hal ini bertujuan untuk mengelola dan menyediakan aplikasi dengan lebih efisien dan andal, meningkatkan waktu pemasaran produk, mengurangi biaya infrastruktur, serta meningkatkan kepuasan pelanggan dengan layanan yang lebih responsif dan dapat diandalkan.

B. Rumusan Masalah

Berikut adalah rumusan masalah dari Proyek Akhir ini:

1. Bagaimana cara merancang dan mengimplementasikan sebuah *container orchestration cluster* yang dapat mengelola infrastruktur dengan efisien dan efektif untuk

mendukung pengembangan produk Coofis Verse?
2. Bagaimana cara meningkatkan skalabilitas dan kinerja produk Coofis Verse dengan menggunakan *container orchestration cluster*?

C. Tujuan

Berikut adalah tujuan dari Proyek Akhir ini:

1. Merancang arsitektur *container orchestration cluster* yang sesuai dengan kebutuhan produk Coofis Verse dan infrastruktur PT ARM Solusi.
2. Mengimplementasikan *container orchestration cluster* dengan menggunakan platform *container* terbaik dalam *containerization* dan *orchestration*.
3. Menyediakan otomatisasi yang efisien untuk *container* aplikasi Coofis Verse.

D. Manfaat

Berikut adalah manfaat dari Proyek Akhir ini:

1. Dapat dengan cepat membangun dan mengembangkan produk Coofis Verse dengan lebih efisien dan konsisten.
2. Dapat mengurangi beban kerja administratif dan mempercepat siklus pengembangan dan penyebaran, sehingga menghemat waktu dan biaya operasional

II. KAJIAN TEORI

A. Microservice

Microservice adalah desain infrastruktur yang membagi aplikasi menjadi berbagai layanan kecil yang saling terhubung, berbeda dari aplikasi monolitik yang besar dan terintegrasi. Arsitektur *microservice* bertujuan untuk meningkatkan fleksibilitas, skalabilitas, dan efisiensi dalam pengembangan perangkat lunak dengan mengelola layanan secara terpisah namun terkoordinasi. Sistem ini dirancang untuk beroperasi secara terdistribusi, memungkinkan setiap layanan untuk dikelola dan di-scale secara independen [5].

Jika terjadi masalah pada salah satu layanan, perbaikannya hanya mempengaruhi layanan tersebut tanpa memengaruhi layanan lainnya, sehingga meningkatkan produktivitas pengembangan dan pemeliharaan aplikasi.

B. Container Orchestration

Sebagai alternatif dari virtualisasi berbasis *hypervisor*, *container* adalah metode virtualisasi sistem operasi yang ringan, memungkinkan aplikasi beserta semua dependensinya, seperti kode dan konfigurasi, untuk dikemas dan dijalankan dalam lingkungan *cloud* [6]. *Container Orchestration* adalah proses otomatisasi penyebaran *container* di *cluster* yang terdiri dari beberapa *node*.

Orchestration memungkinkan pengguna untuk menyebarkan aplikasi ke seluruh *cluster* dan mengelola beban kerja *multi-container* yang kompleks. Dalam proyek ini, dibuat beberapa *container* seperti *backend, frontend, notes, portal, dan reimburse*.

C. Kubernetes

Kubernetes adalah sistem *container orchestration* yang memastikan pengelolaan container secara efektif dalam memproses workload di mesin fisik atau virtual [7] dan dikembangkan oleh Google serta didukung *Cloud Native Computing Foundation (CNCF)*. Kubernetes merupakan sistem berisfat *open-source* dan dirancang untuk mengelola aplikasi *container* dengan lebih efisien. Sistem ini mengelola *cluster*, yaitu sekelompok mesin atau *node*, di mana setiap *node* dapat menjalankan beberapa *container*, memungkinkan proses *deployment* dan *operations* serta pengelolaan aplikasi dilakukan secara efisien.

D. Docker

Docker adalah platform *open-source* yang memudahkan pengemasan dan pelaksanaan aplikasi dalam lingkungan terisolasi yang dikenal sebagai *container*. Dengan Docker, proses deployment dan pengembangan aplikasi menjadi lebih efisien karena penggunaan sumber daya yang hemat dan penyediaan *environment* yang stabil di berbagai perangkat, seperti *cloud server* atau komputer pribadi.

Docker memungkinkan layanan untuk dipaketkan bersama pustaka dan perangkat lunak yang dibutuhkan dalam satu unit *container* [8].

Dalam proyek ini, beberapa komponen *Docker* yang digunakan adalah:

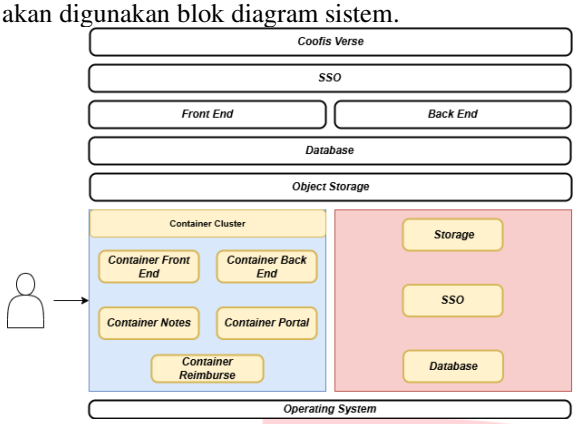
1. Docker *Image*: Template read-only untuk membuat container, di mana setiap layer menunjukkan perubahan dari *image*.
2. *Dockerfile*: File yang berisi perintah instalasi dan konfigurasi untuk membangun *image*.
3. Docker *Build*: Perintah untuk membuat *image* Docker berdasarkan perintah dalam *Dockerfile*.
4. Docker *Push*: Perintah untuk meng-upload *image* ke Docker *registry*.
5. Docker *Pull*: Perintah untuk mengunduh *image* dari Docker *registry* ke perangkat lokal.
6. Docker *Registry*: Tempat penyimpanan *image* Docker, yang dapat diatur sebagai private atau publik.

III. PERANCANGAN DAN MODEL SISTEM

A. Deskripsi

Pada pengerjaan Proyek Akhir ini dilakukan perancangan dan implementasi *Container Orchestration Cluster* untuk produk Coofis Verse menggunakan *Kubernetes*. Platform ini dipilih karena kelebihan dan kekurangan yang sesuai dengan kebutuhan Coofis Verse, sebuah produk PT. Andal Rancang Multi Solusi untuk tata kelola persuratan dengan fitur seperti dashboard, surat internal, surat eksternal, dan surat disposisi. Untuk membuat

arsitektur produk menjadi lebih mudah dipahami, akan digunakan blok diagram sistem.



GAMBAR 3. 1
Blok Diagram Sistem

Dari blok diagram diatas dapat disimpulkan bahwa penulis berfokus dalam penyediaan *container*. Dalam hal ini, penulis membuat beberapa *container* yang terdiri dari *container frontend*, *container backend, container notes, container portal,container reimburse*.

B. Kebutuhan Perangkat

1. Spesifikasi Perangkat Keras (Hardware)

Perangkat keras yang digunakan adalah sebuah laptop dengan spesifikasi:

TABEL 3. 1
Spesifikasi Hardware

a.	Device name: LAPTOP-5G9VM8OL
b.	Processor: ASUS 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz 3.11 GHz.
c.	Installed RAM 8.00 GB (7.69 GB usable)

2. Spesifikasi Perangkat Lunak (Software)

Perangkat lunak yang digunakan adalah beberapa *software* dengan spesifikasi:

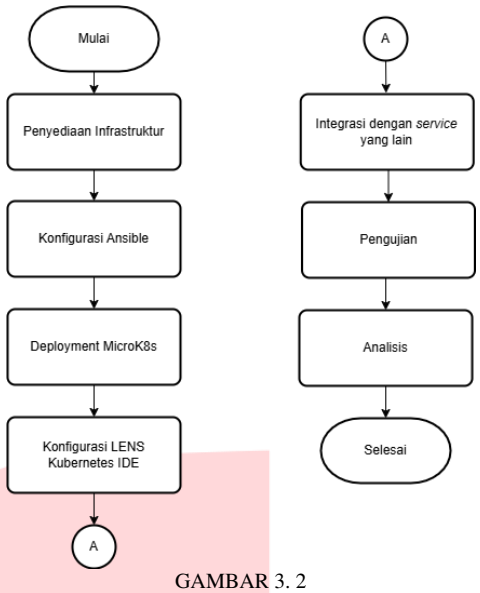
TABEL 3. 2
Spesifikasi Software

Termius	8.12.7
Lens Kubernetes IDE	2024.4.230844-latest
Apache JMeter	5.6.3

C. Perancangan Sistem

Pada perancangan sistem ini mengenai implementasi *container cluster orchestration* pada produk Cofis Verse dengan menggunakan Kubernetes Microk8s dan dilakukan dengan dua proses yaitu *deployment* dan *operations*.

1. Deployment



GAMBAR 3. 2
Flowchart Deployment

Pada tahap ini akan dijelaskan mengenai proses perencanaan *container cluster* dengan menggunakan MicroK8s yang dilakukan melalui tahapan *deployment*.

a. Tahap pertama, dilakukan penyediaan infrastruktur yang meliputi beberapa hal seperti penyiapan server fisik atau memastikan server virtual machine sudah berjalan, instalasi sistem operasi Linux, instalasi tools otomatisasi Ansible dan konfigurasi jaringan.

GAMBAR 3. 3
Penyediaan Server

b. Tahap kedua, Tahap kedua, melakukan konfigurasi pada Ansible. Konfigurasi ini meliputi beberapa command yang berisikan perintah untuk melakukan instalasi pada tools yang nantinya akan digunakan dalam pengerjaan proyek akhir seperti MicroK8s, MinIO, MongoDB, Keycloak. Dan, ansible akan dijalankan pada proses operations menggunakan ansible playbook.

```
root@telkombg:~/ansible# ls -l
total 44
-rw-r--r-- 1 root root   79 May 14 08:10 db.yaml
-rw-r--r-- 1 root root  556 May 14 08:10 example_hosts
drwxr-xr-x 2 root root 4096 Jul 18 16:07 group_vars
-rw-r--r-- 1 root root  560 Jul 18 16:02 host.ini
drwxr-xr-x 2 root root 4096 Jul 11 16:04 key
-rw-r--r-- 1 root root   83 May 14 08:10 kube.yaml
-rw-r--r-- 1 root root   99 May 14 08:10 reverseproxy.yaml
drwxr-xr-x 9 root root 4096 May 14 08:10 roles
-rw-r--r-- 1 root root  153 May 14 08:55 site.yaml
-rw-r--r-- 1 root root   81 May 14 08:10 sso.yaml
-rw-r--r-- 1 root root   89 May 14 08:10 storage.yaml
root@telkombg:~/ansible#
```

GAMBAR 3. 4
Ansible

c. Tahap ketiga, yaitu melakukan *deployment* pada microk8s dengan mendapatkan *source code* untuk konfigurasi dengan menggunakan *command "sudo microk8s config"*.
d. Tahap keempat, dilakukan konfigurasi pada tools Lens Kubernetes IDE dengan melakukan ekspor microk8s config yang telah disiapkan pada langkah sebelumnya.

```
root@telkombg:~# sudo microk8s config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRU
UF3d01NVEF1TVRVeUxqRTRNeTR4TUI0WERUSTBNRFV
lCQ2dLQ0FRRUF6enBmNEFwcGRSNWt1RC9CTDdXa0NS
sZUYvUi9lWHRRdjkraW5XeDk4cVNmN3ZyN2hUM2VvK
Zk5xZTZGKzk4bGJSSWxkb2hLSEhvaFBmcGxJTk5IZU
DJiL25KWDNzcHgvSXdId1lEVlIwakJCZ3dGb0FVUWx
5xSjNXYnlaVGNhUDdTUkRGdHViT1FpVFJ4Q0IyZ0lw
4a1ptVHN2SkNIQ0lKRXYxS2JmK053SURhbUxBb3JYb
VDVFUmdlb2cvcDN0K1hEMlNnNW5wOGlrWWRIMXcKWw
    server: https://139.180.155.23:16443
  name: microk8s-cluster
contexts:
- context:
    cluster: microk8s-cluster
    user: admin
  name: microk8s
```

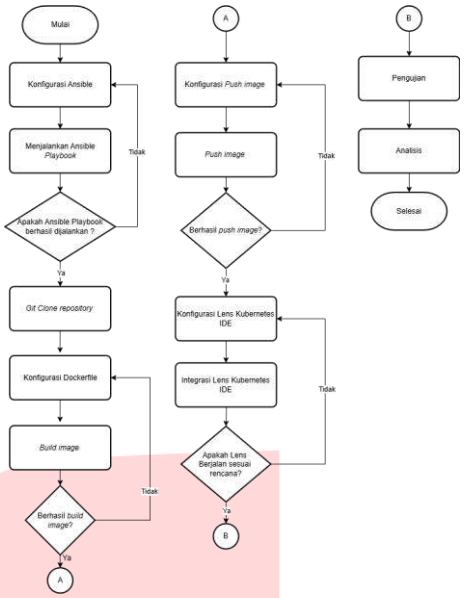
GAMBAR 3. 5
MicroK8s Config

e. Tahap kelima, dilakukan integrasi terhadap service yang lain, seperti melakukan integrasi terhadap SSO (Single Sign On), MinIO, MongoDB dengan membutuhkan beberapa file YAML.

```
root@telkombg:~# cd lens
root@telkombg:~/lens# ls
backend  frontend  notes  portal  reimburse  sso
root@telkombg:~/lens#
```

GAMBAR 3. 6
Konfigurasi Lens

2. Operations



GAMBAR 3. 7
Flowchart Operations

Pada tahap ini akan dijelaskan mengenai proses perencanaan *container cluster* dengan menggunakan MicroK8s yang dilakukan melalui tahapan *operations*.

1. Tahap pertama, yaitu memastikan konfigurasi ansible yang dibuat telah sesuai. Selanjutnya, melakukan *running* terhadap ansible yang telah dikonfigurasi pada tahap deployment. *Running* ansible, dilakukan dengan menggunakan *command* "ansible-playbook -I <*file* inventory> <nama .yaml yang akan dijalankan>. Selanjutnya, hal yang dilakukan selanjutnya adalah dengan memastikan bahwa *tools* yang kita *install* telah berjalan.

```
root@telkombg:~# sudo microk8s status
microk8s is running
high-availability: no
  datastore master nodes: 127.0.0.1:19001
  datastore standby nodes: none
addons:
```

GAMBAR 3. 8
Status MicroK8s

2. Tahap kedua, melakukan *git clone repository* pada Gitea dengan menggunakan *command* "*git clone <link repository remote> <nama direktori yang akan dibuat>*".
3. Tahap ketiga, yaitu melakukan konfigurasi pada Dockerfile yang nantinya akan digunakan pada proses *build image*.
   a. Dockerfile *frontend*

```
FROM node:18-bullseye-slim as builder
WORKDIR /app
```

   b. Dockerfile *backend, reimburse,portal,notes*

```
FROM python:3.10-slim-bullseye
ENV APP_HOME=/app
```

4. Tahap keempat, melakukan *build image*

menggunakan Docker dengan *command* "Docker build -t <nama images> .".

a. *Build image*

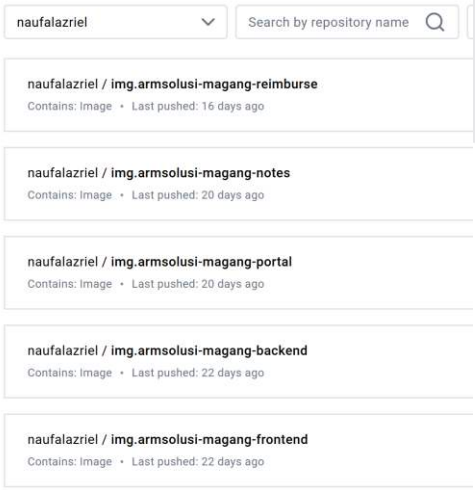
```
ubuntu@ip-172-31-19-167:~/fix-pa/frontend-pa$ Docker build -t naufalazriel/img.armsolusi-magang-frontend:latest
ubuntu@ip-172-31-19-167:~/fix-pa/backend-pa$ Docker build -t naufalazriel/img.armsolusi-magang-backend:latest
ubuntu@ip-172-31-19-167:~/fix-pa/notes-pa$ Docker build -t naufalazriel/img.armsolusi-magang-notes:latest
ubuntu@ip-172-31-19-167:~/fix-pa/portal-pa$ Docker build -t naufalazriel/img.armsolusi-magang-portal:latest
ubuntu@ip-172-31-19-167:~/fix-pa/reimburse-pa$ Docker build -t naufalazriel/img.armsolusi-magang-reimburse:latest
```

5. Tahap kelima, melakukan *push image*. *Image* yang telah dibuat, selanjutnya akan disimpan pada *image registry* menggunakan Docker Hub.

a. *Push Image*

```
ubuntu@ip-172-31-19-167:~/fix-pa/frontend-pa$ docker push naufalazriel/img.armsolusi-magang-frontend:latest
ubuntu@ip-172-31-19-167:~/fix-pa/backend-pa$ docker push naufalazriel/img.armsolusi-magang-backend:latest
ubuntu@ip-172-31-19-167:~/fix-pa/notes-pa$ docker push naufalazriel/img.armsolusi-magang-notes:latest
ubuntu@ip-172-31-19-167:~/fix-pa/portal-pa$ docker push naufalazriel/img.armsolusi-magang-portal:latest
ubuntu@ip-172-31-19-167:~/fix-pa/reimburse-pa$ docker push naufalazriel/img.armsolusi-magang-reimburse:latest
```

Pada Gambar 3.9 dibawah, dapat dilihat bahwa *images* telah berhasil disimpan ke *images registry.*



GAMBAR 3. 9
Image Registry

6. Tahap keenam yaitu dilakukan konfigurasi pada Lens Kubernetes IDE dengan membutuhkan file .yaml untuk membuat *container*.

a. Konfigurasi *Frontend*

1) *Deployment*

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: frontend
```

2) *Service*

```
apiVersion: v1
kind: Service
metadata:
  name: frontend
  labels:
    app: frontend
```

3) *Ingress*

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: frontend
```

b. Konfigurasi *Backend*

1) *Deployment*

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend
  labels:
    app: backend
```

2) *Service*

```
apiVersion: v1
kind: Service
metadata:
  name: backend
  labels:
    app: backend
```

3) *Ingress*

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: backend
```

c. Konfigurasi *Notes*

1) *Deployment*

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: notes
  labels:
    app: notes
```

2) *Service*

```
apiVersion: v1
kind: Service
metadata:
  name: notes
```

3) *Ingress*

```
apiVersion: networking.k8s.io/v1
```

```
kind: Ingress
metadata:
  name: notes
```

d. Konfigurasi Portal

1) *Deployment*

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: portal
  labels:
    app: portal
```

2) *Service*

```
apiVersion: v1
kind: Service
metadata:
  name: portal
  labels:
    app: portal
```

3) *Ingress*

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: portal
```

e. Konfigurasi *Reimburse*

1) *Deployment*

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: reimburse
  labels:
    app: reimburse
```

2) *Service*

```
apiVersion: v1
kind: Service
metadata:
  name: reimburse
```

3) *Ingress*

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: reimburse
```

f. Konfigurasi SSO

1) *Service*

```
apiVersion: v1
kind: Service
metadata:
  name: sso
```

2) *Endpoint*

```
apiVersion: v1
kind: Endpoints
metadata:
  name: sso
```

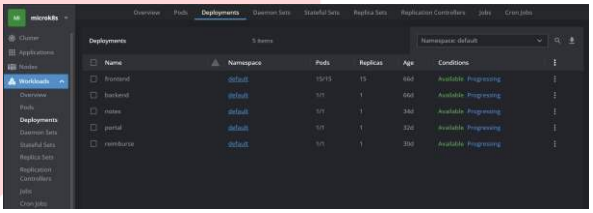
3) *Ingress*

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: sso
```

IV. HASIL DAN PENGUJIAN

A. Hasil Lens Kubernetes IDE

Pada perancangan dengan menggunakan Lens Kubernetes, dihasilkan beberapa *container* yang berhasil dibuat dengan menggunakan beberapa *file* yang dikonfigurasi sebelumnya. Pada Gambar 4.1 dapat terlihat bahwa proses konfigurasi antara masing-masing *service* kedalam Lens Kubernetes telah berhasil dan dibuktikan dengan status *running* masing-masing *pods*.



GAMBAR 3. 10
Hasil Perancangan Container

B. Pengujian Fungsional

Pengujian fungsional dilakukan pada sistem yang dirancang untuk memastikan bahwa semua fungsi aplikasi atau sistem beroperasi sesuai harapan. Berikut adalah hasil pengujian yang dilakukan:

TABEL 4. 1
Pengujian Fungsional

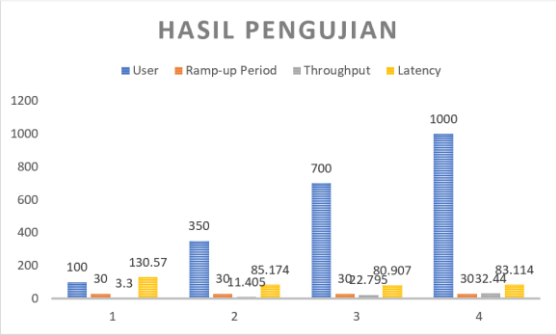
No	Case Pengujian	Deskripsi Pengujian	Hasil Pengujian
1.	Proses Konfigurasi	Melakukan konfigurasi *file deployment, service, ingress,* dan *endpoint* dengan menggunakan Lens Kubernetes IDE.	Proses konfigurasi *file deployment, service, ingress,* dan *endpoint* dapat dilakukan dengan baik.
2.	Deployment pada produk Coofis	Melakukan proses *deploy* pada produk Coofis Verse kedalam *cluster* Kubernetes dengan menggunakan Lens Kubernetes IDE.	Proses *deploy* produk Coofis Verse berhasil dilakukan.
3.	*Scalling* pada *pods*	Menguji cluster dengan melakukan *scaling* aplikasi, yaitu meningkatkan jumlah pod dari 1 menjadi 15.	Hasil menunjukkan bahwa *scaling* dilakukan dengan baik dan waktu proses *scaling* adalah 10 detik.
4.	*Monitoring Resource*	Menguji sistem *monitoring* untuk memantau penggunaan CPU dan memori selama operasi aplikasi Coofis.	Penggunaan CPU dan memori dapat di*monitoring*, yang dimana terdapat grafik penggunaan CPU dan memori pada Lens Kubernetes IDE
5.	*Logging* dan *Troubleshooting*	Menguji akses ke log dan kemudahan *troubleshooting* menggunakan Lens, memeriksa kejelasan log dan efisiensi proses *troubleshooting*.	Hasil menunjukkan log dapat terbaca jelas dan proses *troubleshooting* dapat dilakukan dengan cepat.

3. Hasil Pengujian Kinerja Sistem

TABEL 4. 2
Hasil Pengujian Keseluruhan

No	Parameter	Server Name	Pengujian			
			User	Ramp-up Period	Throughput	*Latency*
1	GET	new.mainke.cloud	100	30 sec	3.3 req/sec	130.57ms
2	GET	new.mainke.cloud	350	30 sec	11.405 req/sec	85.174ms
3	GET	new.mainke.cloud	700	30 sec	22.795 req/sec	80.907ms
4	GET	new.mainke.cloud	1000	30 sec	32.44 req/sec	83.114ms

Dari keseluruhan pengujian, dapat disimpulkan bahwa nilai *throughput* akan bertambah seiring bertambahnya juga jumlah *user*. Sedangkan, nilai *latency* akan menurun seiring bertambahnya jumlah *user*.



GAMBAR 4. 1
Grafik Pengujian

Dari grafik pengujian diatas, dapat disimpulkan bahwa nilai troughput meningkat saat bertambahnya jumlah *user*. Pada hal ini *throughput* didapat nilai 3.3 req/sec saat 100 *user*, kemudian meningka menjadi 11.405/sec saat ditambah menjadi 350 *user*, kemudian meningkat lagi menjadi 22.795/sec saat 700 *user*, hingga didapat nilai tertinggi yaitu 32.44req/sec saat 1000 *user*. Sehingga, dapat dikatakan bahwa server mampu menangani jumlah permintaan yang banyak dari *user*.

Sedangkan, pada hasil *latency* dapat disimpulkan bahwa nilainya menurun saat adanya peningkatan jumlah *user*, yang dimana nilainya yaitu dari didapat 130.57 ms pada 100 *user*, lalu menurun menjadi 85.174 ms saat 350 *user*, kemudian mengalami pernurunan kembali menjadi 80.907 ms saat 700 *user*, serta terdapat sedikit peningkatan *latency* saat jumlah *user* mencapai 1000 yaitu menjadi 83.114 ms. Secara umum, latensi yang rendah adalah indikator yang baik dari kinerja suatu sistem.

V. KESIMPULAN

A. Kesimpulan

Kesimpulan yang didapat dengan berdasarkan dari hasil perancangan, pengujian serta analisa adalah sebagai berikut:

1. Perancangan *container cluster* pada proyek akhir ini menggunakan Kubernetes *MicroK8s* yang pengelolaan *container*-nya menggunakan *software* Lens Kubernetes IDE.
2. Dengan menggunakan Kubernetes *MicroK8s* untuk perancangan *container* memungkinkan proses *deployment*, *scalling*, dan *monitoring* serta manajemen aplikasi menjadi lebih efisien.
3. Berdasarkan hasil perancangan, *service* yang terdapat dalam *container* dapat diakses dengan menggunakan domain yang telah diatur pada saat proses *deploy*.

B. Saran

Saran yang dapat diberikan setelah dilakukan proses perancangan adalah sebagai berikut:

1. Berdasarkan hasil pengujian, sistem ini belum dapat bekerja dengan optimal. Sehingga perlu dilakukan pengembangan lebih lanjut
2. Melakukan pengujian dengan skala pengguna yang lebih besar untuk memastikan bahwa sistem tetap stabil dan responsif di bawah beban yang lebih tinggi.
3. Melakukan *monitoring real-time* terhadap kinerja sistem agar membantu dalam mengidentifikasi dan mengatasi masalah kinerja dengan lebih cepat

REFERENSI

[1] J. Riset, M. Bidang, T. Informasi, D. Radya Briantama, and N. D. Hendrawan, "'Bimasakti' Aplikasi Persuratan Digital Berbasis Web Untuk Manajemen Dokumen Dengan Metode Addie." [Online]. Available: https://ejournal.unikama.ac.id/index.php/JFTI

[2] K. F. Ribawanto and D. Pramono, "Pengembangan Sistem Nota Dinas Elektronik dengan Tanda Tangan Elektronik Studi Kasus PT Andal Rancang Multi Solusi (Arm Solusi)," 2022. [Online]. Available: http://j-ptiik.ub.ac.id

[3] N. Surname and J. Mukaj, "Containerization: Revolutionizing Software Development And Deployment Through Microservices Architecture Using Docker And Kubernetes A Thesis Submitted To The Faculty Of Achitecture And Engineering Of Epoka University," 2023.

[4] Z. Zhong, M. Xu, M. A. Rodriguez, C. Xu, and R. Buyya, "Machine Learning-based Orchestration of Containers: A Taxonomy and Future Directions," Jun. 2021, [Online]. Available: http://arxiv.org/abs/2106.12739

[5] A. Sinambela and F. Farady Coastera, "Implementasi Arsitektur Microservices Pada Rancang Bangun Aplikasi Marketplace Berbasis Web," 2021. [Online]. Available: http://ejournal.unib.ac.id/index.php/rekursif/1

[6] Z. Zhong, M. Xu, M. A. Rodriguez, C. Xu, and R. Buyya, "Machine Learning-based Orchestration of Containers: A Taxonomy and Future Directions," Jun. 2021, [Online]. Available: http://arxiv.org/abs/2106.12739

[7] M. A. Nugroho and C. Subiyantoro, "Analisis Cluster Container Pada Kubernetes Dengan Infrastruktur Google Cloud Platform."

[8] M. F. ; A. Romadlon, "Implementasi Docker Untuk Pengelolaan Banyak Aplikasi Web".