Perancangan Infrastruktur Kubernetes Untuk Aplikasi Data Center Infrastructure Management (Dcim) Studi Kasus Pt. Pelayaran Nasional Indonesia (Pelni)

1st Vinandita Ayu Kinanti
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
vinanditaak@student.telkomuniversity.ac.id

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

3rd Chandra Mahendra Putra
Pt. Pelayaran Nasional Indonesia
Bandung, Indonesia
chandra@pelni.co.id

Abstrak - Di era digital, PT. Pelayaran Nasional Indonesia (PELNI) harus memperkuat infrastruktur IT untuk mengelola data dengan efesien,a aman, mengevaluasi infrastruktur Kubernetes berbasis microservices untuk aplikasi Data Center Infrastructure Management (DCIM). Prosesnya meliputi identifikasi kebutuhan, pemilihan arsitektur, pengaturan kluster Kubernetes, dan deployment aplikasi sebagai layanan terpisah. Hasilnya menunjukkan Kubernetes meningkatkan efesiensi operasional dan ketersediaan data center PT.PELNI, sementara arsitektur microservices memungkinkan layanan yang lebih fault tolerant, skalabel, dan mudah dipelihara, serta mengidentifikasi tantangan dan peluang teknis dalam penerapannya.

Kata kunci : kubernetes, dcim, microservices

I. PENDAHULUAN

Dalam era digital, perusahaan perlju memperkuat infrastruktur IT untuk mengelola data dengan efesien, aman, dan terukur. PT. Pelayaran Nasional Indonesia (PELNI) menghadapi kebutuhan mendesak untuk meningkatkan manajemen infrastruktur *data center* yang kompleks. Dengan menggunakan Kubernetes dan arsitektur *microservices*, PT. PELNI memperbaharui sistem manajemen *Data Center Infrastructure Management* (DCIM) mereka, memungkinkan aplikasi DCIM dibangun sebagai layanan terpisah yang lebih mudah dikembangkan, diimplementasikan, dan dipelihara.

Proyek ini bertujuan merancang, mengimplementasikan, dan mengevaluasi infrastruktur Kubernetes untuk aplikasi DCIM di PT. PELNI. Prosesnya mencakup identifikasi kebutuhan sistem, pemilihan arsitektur yang tepat, pengaturan kluster Kubernetes, konfigurasi, pengaturan kontainer, dan *deployment* aplikasi. Hasilnya menunjukkan peningkatan efesiensi operasional dan ketersediaan layanan, serta layanan yang lebih *fault tolerant* dan mudah dipelihara. Studi ini juga mengidentifikasi tantangan teknis dan peluang dari penerapan Kubernetes dalam lingkungan perusahaan besar, memberikan panduan praktis bagi implementasi di sektor industri serupa.

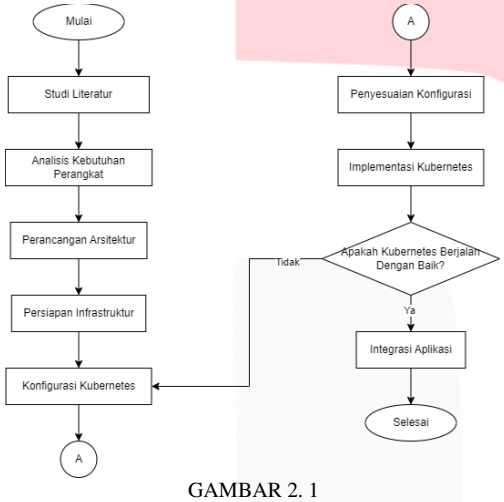
Rumusan masalah meliputi bagaimana merancang dan mengevaluasi infrastruktur Kubernetes untuk mendukung aplikasi DCIM. Batasan masalah mencakup perancangan dan implementasi infrastruktur menggunakan Kubernetes untuk aplikasi DCIM, tanpa mencakup aplikasi atau aspek lain. Proyek ini juga akan menyediakan data yang diperlukan untuk analisis dan evaluasi efektivitas infrastruktur Kubernetes terhadap aplikasi DCIM, dengan manfaat meliputi peningkatan efisiensi operasional dan kualitas layanan aplikasi DCIM.

II. PERANCANGAN DAN MODEL SISTEM

Pada Proyek Akhir ini dilakukan perancangan infrastruktur untuk mendukung aplikasi *Data Center Infrastructure Management* (DCIM). Pemilihan Kubernetes sebagai solusi infrastruktur dilakukan karena teknologi ini memiliki kelebihan dalam hal skalabilitas, otomatisasi, dan manajemen *container* yang efesien, yang sangat sesuai dengan kebutuhan operasional perusahaan. Studi kasus akan difokuskan pada penerapan di lingkungan *data center* PT. PELNI.

A. Tahap Pengerjaan

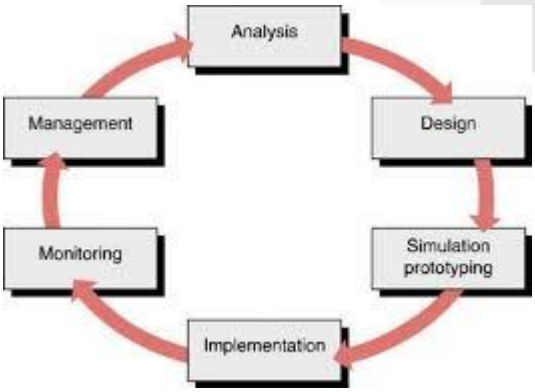
Proses dimulai dengan mempelajari konsep dasar Kubernetes, termasuk arsitektur dan komponen utamanya. Setelah memahami kebutuhan aplikasi DCIM, arsitektur infrastruktur dirancang menggunakan Kubernetes sebagai sistem orkestrasi utama. Tahap berikutnya adalah pemilihan perangkat keras, menggunakan Orange Pi sebagai server node dalam kluster Kubernetes. Setiap node dikonfigurasi dengan sistem operasi yang sesuai dan instalasi Kubernetes untuk membentuk kluster. Selanjutnya, setiap node Orange Pi dikonfigurasi dan dihubungkan ke dalam kluster. Tahap akhir meliputi uji coba dan evaluasi kinerja sistem, mencakup fungsionalitas, keamanan, skalabilitas, dan kinerja infrastruktur Kubernetes. Penyempurnaan dan koreksi dilakukan untuk memastikan sistem beroperasi sesuai harapan.



GAMBAR 2. 1
Flowchat Perancangan Sistem

B. Metode

Proses perancangan sistem ini dilakukan dengan metode NDLC. Dari ke-enam fase yang terdapat pada NDLC, penulis mengguan 6 (enam) fase anatara lain sebagai berikut: *analysis, design, simulation prototyping, implementation, monitoring,* dan *management.*

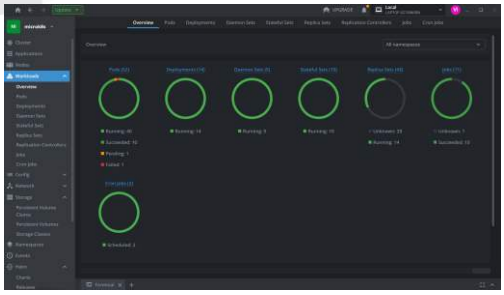


GAMBAR 2. 2
Metode NDLC

Pada tahap *analysis* penulis melakukan pengumpulan data dengan cara studi literatur, penulis membaca jurnal ilmiah untuk mendapatkan informasi mengenai Kubernetes, *microservices*, DCIM, dan NetBox, data yang telah terkumpul kemudian dianalisis. Hasil dari studi literatur ini menunjukkan bahwa belum ada penelitian yang secara khusus mengkaji tentang penerapan Kubernetes untuk aplikasi DCIM seperti NetBox. Dari studi kasus PT. Pelni yang diambil sebagai conton, penulis menemukan permasalahan yang mendorong untuk dilakukan peneletian lebih lanjut mengenai desain infrastruktur Kubernetes untuk aplikasi DCIM.

Pada tahap *Design* dan *Simulation Prototype*, penulis membuat rancangan sistem menggunakan Kubernetes. Rancangan ini mencakup alokasi pengalamatan IP untuk setiap *node* dan spesifikasi perangkat keras serta perangkat lunak yang diperlukan. Dalam rancangan ini, tiga server Orange Pi digunakan sebagai node dalam kluster Kubernetes, yang masing-masing difungsikan sebagai worker node: worker 1 dengan IP 172.16.150.8, worker 2 dengan IP 172.16.150.27, dan worker 3 dengan IP 172.16.150.46. Sistem operasi yang digunakan adalah Ubuntu versi 1.1.8, dipilih sesuai dengan kompatibilitas *hardware* dari Orange Pi. Dilakukan instalasi dan konfigurasi awal sistem yang telah dirancang yang melibatkan pengujian aplikasi yang telah di-deploy menggunakan Kubernetes untuk memastikan bahwa semua komponen bekerja secara efektif dan efisien sesuai dengan desain. Uji coba dilakukan dengan mengakses aplikasi dalam kondisi seluruh server node aktif, menilai kinerja dan fungsi aplikasi yang di-deploy.

```
orangepi@worker01:~$ microk8s status
microk8s is running
high-availability: yes
   datastore      master      nodes:
   172.16.150.8:19001
   172.16.150.27:19001
   172.16.150.46:19001
  datastore standby nodes: none
```



GAMBAR 2. 3
Workloads Kubernetes

Pada tahap *Management*, penulis melakukan evaluasi berkelanjutan dan pengelolaan sumber

daya sistem. Manajemen ini termasuk pembaruan periodik pada software, skala sumber daya sesuai dengan kebutuhan, dan penyesuaian konfigurasi untuk mengoptimalkan performa dan keamanan keseluruhan sistem.

Dengan mengikuti tahapan NDLC ini, penulis berharap untuk mengembangkan sistem infrastruktur yang kuat, skalabel, dan efisien untuk mendukung operasi DCIM di PT. PELNI menggunakan teknologi Kubernetes dan arsitektur berbasis *microservices*.

C. Perancangan Sistem

1. *Install MicroK8s*

Install MicroK8s dan pastikan mengikuti panduan yang disediakan oleh MicroK8s untuk sistem operasi yang digunakan.

```
orangepi@worker01:~$ sudo snap install microk8s
snap "microk8s" is already installed, see 'snap help refresh'
```

2. *Join the group*

Microk8s membuat grup khusus untuk memudahkan penggunaan perintah yang memerlukan hak admin. Untuk menambahkan pengguna saat ini ke grup dan mendapatkan akses ke direktori *cache* .kube, jalankan tiga penrintah berikut:

```
sudo usermod -a -G microk8s $USER
mkdir -p ~/.kube
chmod 0700 ~/.kube
```

3. *Check the status*

MicroK8s memiliki perintah bawaan untuk menampilkan statusnya. Selama instalasi, dapat menggunakan *command flag* '—wait ready' untuk menunggu hingga layanan Kubernetes ter-install.

```
microk8s status --wait-ready
```

```
orangepi@worker01:~$ microk8s status
microk8s is running
high-availability: yes
 datastore master nodes: 172.16.150.8:19001 172.16.150.27:19001 172.16.150.46:19001
 datastore standby nodes: none
addons:
 enabled:
   dns                  # (core) CoreDNS
   ha-cluster           # (core) Configure high availability on the current node
   helm                 # (core) Helm - the package manager for Kubernetes
   helm3                # (core) Helm 3 - the package manager for Kubernetes
   hostpath-storage     # (core) Storage class; allocates storage from host directory
   ingress              # (core) Ingress controller for external access
   observability        # (core) A lightweight observability stack for logs, traces and metrics
   storage              # (core) Alias to hostpath-storage add-on, deprecated
 disabled:
   cert-manager         # (core) Cloud native certificate management
   cis-hardening        # (core) Apply CIS K8s hardening
   community            # (core) The community addons repository
   dashboard            # (core) The Kubernetes dashboard
   host-access          # (core) Allow Pods connecting to Host services smoothly
   kube-ovn             # (core) An advanced network fabric for Kubernetes
   mayastor             # (core) OpenEBS MayaStor
   metallb              # (core) Loadbalancer for your Kubernetes cluster
   metrics-server       # (core) K8s Metrics Server for API access to service metrics
   minio                # (core) MinIO object storage
   prometheus           # (core) Prometheus operator for monitoring and logging
   rbac                 # (core) Role-Based Access Control for authorisation
   registry             # (core) Private image registry exposed on localhost:32000
   rook-ceph            # (core) Distributed Ceph storage using Rook
```

4. *Access Kubernetes*

microK8s sudah dilengkapi dengan versi kubectl sendiri untuk mengakses Kubernetes. Gunakan perintah ini untuk memonitor dan mengontrol Kubernetes. Missal, untuk melihat node:

```
orangepi@worker01:~$ microk8s kubectl get no
NAME     STATUS ROLES   AGE   VERSION
worker02 Ready  <none>  91d   v1.28.10
worker03 Ready  <none>  5d5h  v1.29.5
worker01 Ready  <none>  91d   v1.28.10
```

5. *Deploy an app*

```
microk8s kubectl create deployment nginx --image=nginx
```

Instalasi memerlukan waktu satu hingga dua menit. Status dapat dilihat menggunakan *command:*

```
orangepi@worker01:~$ microk8s kubectl get pods
NAME                     READY STATUS  RESTARTS AGE
nginx-7898db687-485sz    1/1   Running 0        15d
```

6. *Use add-ons*

Terdapat banyak fitur tambahan yang tersedia melalui command 'add-ons'. Menambahkan manajemen DNS untuk memfasilitasi komunikasi antara layanan. Untuk aplikasi yang memerlukan penyimpanan, *add-on* 'hostpath-storage' menyediakan ruang direktori pada host.

```
microk8s enable dns
microk8s enable hostpath-storage
```

7. *Adding a Node*

Untuk membuat kluster dari dua atau lebih *instance* MicroK8s yang sudah berjalan, gunakan perintah 'microk8s add-node'.

```
microk8s add-node
```

Dapat melihat bahwa node telah bergabung dengan *command:*

```
orangepi@worker01:~$ microk8s add-node
From the node you wish to join to this cluster, run the following:
microk8s join 172.16.150.8:25000/35dbe0ed399d9370eba8a3d4ab468155/30365318528f

Use the '--worker' flag to join a node as a worker not running the control plane, eg:
microk8s join 172.16.150.8:25000/35dbe0ed399d9370eba8a3d4ab468155/30365318528f --worker

If the node you are adding is not reachable through the default interface you can use one of the following:
microk8s join 172.16.150.8:25000/35dbe0ed399d9370eba8a3d4ab468155/30365318528f
```

8. *Starting and Stopping MicroK8s*

MicroK8s akan terus berjalan hingga diputuskan untuk menghentikannya. Untuk menghentikan dan memulai MicroK8s dengan *command*:

```
microk8s stop
microk8s start
```

III. HASIL DAN PENGUJIAN

Pada bab ini dipaparkan hasil perancangan yang telah dirancang dan juga hasil dari pengujian fungsionalitas. Tahap pengujian merupakan langkah krusial yang dilakukan setelah sistem selesai dirancang, dengan tujuan untuk memastikan bahwa sistem berfungsi dengan baik. Pengujian ini dilakukan untuk mengumpulkan data yang dapat digunakan sebagai dasar dalam analisis kinerja sistem secara keseluruhan.

A. Hasil Sistem Kubernetes

Instalasi dan konfigurasi Kubernetes telah berhasil dikerjakan pada 3 (tiga) server yang telah di-inisialisasi sebagai node worker dalam kluster. Tiap server telah dikonfigurasi dengan spesifikasi yang memadai untuk mendukung deployment arsitektur berbasis microservices. Proses deployment ini menggunakan Kubernetes sebagai sistem orkestrasi kontainer yang akan menangani distribusi dan manajemen kontainer secara otomatis dalam lingkungan operasional yang aktif.

```
orangepi@worker01:~$ microk8s kubectl get no
NAME             STATUS     ROLES       AGE
VERSION
worker02      Ready         <none>      91d
v1.28.10
worker03      Ready        <none>      5d5h
v1.29.5
worker01      Ready         <none>      91d
v1.28.10
```

B. Penggunaan CPU dan Memory

Berikut adalah data hasil pemantauan penggunaan CPU dan memory yang diperoleh dari dashboard Grafana. Penulis mengambil data terakhir pada tanggal 31 Mei 2024, yang kemungkinan akan mengalami perubahan di kemudian hari. Data ini mencakup pemakaian sumber daya CPU dan memory pada Kubernetes cluster yang menjalankan layanan Prometheus, Grafana, dan NetBox. Informasi yang disajikan diambil secara langsung dari metrik yang dikumpulkan oleh Prometheus dan divisualisasikan melalui Grafana, memberikan gambaran yang akurat mengenai performa dan beban kerja sistem. Rincian penggunaan sumber daya tersebut dapat dilihat pada tabel di bawah ini.

TABEL 3. 1
Penggunaan CPU dan Memory

No	Pod	CPU Usage	Memory Usage
1	netbox1-postgresql-0	-	-
2	netbox-redis-replicas-1	0.02	5.25 MiB
3	netbox-redis-replicas-2	0.04	12.17 MiB
4	netbox-redis-replicas-0	0.02	4.07 MiB
5	prometheus-deployment-556fbcc476-sc67n	0.30	837.68 MiB
6	prometheus-deployment-556fbcc476-jxczl	0.27	882.14 MiB
7	kube-prom-stack-prometheus-node-exporter-c68cz	0.04	18.04 MiB
8	nginx-ingress-microk8s-controller-qg85t	0.00	142.63 MiB
9	nginx-7898db687-485sz	0.00	6.91 MiB
10	netbox-worker-5b9b6b6bdc-92lm2	0.00	129.09 MiB
11	netbox-postgresql-0	0.02	53.41 MiB
12	netbox-5cd5bc89f-qjdr7	0.00	310.91 MiB

13	csi-nfs-node-6jndc	0.00	38.90 MiB
14	kube-prom-stack-grafana-777db85ffd-cg44t	0.02	304.31 MiB
15	grafana-764fd94976-5t99h	0.03	130.46 MiB
16	csi-nfs-controller-f89bcff87-dtc7w	0.01	47.71 MiB
17	prometheus-kube-prom-stack-kube-prome-prometheus-0	0.25	1.73 GiB
18	calico-node-552xf	0.06	82.34 MiB

C. Performance Test

Pada tahap pengujian, setiap aplikasi akan dilakukan performance test menggunakan aplikasi JMeter. Pengujian ini dilakukan dengan mengatur Number of Thread = 250 user, Ramp-Up Period = 10 second, dan Loop Count = 1, itu artinya 1 thread yang dijalankan setiap 25 detik. Dapat disimpulkan bahwa pada pengujian ini akan dilakukan oleh 250 user dengan jeda masing-masing user selama 25 detik dan masing-masing user melakukan akses sebanyak 1 kali.

User	***Ramp-Up Period***	**Prometheus**	**Grafana**	**NetBox**
250	10	24.7	23.6	24.9

Hasil performance test menggunakan JMeter telah memvalidasi bahwa ketiga aplikasi, yaitu Prometheus, Grafana, dan NetBox menunjukan kapasitas penanganan permintaan yang sangat tinggi serta stabilitas operasional yang baik. Ketiga aplikasi ini telah berhasil menunjukkan performa yang efesien dan kehandalan dalam menangani puncak beban permintaan. Penggunaan Kubernetes sebagai platform container orchestration memungkinkan pengelolaan sumber daya yang lebih efektif dan otomatisasi penyebaran aplikasi, meningkatkan skalabilitas yang dinamis, monitoring yang efektif, dan manajemen infrastuktur yang baik.

IV. KESIMPULAN

Berdasarkan hasil perancangan, pengujian, dan analisis yang telah dilakukan, dapat disimpulkan bahwa sistem Kubernetes yang dirancang telah berhasil menjalankan semua fungsinya dengan baik sesuai yang diharapkan. Pengujian fungsionalitas menunjukkan bahwa sistem beroperasi secara optimal. Selain itu, implementasi dan pengujian juga menunjukkan bahwa Kubernetes dapat terintegrasi dengan baik dengan Prometheus, Grafana, dan NetBox, memperkuat infrastruktur monitoring dan manajemen data center. Infrastruktur Kubernetes yang dirancang juga mampu menangani beban kerja yang tinggi dengan efisiensi dan stabilitas, menunjukkan kesiapan sistem untuk diimplementasikan dalam lingkungan operasional yang dinamis dan intensif.

REFERENSI

[1] A. Tanuwijaya, H. Novianus Palit, and A. Noertjahyana, “Penerapan Microservices dan Amazon Elastic Container Service untuk Mendukung Scalability,” 2021.

[2] K. Yedutun, A. Noertjahyana, and H. Novianus Palit, “Implementasi Container Kubernetes untuk Mendukung Scalability,” 2015.

[3] A. Nurul Huda and S. Kusumawardani, “Huda, Kusumawardani-Kubernetes Cluster Management for Cloud Computing Platform: A Systematic Literature Review 75 KUBERNETES CLUSTER MANAGEMENT FOR CLOUD COMPUTING PLATFORM: A SYSTEMATIC LITERATURE REVIEW MANAJEMEN KLASTER KUBERNETES PADA PLATFORM KOMPUTASI AWAN: TINJAUAN LITERATUR SISTEMATIK,” 2022.

[4] Editor, “Tentang PT. Pelni,” pelni.co.id.

[5] T. Sinta Peringkat, berdasarkan S. Dirjen Penguatan RisBang Kemenristekdikti, A. Firmansyah, and N. Merlina, “PREDIKSI POLA PENJUALAN TIKET KAPAL PT. PELNI CABANG MAKASSAR MENGGUNAKAN METODE ALGORITMA APRIORI,” 2020.

[6] Editor, “Apa itu Kubernetes?,” kubernetes.io.

[7] L. Widyawati, H. Santoso, and H. Budiman, “ANALISA PENERAPAN SERVER DEPLOYMENT MENGGUNAKAN KUBERNETES UNTUK MENGHINDARI SINGLE OF FAILURE,” 2021.

[8] M. Harris, “Data Center InfrastruCture ManageMent,” 2015. [Online]. Available: http://www.wiley.com/go/datacenterhandbook

[9] R. M. F. M. M. W. W. T. Muh. Usman C, “Aplikasi Sistem Monitoring Server Menggunakan Device Orange Pi Berbasis Web Service Studi Kasus PT. MNC Televisi Indonesia – MNC Group,” 2022.

[10] M. Dicky Syahputra Lubis *et al.*, “MEMBANGUN ROUTER PADA JARINGAN KOMPUTER MENGGUNAKAN UBUNTU OS,” *Jurnal Teknik Informatika Kaputama (JTIK)*, vol. 4, no. 2, 2020.

[11] L. M. Alchuluq and F. Nurzaman, “ANALISIS PADA ARSITEKTUR

MICROSERVICE UNTUK LAYANAN BISNIS TOKO ONLINE," 2021.

[12] S. Saidah and R. Syaban, "IMPLEMENTASI ARSITEKTUR MICROSERVICES PADA APLIKASI POINT OF SALE TOKO FLYOVER STIKER," 2023. [Online]. Available: https://flyoverstiker.online/,

[13] Editor, "NetBox Documentation," docs.netbox.dev.

[14] M. Fadlulloh and R. Bik, "IMPLEMENTASI DOCKER UNTUK PENGELOLAAN BANYAK APLIKASI WEB (Studi Kasus : Jurusan Teknik Informatika UNESA)," 2017.

[15] K. Thias Widagdo, I. Bayu, and Y. A. Susetyo, "Pemodelan Sistem Monitoring Sensor Curah Hujan Menggunakan Grafana," 2018.

[16] D. Rahman and H. Amnur, "Indri Rahmayuni 133 Monitoring Server dengan Prometheus dan Grafana serta Notifikasi Telegram Jurnal Ilmiah Teknologi Sistem Informasi," 2020. [Online]. Available: http://jurnal-itsi.org

[17] R. Normadhoni, S. Putri Dewanti, W. Cahyo Namaskara, D. Yusfi Akhadi, and R. Fauzi, "Penggunaan Bot Telegram sebagai Announcemnt System dalam Dunia Parenting," 2021. [Online]. Available: http://jurnalilmiah.org/journal/index.php/jet