

Implementasi Penggunaan Kubernetes Cluster Google Cloud Platform untuk Deployment Aplikasi Wiki.js

Ahmad Kusumo Haryo¹, Chandra Kusuma Dewa²

¹²Program Studi Teknik Informatika, Universitas Islam Indonesia

¹19523120@students.uui.sc.id, ²chandra.kusuma@uui.ac.id

Abstract

Every company has its own digital product, with the rapid pace of digital technology and the needs of users, it requires companies to be able to deliver their digital services, so they are always available and can be accessed at any time. In order to present digital services that are available at any time and in large quantities, the developers in the company create applications according to the large number of requests and needs, this creates problems in several ways, for example the more applications, the more servers are needed, of course this adds costs to application development, besides that there are dependencies between one application and another are make the application experience problems when the developer releases the latest version of the application. Google cloud platform is one of the platforms cloud computing which has many services, such as Google Kubernetes Engine. Kubernetes is an orchestration technology container which allows developer for management of containers in large numbers, this can be the one-off benefit for the application, so that is not easy to experience downtime, because container itself has separate resources from other applications. In this research will be tested for implementation deployment wiki.js application via kubernetes. The trial results of this research are the wiki.js application which can be accessed by utilizing the Kubernetes cluster technology.

Keywords: Kubernetes, cloud computing, google cloud platform, deployment, wikisjs

Abstrak

Setiap perusahaan memiliki produk digitalnya masing-masing, dengan pesatnya teknologi digital dan kebutuhan dari pengguna, mengharuskan perusahaan untuk dapat men-deliver layanan digitalnya agar selalu tersedia dan dapat diakses kapanpun. Demi menyajikan layanan digital yang ada setiap saat dan dalam jumlah yang banyak, para pengembang di perusahaan membuat aplikasi sesuai dengan banyaknya permintaan dan juga kebutuhan, hal ini menimbulkan masalah pada beberapa hal, misalnya semakin banyaknya aplikasi, semakin banyak pula server yang dibutuhkan, tentunya hal ini menambah biaya untuk pengembangan aplikasi, selain itu adanya dependensi antara aplikasi satu dengan yang lainnya membuat aplikasi tersebut mengalami masalah apabila pengembang mengeluarkan versi terbaru dari aplikasi. Google cloud platform merupakan salah satu platform cloud computing terkenal yang memiliki banyak layanan, seperti diantaranya Google Kubernetes Engine. Kubernetes merupakan teknologi orkestrasi container yang memungkinkan pengembang untuk memanajemen container dalam jumlah yang besar, hal tersebut dapat menguntungkan aplikasi agar tidak mudah mengalami downtime, karena container sendiri memiliki sumber daya yang terpisah dari aplikasi lainnya. Dalam penelitian ini akan dilakukan uji coba untuk penerapan - deployment aplikasi wiki.js melalui kubernetes. Hasil uji coba dari penelitian ini adalah aplikasi wiki.js yang dapat diakses dengan pemanfaatan dari teknologi cluster kubernetes.

Kata kunci: Kubernetes, cloud computing, google cloud platform, deployment, wikisjs

1. Pendahuluan

Dengan pesatnya perkembangan teknologi pada zaman sekarang ini membuat perusahaan harus berinovasi dalam menghasilkan produk – produk yang bermanfaat dan menguntungkan bagi perusahaan. Revolusi industri adalah perubahan cara hidup dan proses kerja manusia secara fundamental, yang mana teknologi informasi

diintegrasikan dalam kehidupan dengan dunia digital [1]. Apalagi dengan adanya revolusi industri 4.0, yang mengharuskan perusahaan untuk menggunakan teknologi sebagai upaya dalam bertransformasi secara digital dengan mengintegrasikan perkembangan teknologi di lini produksi sehingga teknologi merupakan bagian yang tidak terpisahkan pada proses bisnis di

sektor industri. Maka dari itu, banyak perusahaan berusaha untuk beradaptasi dan bertransformasi menuju ke ranah digital yang mana hal tersebut merupakan salah satu peluang besar demi kemajuan perusahaan.

Demi menghasilkan produk – produk digital yang berkualitas, setiap perusahaan yang bergerak dalam beberapa aspek perindustrian terus berinovasi mengembangkan berbagai layanan digital, dengan *output* berupa aplikasi yang diproyeksikan dapat memenuhi kebutuhan bisnis produk digital. Tentunya hal ini juga perlu disesuaikan dengan beberapa hal yang dibutuhkan agar layanan digital tersebut dapat disebarkan secara luas dan diraih konsumen seperti halnya perangkat yang mumpuni, SDM yang memiliki kapabilitas, dan pengelolaan yang berkualitas sehingga transformasi digital dapat berjalan sesuai dan berkesinambungan. Dari banyaknya aplikasi yang dikembangkan, tentu diharuskan juga memiliki sebuah lingkungan pengembangan aplikasi yang baik. Dengan banyaknya permintaan dari konsumen, maka dari itu pengembang melakukan banyak hal untuk mengembangkan aplikasi secara masif. Hal tersebut tentu membuat sumber daya yang dibutuhkan perlu ditingkatkan serta pemeliharaan sumber daya pastinya akan mengeluarkan biaya yang tidak sedikit, terlebih jika perangkat yang dimiliki dalam pengembangan aplikasi merupakan perangkat fisik, seperti halnya server, dsb.

Dengan adanya teknologi *cloud computing* hal pengembangan aplikasi dapat direalisasikan dengan tanpa adanya server fisik yang memerlukan tempat dan biaya tambahan dalam pengelolaannya. Salah satu layanan *cloud computing* yang terkemuka yaitu Google Cloud Platform. Google sebagaimana yang kita kenal merupakan salah satu raksasa teknologi yang layanannya banyak digunakan sebagai salah satu mesin mencari paling populer di dunia. Selain itu, Google menyediakan layanan lain seperti hiburan, dan layanan pendukung lainnya. Dengan penggunaan layanan dari GCP sebagai penyedia layanan infrastruktur teknologi informasi, pengembang bisa dengan mudah melakukan *research and development* terhadap aplikasi digital yang akan dikembangkan, dimulai dari riset, percobaan, hingga pada tahap perilisan aplikasi. Banyak layanan GCP yang dapat dimanfaatkan, seperti halnya Kubernetes Cluster.

Kubernetes cluster digunakan untuk mengelola aplikasi yang berbasis dari *container*. Setiap *container* yang berisi aplikasi tersebut dikelola menggunakan Kubernetes sebagai orkestrator kontainer aplikasi. Penggunaan kubernetes ini didasari karena banyaknya aplikasi yang ingin dikembangkan berdasarkan dari kebutuhan perusahaan dan juga untuk memudahkan pengembang dalam melakukan *maintenance* terhadap aplikasi yang dikembangkan. Dalam penelitian ini berisi

tentang langkah-langkah penggunaan dan juga implementasi untuk men-*deploy* aplikasi dengan menggunakan kubernetes. Dengan fungsinya yang sebagai *clustering management container* membuat layanan Kubernetes dapat membuat beberapa aplikasi agar selalu tersedia dan dapat menerima banyak permintaan.

1.1 Cloud Computing

Penggunaan Teknologi Informasi dari setiap tahunnya pasti akan berkembang, baik dari segi sistem informasi, bahasa pemrograman, dan bahkan infrastruktur yang merupakan salah satu pendukung bagi berjalannya sebuah aplikasi. *Cloud Computing* atau komputasi awan merupakan suatu menakisme yang mana terdapat sekumpulan Teknologi Informasi yang sumberdayanya saling terhubung dan nyaris tanpa batas. *Cloud Computing* sendiri merupakan bentuk komputasi terdistribusi, hal ini memungkinkan pengguna untuk mengaksesnya secara *remote* atau secara jauh asalkan perangkat yang digunakan terkoneksi dengan internet. Dikarenakan infrastruktur ataupun aplikasi yang dimiliki dan dikelola sepenuhnya oleh pihak ketiga [2], hal tersebut memungkinkan pengguna untuk menggunakan sumber daya tersebut tergantung dari kebutuhan pengguna. Berbeda dengan komputasi tradisional, pengguna diharuskan untuk memiliki perangkat keras dan perangkat lunaknya sendiri, hal ini tentu akan mengharuskan pengguna untuk mengeluarkan lebih banyak dana untuk pengelolaan dan pemeliharaan dari perangkat tersebut.

Terdapat 5 karakteristik yang ada pada *cloud computing*, yaitu [3]:

- On-demand. Hal ini memungkinkan pengguna untuk dapat mengonfigurasi layanan sesuai kebutuhan yang diinginkan.
- Broad network access. Layanan ini merupakan akses jaringan yang luas yang memungkinkan untuk dapat tersedia dari mana saja dan kapan saja dengan perangkat yang dapat mengakses layanan (seperti, laptop, smartphone, dll.).
- Resource pooling. Sumberdaya komputasi yang dimiliki penyedia layanan dapat digunakan secara bersamaan oleh beberapa pengguna.
- Rapid elasticity. Layanan ini disediakan secara elastis dan cepat untuk memenuhi kebutuhan, dengan ketersediaan dan kapasitas yang tidak terbatas sehingga dapat disesuaikan dengan spesifikasi yang diinginkan setiap saat.
- Measured Service. Terdapat sistem yang secara otomatis dapat mengawasi dan memaksimalkan sumber daya dengan memanfaatkan pengukuran untuk dapat mengukur penggunaan sumber daya komputasi yang telah terpakai.

1.2 Google Cloud Platform

Dengan seiring berjalannya waktu, komputasi awan yang sangat efektif saat ini banyak dibutuhkan oleh beberapa konsumen. Beberapa perusahaan yang bergerak di bidang IT banyak yang membuat dan menyewakan jasa layanan komputasi awan, seperti yang terkenal adalah *Amazon Web Service (AWS)*, *Microsoft Azure*, dan juga perusahaan lain yang menyediakan jasa layanan komputasi awannya dengan keunggulan yang berbeda, tidak terkecuali Google. Sebagai perusahaan raksasa di bidang IT, Google membuat salah satu platform komputasi awan mereka sendiri yang dinamai *Google Cloud Platform (GCP)*.

Google Cloud Platform merupakan layanan untuk memberikan kemudahan bagi setiap perusahaan untuk mengembangkan aplikasi dalam bisnis menawarkan berbagai produk layanan yang dapat digunakan untuk membangun infrastruktur server dengan tingkat standar yang tinggi. Setiap produk dan layanannya disediakan untuk memiliki serangkaian fitur dan manfaat yang berbeda sesuai dengan kebutuhan penggunaannya [4].

1.3 Kubernetes

Kubernetes adalah suatu layanan *open-source* yang mampu melakukan orkestrasi container dalam jumlah yang besar yang memiliki tugas untuk melakukan penjadwalan, penskalaan, recovery, dan monitoring container. Pada awalnya sistem ini dikembangkan oleh Google sebagai sistem internal mereka yang digunakan untuk membantu developer dan teknisi infrastruktur TI untuk mengelola ribuan server yang ada di Google yang dinamai dengan Borg kemudian Omeda pada beberapa tahun setelahnya. Dengan fungsinya yang sebagai clustering management container membuat layanan Kubernetes dapat membuat sebuah aplikasi selalu tersedia dan dapat menerima banyak permintaan. Pada penggunaan Kubernetes cluster dibutuhkan service sebagai jembatan yang dapat mengakses satu atau lebih dari satu Pod, dan service harus di ekspos ke luar jaringan agar aplikasi dari luar Kubernetes cluster dapat mengakses Pod yang berada diluar service tersebut [5].

Berikut merupakan arsitektur dari kubernetes, yang terdiri dari [6]:

- a. Master Node. Merupakan master yang berfungsi untuk mengontrol keseluruhan unit dari cluster, mengatur workload, dan komunikasi antar sistem. Berikut ini merupakan komponen yang ada pada master node:
 1. Etcd. Merupakan storage untuk menyimpan data cluster kubernetes.
 2. API server. Merupakan penyedia antarmuka internal ataupun eksternal pada kubernetes.
 3. Scheduler. Merupakan komponen pengatur jadwal untuk node dan pod, berfungsi untuk

memastikan beban kerja tidak melebihi sumber daya yang ada.

4. Controller manager. Berfungsi untuk memonitor cluster agar sesuai dengan konfigurasi yang diinginkan.

b. Worker Node. Sesuai namanya, worker node merupakan sistem pekerja atau minion yang menjadi tempat mesin container (beban kerja) digunakan. Berikut adalah komponen yang ada pada worker node [6]:

1. Kubelet. Berfungsi untuk memastikan pod beroperasi pada master node.
2. Kube – proxy. Berfungsi meneruskan koneksi dan mengatur jaringan serta membantu abstraksi layanan kubernetes.
3. Container runtime. Merupakan komponen yang bertanggung jawab untuk menampung aplikasi pada container serta menjalankan container.
4. Docker merupakan salah satu container organizer yang dapat digunakan dalam layanan Kubernetes.

Kubernetes memiliki beberapa objek dasar, diantaranya [6]:

1. Pod. Merupakan unit yang terkecil dan sederhana yang ada pada Kubernetes. Berfungsi untuk menjalankan docker images pada container.
2. Service. Merupakan abstraksi layanan kubernetes yang berfungsi untuk mengarahkan request ke beberapa pod dengan IP Address.
3. Volume. Objek yang berfungsi sebagai media penyimpanan data suatu container.
4. Namespace. Merupakan objek yang berfungsi untuk memisahkan resource dari cluster.
5. Node. Objek ini merupakan mesin pekerja, bisa berupa virtual machine atau mesin fisik. Setiap node dapat berisi banyak pod serta master.

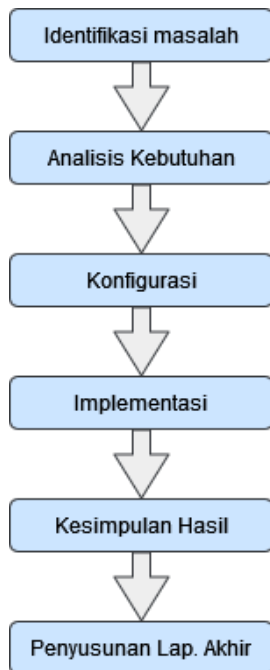
1.4 Wiki.js

Saat mendengar kata “wiki”, kebanyakan orang akan mengingat atau menyebutkan Wikipedia, yang merupakan ensiklopedia online yang mana penggunaannya dapat memberikan kontribusi pemikiran dan pengetahuan mereka ke setiap halaman Wikipedia. Padahal wiki adalah salah satu jenis aplikasi web yang merupakan halaman web dimana para pengguna halaman web tersebut dapat mengorganisir, mengedit, ataupun meninjau isi dari halaman web dengan mudah dan juga dapat berkolaborasi dengan pengguna lainnya dan masih ada banyak aplikasi wiki selain Wikipedia. Aplikasi wiki ini diciptakan oleh Ward Cunningham di tahun 1994. Sampai saat ini ada beberapa aplikasi wiki yang dijalankan dalam berbagai bahasa pemrograman yang termasuk diantaranya aplikasi wiki *open-source* yang berjalan dari bahasa pemrograman Javascript

seperti Wiki.js. Wiki.js merupakan salah satu perangkat lunak wiki berkemampuan modern yang didukung oleh Node.js, Markdown, dan Git, dikembangkan oleh Nicolas Giard. Dengan rilis awal di bulan September 2016, Wiki.js menjadi perangkat lunak termuda dari aplikasi Wiki [7].

2. Metode Penelitian

Dalam pengimplementasian penyebaran aplikasi Wiki.js menggunakan Kubernetes, penulis membagi penelitian ini kedalam beberapa tahapan.



Gambar 1. Tahap Penelitian

2.1 Identifikasi Masalah

Dalam tahapan ini identifikasi dilakukan terhadap temuan yang kemudian dianalisa sehingga dalam tahap ini menghasilkan permasalahan berupa bagaimana mengimplementasikan penggunaan teknologi *kubernetes cluster google cloud platform* untuk menyebarkan aplikasi *wiki.js*.

2.2 Analisis Kebutuhan

Pada tahap analisis kebutuhan berupa tahap dimana menentukan alat dan bahan yang akan digunakan. Berdasar dari identifikasi masalah yang telah dilakukan, penelitian ini menggunakan *Google Kubernetes Engine (GKE)* versi 1.25.8-gke.500 yang merupakan versi bawaan atau *default* dari yang ada pada platform Google. Selain itu kebutuhan lainnya merupakan basis data. Basis data yang digunakan adalah instance *cloud SQL* yang merupakan salah satu layanan yang ada di GCP dengan menggunakan PostgreSQL versi 14. Basis data tersebut diperuntukan untuk penyimpanan data dari

aplikasi *Wiki.js* yang akan di-*deploy* dengan menggunakan arsitektur Kubernetes.

2.3. Konfigurasi Sistem

Pada tahap konfigurasi sistem merupakan tahap dimana men-setting secara teknis seluruh pendukung dalam arsitektur Kubernetes yang akan digunakan untuk menyebarkan atau men-*deploy* aplikasi *wiki.js* agar sesuai dengan yang diinginkan. Dalam konfigurasi ini hal yang perlu dikonfigurasi yaitu seperti konfigurasi jaringan pada SQL instance ke Kubernetes Cluster, serta konfigurasi pada Kubernetes Cluster.

2.4 Implementasi

Dari tahapan – tahapan yang sudah dilakukan sebelumnya, tahap ini merupakan pengimplementasian deployment aplikasi *wiki.js* dengan menggunakan arsitektur Kubernetes Google Cloud Platform yang telah dikonfigurasi sehingga menghasilkan fungsionalitas yang diinginkan.

2.5 Kesimpulan Hasil

Pada tahap ini merupakan kesimpulan hasil dari penelitian yang didasari oleh bekerja atau tidaknya fungsionalitas aplikasi yang di-*deploy* tersebut.

2.6 Penyusunan Laporan Akhir

Tahapan terakhir merupakan pembuatan laporan mengenai hasil penelitian secara tertulis dan berlandaskan dengan kajian – kajian teoritis.

3. Hasil dan Pembahasan

Dari tahapan penelitian yang sudah dilakukan dimulai dari identifikasi masalah, analisis kebutuhan, konfigurasi perangkat, implementasi, serta hasil dan pembahasan yang telah didapatkan dapat dilihat sebagai berikut.

3.1 Konfigurasi Sistem

Pada gambar 2 dan 3 dapat dilihat hasil konfigurasi untuk membuat Kubernetes Cluster. Pembuatan kubernetes cluster pada Google Cloud Platform dilakukan menggunakan menu yang ada pada *interface* Google Cloud Platform, mesin kubernetes dibuat dengan memasukkan nama cluster lalu memilih zona sesuai yang diinginkan, pada penelitian ini zona bawaan yang digunakan adalah zona *asia-southeast2*.

Cluster basics

Create an Autopilot cluster by specifying a name and region. After the cluster is created, you can deploy your workload through Kubernetes and we'll take care of the rest, including:

- ✓ **Nodes:** Automated node provisioning, scaling, and maintenance
- ✓ **Networking:** VPC-native traffic routing for public or private clusters
- ✓ **Security:** Shielded GKE Nodes and Workload Identity
- ✓ **Telemetry:** Cloud Operations logging and monitoring

Name
autopilot-cluster-1

Cluster names must start with a lowercase letter followed by up to 39 lowercase letters, numbers, or hyphens. They can't end with a hyphen. You cannot change the cluster's name once it's created.

Region
asia-southeast2

The regional location in which your cluster's control plane and nodes are located. You cannot change the cluster's region once it's created.

NEXT: NETWORKING

RESET SETTINGS

Gambar 2. Konfigurasi Kubernetes Cluster

Cluster basics

Name	autopilot-cluster-1	
Location type	Regional	
Region	asia-northeast2	
Default node zones	asia-northeast2-b asia-northeast2-c asia-northeast2-a	
Release channel	Regular channel	UPGRADE AVAILABLE
Version	1.25.8-gke.500	
External endpoint	34.97.196.69	
Internal endpoint	10.174.0.2	

Gambar 3. Hasil konfigurasi Kubernetes Cluster

Selanjutnya, secara otomatis kubernetes cluster yang telah dibuat akan memiliki sebuah IP public nya sendiri, dalam penelitian ini IP Public yang dimiliki adalah 34.97.196.69.

Instance info

Instance ID
wikijs-db

PASSWORD POLICY

Database version
PostgreSQL 14

Database version upgrades are available. [UPGRADE](#)

Choose region and zonal availability

For better performance, keep your data close to the services that need it. Region is permanent, while zone can be changed any time.

Region

asia-southeast2 (Jakarta)

Summary

Region	asia-southeast2
DB Version	PostgreSQL 14
vCPUs	2 vCPU
Memory	8 GB
Storage	100 GB
Network throughput (MB/s)	500 of 2,000
Disk throughput (MB/s)	Read: 48.0 of 2,000 Write: 48.0 of 1,000
IOPS	Read: 3,000 of 10,000 Write: 3,000 of 10,000
Connections	Public IP
Backup	Automated
Availability	Single zone
Point-in-time recovery	Enabled

Gambar 4. Konfigurasi SQL Instance

Selanjutnya dapat dilihat pada gambar 4 merupakan pembuatan dari SQL instance yang akan menyimpan data aplikasi dari wiki.js. Cara pembuatan SQL instance di GCP dapat dicari melalui menu layanan yang ada di GCP. SQL instance yang digunakan merupakan PostgreSQL dengan versi 14 dengan spesifikasi standar yang memiliki kapasitas storage sebesar 100 GB, dengan 2 core virtual CPU, serta memiliki memori sebesar 8 GB dan berada di zona *asia-southeast2*.

SUMMARY NETWORKING SECURITY CONNECTIVITY TESTS

Choose how you want your source to connect to this instance, then define which networks are authorized to connect. [Learn more](#)

You can use the Cloud SQL Proxy for extra security with either option. [Learn more](#)

Instance IP assignment

☐ Private IP

Assigns an internal, Google-hosted VPC IP address. Requires additional APIs and permissions. Can't be disabled once enabled. [Learn more](#)

☒ Public IP

Assigns an external, internet-accessible IP address. Requires using an authorized network or the Cloud SQL Proxy to connect to this instance. [Learn more](#)

Authorized networks

You can specify CIDR ranges to allow IP addresses in those ranges to access your instance. [Learn more](#)

kost-ip (125.165.0.0/16)

gke-1 (34.97.0.0/16)

ip-hotspot (182.4.0.0/16)

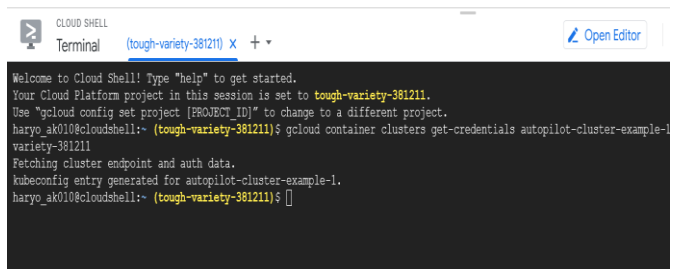
ADD A NETWORK

Gambar 5. Konfigurasi jaringan pada instance SQL

Setelah itu, dilakukan sebuah konfigurasi jaringan di SQL instance yang telah dibuat. Pada gambar 5, hal ini dilakukan agar kubernetes cluster memiliki akses terhadap database SQL instance yang mana dimaksudkan untuk penyimpanan data dari aplikasi wiki.js. Konfigurasi dilakukan dengan cara memasukkan sub IP dari kubernetes cluster yang dimiliki yaitu 34.97.0.0/16.

3.2 Implementasi

Pada tahapan implementasi ada beberapa hal yang perlu dibuat, dalam penggunaan kubernetes cluster pada penelitian ini digunakan terminal untuk membuat file konfigurasi aplikasi dengan bentuk *yaml* file dan nanti konfigurasi tersebut akan di-*apply* pada mesin kubernetes.



Gambar 6. Membuat namespace dengan nama wikijs

Hal-hal yang perlu dilakukan untuk men-*deploy* aplikasi wiki.js adalah, yang pertama membuat sebuah *namespace* dengan nama wikijs melalui terminal dengan memasukkan perintah seperti pada gambar 7. Pembuatan *namespace* ini nantinya bertujuan untuk memisahkan sumberdaya cluster.

```
$ kubectl create namespace wikijs
$ kubectl get namespace
NAME                                STATUS AGE
default                            Active 10h
gke-gmp-system                      Active 10h
```

```
gke-managed-filestorecsi Active 10h
gmp-public Active 10h
kube-node-lease Active 10h
kube-public Active 10h
kube-system Active 10h
wikijs Active 22s
```

Gambar 7. Melihat hasil namespace

Hasil dari perintah pada gambar 7 memperlihatkan list namespace apa saja yang telah dibuat, dan dapat dilihat bahwa namespace wikijs telah dibuat.

Selanjutnya dibuat sebuah file berekstensi *yaml* lain yang diberi nama *configMap*, dalam file tersebut isinya merupakan konfigurasi untuk menghubungkan aplikasi ke database SQL instance dengan jenis PostgreSQL yang dimiliki. Perintah untuk membuat *configMap* file dapat dilihat pada gambar 8.

```
$ nano wikijs-config.yaml
apiVersion: v1
data:
  DB_HOST: "34.101.177.111"
  DB_PORT: "5432"
  DB_USER: "postgres"
  DB_TYPE: "postgres"
  DB_NAME: "postgres"
kind: ConfigMap
metadata:
  name: postgres-config
  namespace: wikijs
```

Gambar 8. Isi file konfigurasi *ConfigMap*

Pada gambar 8 terlihat isi konfigurasi berupa beberapa variabel. Jenis atau *kind* dari file *yaml* tersebut berjenis *ConfigMap*, hal ini perlu diperhatikan sebab untuk meng-*apply* konfigurasi diperlukan jenis (*kind*) yang sesuai agar mesin dapat mengenali file konfigurasi tersebut dibuat untuk konfigurasi berjenis apa. Pada variabel *DB_HOST* diisi dengan IP address dari SQL instance yang telah dibuat sebelumnya, yaitu '34.101.177.111'. Pada variabel *DB_PORT* merupakan port *default* atau bawaan dari PostgreSQL yaitu '5432'. Pada variabel *DB_USER* diisi dengan nama dari SQL apa yang kita pakai, yaitu 'postgres', begitu juga dengan variabel *DB_TYPE* diisi dengan value yang sama. Lalu pada variabel *DB_NAME* diisi dengan nama database yang ada di SQL instance yang telah dibuat.

```
$ kubectl apply -f wikijs-config.yaml
configmap/postgres-config created

$ kubectl get configmaps -n wikijs
NAME          DATA  AGE
postgres-config 5      13s
```

Gambar 9. Melihat hasil konfigurasi *ConfigMap*

Setelah file konfigurasi selesai, pada gambar 9 merupakan perintah untuk meng-*apply* file konfigurasi

dan perintah untuk memperlihatkan hasil dari *configMap* yang telah dibuat.

```
$ nano wikijs-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: postgres
  namespace: wikijs
type: Opaque
stringData:
  password: keamanan
```

Gambar 10. Melihat hasil konfigurasi *Secret*

```
$ kubectl apply -f wikijs-secret.yaml
secret/postgres created

$ kubectl get secret -n wikijs
```

NAME	TYPE	DATA	AGE
postgres	Opaque	1	18s

Gambar 11. Melihat hasil konfigurasi *Secret*

Setelah membuat file konfigurasi *configMap*, akan dibuat file konfigurasi berjenis *Secret*. Pada gambar 10 merupakan perintah untuk membuat file konfigurasi. *Secret* berguna untuk menyimpan dan mengatur informasi yang tidak untuk diekspos, seperti halnya kata sandi. Pada gambar 11 perintah untuk meng-*apply* serta melihat bahwa konfigurasi *secret* telah terbuat.

```
$ nano wikijs-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: wikijs
  namespace: wikijs
spec:
  type: LoadBalancer
  selector:
    app: wikijs
  ports:
    - protocol: TCP
      Port: 80
      targetPort: 3000
```

Gambar 12. Melihat hasil konfigurasi *Service*

```
$ kubectl apply -f wikijs-service.yaml
$ kubectl get svc -n wikijs
NAME      TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
wikijs    LoadBalancer  10.50.0.133    34.97.103.22   80:30624/TCP 30h
```

Gambar 13. Melihat hasil konfigurasi *Service*

Pada gambar 12 merupakan aktivitas untuk membuat sebuah file konfigurasi berjenis *service*. Penggunaan *service* bertujuan untuk mengarahkan *request* atau *traffic* ke beberapa pod menggunakan IP address. Dalam penelitian ini tipe *service* yang digunakan adalah *LoadBalancer* dengan protokol TCP, dan port 80, serta

targetport 3000. Selanjutnya pada gambar 13 merupakan perintah untuk meng-*apply* file konfigurasi dan melihat bahwa *service* telah berjalan.

```
$ nano wikijs-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: wikijs
    name: wikijs
    namespace: wikijs
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wikijs
  template:
    metadata:
```

```
      labels:
        app: wikijs
      spec:
        containers:
          - name: wikijs
            Image:
lsr.io/linuxserver/wikijs:latest
            env:
              - name: DB_HOST
                valueFrom:
                  configMapKeyRef:
                    key: DB_HOST
                    name: postgres-config
              - name: DB_PORT
                valueFrom:
                  configMapKeyRef:
                    key: DB_PORT
                    name: postgres-config
              - name: DB_USER
                valueFrom:
                  configMapKeyRef:
                    key: DB_USER
                    name: postgres-config
              - name: DB_TYPE
                valueFrom:
                  configMapKeyRef:
                    key: DB_TYPE
                    name: postgres-config
              - name: DB_NAME
                valueFrom:
                  configMapKeyRef:
                    key: DB_NAME
                    name: postgres-config
              - name: DB_PASS
                valueFrom:
                  secretKeyRef:
                    name: postgres
                    key: password
        ports:
          - containerPort: 3000
```

protocol: TCP

Gambar 14. Melihat hasil konfigurasi *Deployment*

Kemudian tahapan selanjutnya yaitu membuat file konfigurasi berjenis *deployment*. Dalam *deployment* ini akan berisikan dengan jelas aplikasi apa yang akan di-*deploy*, serta deskripsi dari beberapa state. Pada gambar 14 dituliskan berdasarkan templat yang ada pada dokumentasi kubernetes. Pada kolom container merupakan kolom nama yang diisi nilai 'wikijs', lalu pada kolom image diisikan value dari image aplikasi wikijs yaitu 'lsr.io/linuxserver/wikijs: latest'.

Selanjutnya pada kolom *environment* terdapat beberapa kolom dibawahnya yang berisi beberapa variabel. Pada saat pembuatan file konfigurasi *configMap*, telah dimasukkan beberapa data seperti DB_HOST, DB_PORT, DB_USER, DB_TYPE, dan DB_NAME. Konfigurasi tersebut ditulis dengan maksud agar *deployment* dapat mengambil nilai dari (valueFrom) *configMap* yang telah dibuat sebelumnya. Dengan mengambil referensi dari *ConfigMap* salah satu contohnya yaitu, kata kunci (key) DB_HOST akan mengambil data pada variabel DB_HOST di *ConfigMap*, yang dinamai dengan nama 'postgres-config'.

```
$ kubectl apply -f wikijs-deployment.yaml
$ kubectl get deploy -n wikijs
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
wikijs    1/1     1             1           4h15m

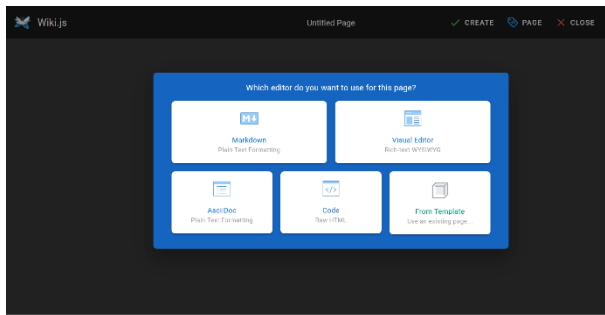
$ kubectl get pods -n wikijs
NAME                                READY   STATUS    RESTARTS   AGE
wikijs-9b967f595-t5m5w             1/1     Running   0          4h17m
```

Gambar 15. Melihat hasil konfigurasi *Deployment*

Setelah konfigurasi file *deployment* selesai, pada gambar 15 dilakukan *apply* konfigurasi dan melihat hasil *deployment* yang sudah berjalan. Dengan berjalannya konfigurasi *deployment*, ini juga akan menjalankan pod.

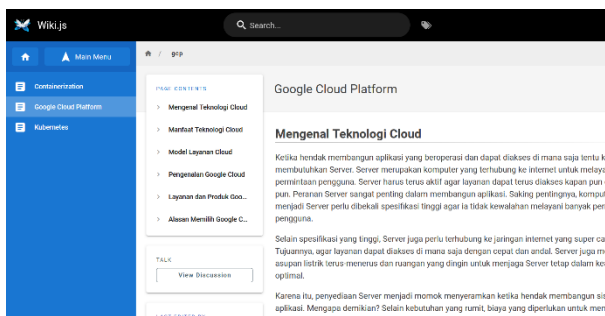
```
$ kubectl logs -f wikijs-9b967f595-t5m5w -n wikijs
[custom-init] No custom files found, skipping...
Loading configuration from /app/wiki/config.yml... OK
2023-06-07T01:05:22.735Z[MASTER]info:=====
2023-06-07T01:05:22.738Z[MASTER]info: Wiki.js 2.5.299 =====
2023-06-07T01:05:22.739Z[MASTER]info:=====
2023-06-07T01:05:22.739Z[MASTER]info: Initializing...
2023-06-07T01:05:25.453Z[MASTER]info: Using database driver pg for postgres [ OK ]
```


ke *dashboard* dan beberapa pilihan menu kostumisasi web wiki.js.



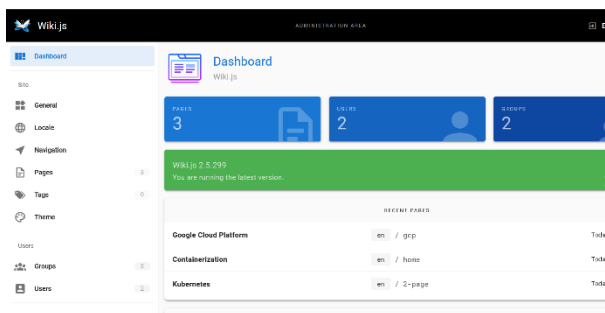
Gambar 21. Halaman awal pembuatan konten wiki.js

Untuk membuat halaman konten pada web berbasis wiki.js, disediakan beberapa editor pilihan dalam kostumisasi konten, diantaranya seperti *markdown*, *visual editor*, *ascii doc*, *code*, dan *from template*. Pada penelitian ini kostumisasi konten pada web berbasis wiki.js akan menggunakan *visual editor* dikarenakan dapat mengedit secara mudah dan fleksibel untuk percobaannya.



Gambar 22. Halaman hasil pembuatan konten wiki.js

Pada gambar 22 merupakan hasil dari pembuatan halaman website wiki.js yang telah dikostumisasi. Dalam penelitian ini website tersebut berisi beberapa konten untuk memberikan suatu informasi seperti apa itu Kubernetes, GCP, dan Kontainerisasi, hal ini merupakan percobaan untuk mengisi konten pada website.



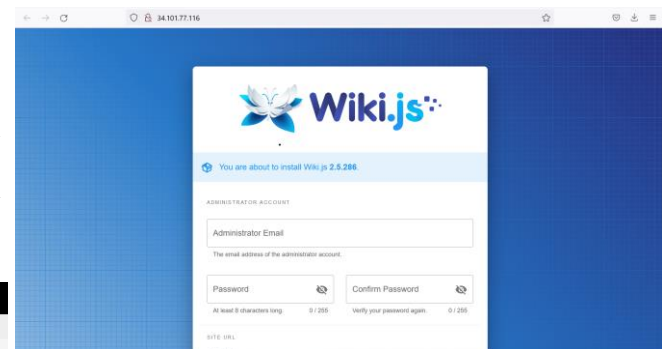
Gambar 23. Halaman *dashboard* wikis

Selanjutnya pada gambar 23 merupakan tampilan pada dashboard untuk manajemen konten dan halaman pada website berbasis wiki.js.

Hasil yang didapatkan setelah melakukan semua tahapan yang ada pada tahap implementasi, aplikasi wiki.js dapat berjalan pada kubernetes cluster dengan mengakses alamat IP yang didapat dari service. Seperti pada gambar 24, alamat IP yang tertera dapat dibuka pada browser.

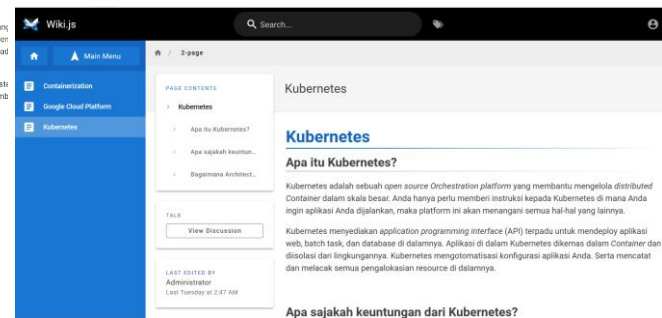
Filter Is system object: False Filter services and ingresses						
☐	Name	Status	Type	Endpoints	Pods	Namespace
☐	wikis	OK	External load balancer	34.97.103.22:80	1/1	wikis

Gambar 24. Melihat IP address pada service



Gambar 25. Halaman pendaftaran dan instalasi wiki.js

Lalu, pada gambar 25 menampilkan hasil dari alamat IP yang dibuka pada browser.



Gambar 26. Hasil kustomisasi website wiki.js

Dengan adanya website berbasis wiki.js ini dapat membuat penggunaannya untuk berkolaborasi dalam menyunting dokumen, seperti halnya sama untuk membuat suatu blog berisi rangkuman informasi terhadap suatu topik tertentu.

4. Kesimpulan

Tujuan dari diadakannya penelitian ini didasari oleh bagaimana caranya untuk dapat menggunakan teknologi kubernetes cluster pada Google Cloud Platform untuk men-deploy aplikasi wiki.js, hal tersebut merupakan identifikasi dari masalah yang ada. Berangkat dari identifikasi masalah yang ada, selanjutnya diuraikan ke dalam beberapa tahap yang diperlukan, diantaranya analisis kebutuhan yang berupa hal – hal apa saja yang

dibutuhkan untuk mendukung penelitian ini. Kemudian tahap konfigurasi sistem hingga ke tahap implementasi merupakan hal teknis yang dilakukan dalam penelitian. Dari pembahasan yang telah dilakukan dapat ditarik kesimpulan yaitu *deployment* aplikasi wiki.js telah terimplementasikan dengan menggunakan teknologi Kubernetes pada Google Cloud Platform sehingga aplikasi website berbasis wiki.js tersebut dapat diakses dan dijalankan pada browser. Namun, pengimplementasian aplikasi ini belum dapat diakses menggunakan DNS dan hanya dapat diakses dengan IP address dari *service*. Kemudian penggunaan wiki.js tentu belum umum digunakan, tidak seperti *wordpress* yang banyak digunakan sebagai *Content Management System* bagi beberapa orang. Namun, penggunaan wiki.js bukan merupakan sebuah hal yang sulit, hanya saja dalam pengembangan aplikasi perlu adanya hal yang sesuai terhadap kebutuhan dalam penggunaannya. Diperkirakan pada penelitian ini untuk selanjutnya *deployment* aplikasi menggunakan teknologi Kubernetes dapat menggunakan *domain name server* yang disesuaikan.

Daftar Rujukan

- [1] H. Hamdan, "INDUSTRI 4.0: PENGARUH REVOLUSI INDUSTRI PADA KEWIRAUSAHAAN DEMI KEMANDIRIAN EKONOMI", *NUSAMBA*, vol. 3, no. 2, pp. 1-8, Oct. 2018, doi: <https://10.29407/nusamba.v3i2.12142>.
- [2] E. Riana, "Implementasi Cloud Computing Technology dan Dampaknya Terhadap Kelangsungan Bisnis Perusahaan Dengan Menggunakan Metode Agile dan Studi Literatur," *JURIKOM (Jurnal Riset Komputer)*, vol. 7, no. 3, p. 439, Jun. 2020, doi: <https://doi.org/10.30865/jurikom.v7i3.2192>.
- [3] A. Ashari and H. Setiawan, "Cloud Computing : Solusi ICT ?," *Cloud Computing : Solusi ICT ?*, vol. 3, no. 2, p. 130558, Oct. 2011, doi: <https://doi.org/10.36706/jsi.v3i2.736>.
- [4] D. L. F. Anissa and R. Andryani, "Penerapan Cloud Computing Dalam Aplikasi Panggil Teknisi Berbasis Android Menggunakan Google Cloud Platform," *J-SAKTI (Jurnal Sains Komputer dan Informatika)*, vol. 6, no. 2, pp. 1292–1300, Sep. 2022, Accessed: May 28, 2023. [Online]. Available: <https://tunasbangsa.ac.id/ejurnal/index.php/jsakti/article/view/533/51>.
- [5] W. Ramadhani and M. A. F. Ridha, "Perbandingan Kinerja Ingress Controller Pada Kubernetes Menggunakan Traefik Dan Nginx," *Jurnal Komputer Terapan*, vol. 8, no. 2, pp. 289–295, Dec. 2022, doi: <https://doi.org/10.35143/jkt.v8i2.5440>.
- [6] K. Yedutun, A. Noertjahyana, and H. N. Palit, "Implementasi Container Kubernetes untuk Mendukung Scalability," *Jurnal Infra*, vol. 7, no. 2, pp. 118–122, Oct. 2019, Accessed: May 28, 2023. [Online]. Available: <https://publication.petra.ac.id/index.php/teknik-informatika/article/view/8774>.
- [7] Jaroslav Šmolík, "Git-based Wiki System," M.S. Thesis, Department of Software Engineering, Czech Technical University in Prague, Prague, Czech Republic, 2018.

