METODOLOGI DESAIN PERANGKAT LUNAK PRAKTIK XII
TOOLS DEVOPS KUBERNETES

DISUSUN OLEH :

5200411007 ADI KANNATASIK
5200411010 FARIZ YUDO PRASETYO
5200411012 MARTIN SETYAWAN WIBOWO

INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS TEKNOLOGI YOGYAKARTA
2021

Daftar Isi

BAB 1 3
PENDAHULUAN 3
1.1 Latar Belakang 3
1.2 Rumusan Masalah 3
1.3 Tujuan 3
BAB 2 4
PEMBAHASAN 4
2.1 Pengertian 4
2.2 Alasan Menggunakan Kubernetes 4
2.3 Komponen 4
1) Cluster 4
2) Object 6
2.4 Kelebihan Kubernetes 6
BAB 3 8
PENGGUNAAN TOOLS DEVOPS 8
3.1 Judul : 8
3.2 Penulis : 8
3.3 Nama Jurnal : 8
Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer 8
3.4 Tahun, Halaman : Vol. 4, No. 1, Januari 2020, hlm. 100-108 8
3.5 Tujuan : 8
3.6 Perancangan Sistem : 8
3.7 Alur Sistem 9
3.8 Perancangan dan Pengujian 11
1) Perancangan Pengujian Skalabilitas 12
2) Perancangan Pengujian Self-Healing 12
3) Perancangan Pengujian Persistent Pod Database 12
3.9 Pengujian 12
3.10 Hasil dan Kesimpulan 13
BAB 4 13
PENUTUP 13

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Kubernetes / K8s adalah platform open source untuk mengelompokkan kontainer yang membentuk suatu aplikasi dalam bentuk unit logis yang memudahkan proses manajemen dan discovery. Kubernetes dibuat berdasarkan pengalaman operasional workloads skala production yang dilakukan oleh Google, yang digabungkan dengan ide-ide terbaik dan best practices yang diberikan oleh komunitas. Platform ini pertama kali dikembangkan oleh Google dan kini dikelola oleh Cloud Native Computing Foundation (CNCF) sebagai platform manajemen kontainer yang cukup populer.

Kontainer sendiri adalah environment dengan sumber daya, CPU, dan sistem file untuk satu aplikasi. Jadi, aplikasi tersebut akan memiliki sumber daya sendiri. Keuntungannya, aplikasi jadi tidak mudah mengalami downtime. Kubernetes memiliki kemampuan untuk melakukan penjadwalan aplikasi, load balancing server dan peningkatan kapasitas kontainer secara otomatis.

Tak heran, Kubernetes kini banyak digunakan untuk membangun microservices, yaitu aplikasi kecil yang menjadi pengembangan dari aplikasi besar dan saling terhubung satu sama lain. Dengan menggunakan Kubernetes, proses pengembangan aplikasi jadi lebih cepat karena proses scale up aplikasi tidak dibuat sekaligus seperti pada pendekatan monolith.

Beberapa perusahaan yang menggunakan microservices pada produk mereka di antaranya adalah Netflix, Amazon dan Apple. Inilah yang membuat Kubernetes menjadi semakin populer berkat kebutuhan penggunaan pada aplikasi modern.

1.2 Rumusan Masalah

1. Apa itu Kubernetes ?
2. Mengapa Menggunakan Kubernetes ?
3. Komponen - Komponen Kubernetes ?
4. Apa kelebihan Kubernetes ?
5. Perbandingan dengan metode Waterfall, Prototype, RAD, Scrum, Agile, DevOps ?

1.3 Tujuan

1. Mengetahui apa itu Kubernetes
2. Mengetahui Alasan Menggunakan Kubernetes
3. Mengetahui Komponen-komponen apa saja yang ada pada Kubernetes
4. Mengetahui kelebihan kubernetes
5. Mengetahui Perbandingan DevOps dengan metode lainya

BAB 2
PEMBAHASAN

2.1 Pengertian

Kubernetes merupakan platform open-source yang digunakan untuk untuk mengotomatisasi penerapan, penskalaan, dan manajemen aplikasi komputer, serta menyediakan konfigurasi dan otomatisasi secara deklaratif. Kubernetes berada di dalam ekosistem yang besar dan berkembang cepat. Service, support, dan tools Kubernetes tersedia secara meluas.

2.2 Alasan Menggunakan Kubernetes

Alasan kita menggunakan Kubernetes karena Kubernetes bisa melakukan penjadwalan seta menjalankan container pada aplikasi di kelompok mesin virtual juga fisik. Disamping itu Keuntungan juga manfaat yang didapatkan dari aplikasi ini adalah bisa melekat secara penuh di container. Kubernetes ini menyediakan infrastruktur demi membantu membangun berbagai lingkungan untuk pengembangan dalam kontainer-sentris.

Lebih jelasnya sekarang ini banyak muncul teknologi baru, salah satunya teknologi container. Penggunaan teknologi ini bisa membuat semua aplikasi yang dimiliki mampu dikemas dengan demikian saat akan dijalankan kapanpun dan dimanapun aplikasi tersebut bisa berjalan seperti biasanya atau sama dengan ketika kita mencobanya.

Disini Kita sebagai pengguna sudah tidak perlu menginstall banyak server yang di dalamnya ada banyak environment, aplikasi, juga kebutuhan yang lain, sebab semuanya telah dibutuhkan oleh aplikasi sudah dipasang pada container. Agar container bisa berjalan, tentu butuh adanya aplikasi. Sedangkan pada umumnya aplikasi yang digunakan adalah Docker. Aplikasi ini di install pada server, lalu dijalankan di docker image maka terbentuknya container serta aplikasi yang sudah berjalan serta dapat digunakan.

2.3 Komponen

1) Cluster

Cluster adalah suatu kelompok berisi server fisik atau VPS untuk menjalankan Kubernetes. Ada dua jenis server yang dibutuhkan, yaitu master node dan worker node.

I. Master Node

Master node adalah server utama yang mengatur semua operasi cluster menggunakan tiga komponen, yaitu kube-apiserver, kube-controller-manager, kube-scheduler dan etcd. Dan berikut adalah penjelasan fungsinya.

a. kube-apiserver:

validasi dan konfigurasi data untuk objek API, yaitu pod, services, volume, dan lainnya. Komponen control plane yang mengekspos API

Kubernetes. Merupakan front-end dari control plane Kubernetes. Komponen ini didesain agar dapat diskalakan secara horizontal.

b. kube-controller-manager:

melakukan monitor cluster agar sesuai dengan konfigurasi data objek di dalam node. Komponen control plane jugalah yang menjalankan pengontrol. Secara logis, setiap pengontrol adalah sebuah proses yang berbeda, tetapi untuk mengurangi kompleksitas, kesemuanya dikompilasi menjadi sebuah biner/(binary) yang dijalankan sebagai satu proses. Kontroler-kontroler ini meliputi :

- **Kontroler Node :** Bertanggung jawab untuk mengamati dan memberikan respons apabila jumlah node berkurang.
- **Kontroler Replikasi :** Bertanggung jawab untuk menjaga jumlah pod agar jumlahnya sesuai dengan kebutuhan setiap objek kontroler replikasi yang ada di sistem.
- **Kontroler Endpoints :** Menginisiasi objek Endpoints (yang merupakan gabungan Pods dan Services).
- **Kontroler Service Account & Token :** Membuat akun dan akses token API standar untuk setiap namespaces yang dibuat.

c. kube-scheduler:

menambah objek baru ke node. Misalnya, menginstall pod ke node tertentu.Komponen control plane ini bertugas mengamati Pod baru yang belum ditempatkan di node manapun dan kemudian memilihkan Node di mana Pod baru tersebut akan dijalankan. Faktor-faktor yang perlu dipertimbangkan untuk keputusan penjadwalan termasuk: kebutuhan sumber daya secara individual dan kolektif, batasan perangkat keras/perangkat lunak/peraturan, spesifikasi afinitas dan nonafinitas, lokalisasi data, interferensi antar beban kerja dan tenggat waktu.

d. Etcd:

ruang penyimpanan key value konfigurasi data cluster / penyimpanan key value konsisten yang digunakan sebagai penyimpanan data klaster Kubernetes. Selalu perhatikan mekanisme untuk membackup data etcd pada klaster Kubernetes kamu.

II. Worker Node

Worker node adalah semua server non master yang berfungsi untuk menjalankan dua komponen, yaitu kubelet dan kube-proxy. Begini penjelasan fungsi komponennya:

- **Kubelet :** komponen untuk memastikan kontainer beroperasi di dalam objek Pod.
- **Kube-proxy :** memelihara network rules dan meneruskan koneksi ke suatu host.
- **Docker image :** file dari aplikasi Docker yang berfungsi untuk membuat kontainer.

2) Object

Di dalam sebuah cluster, terdapat berbagai object, yaitu entitas yang merepresentasikan kondisi dari suatu cluster. Ada berbagai object yang ada pada sebuah cluster Kubernetes, yaitu:

a. Pod

Pod merupakan objek terkecil di dalam cluster kubernetes yang terletak di dalam node. Fungsinya untuk menjalankan docker images yang membentuk sebuah kontainer.

b. Service

Service adalah objek yang digunakan untuk mengarahkan request atau traffic ke beberapa Pod menggunakan IP address. Tujuannya agar Pod bisa diakses dari luar.

c. Volume

Volume adalah objek yang berfungsi untuk penyimpanan data suatu kontainer. Letaknya di luar kontainer. Misalnya, awsElasticBlockStore, azureDisk, azureFile, cephfs, cinder, configMap.

d. Namespace

Namespace adalah objek untuk memisahkan resource atau environment cluster. Dengan namespace, Anda dapat memisahkan tiap cluster project supaya tidak saling terganggu satu sama lain.

2.4 Kelebihan Kubernetes

- **Service Discovery dan Load Balancing**

  Fitur service discovery memudahkan untuk melacak kontainer dengan otomatis. Hal ini tentu sangat penting di dalam mengembangkan aplikasi microservices. Kubernetes dapat mengenali sebuah service berdasar DNS atau IP address server tersebut.

  Tak hanya itu, berkat Load Balancing pengelolaan trafik menjadi lebih mudah. Saat terjadi trafik yang cukup besar, Kubernetes mampu membagi beban secara merata sehingga membuat aplikasi menjadi lebih stabil.

  Dengan kata lain Kubernetes bisa mengekspose container yang digunakan dengan nama DNS atau IP address server sendiri. Ketika trafik berada di container besar, aplikasi ini secara otomatis bisa melakukan load balancing trafik atau penyeimbang beban serta distribusi trafik dengan demikian aplikasi yang dijalankan bisa stabil.

- **Storage Orchestration**

  Storage orchestration. Aplikasi ini dapat juga me-mount di system storage yang digunakan, misalkan storage, local storage dari cloud provider misalkan alicloud dan aws. Itu karena kubernetes memungkinkan kita

melakukan mount pada media penyimpanan (storage) pilihan seperti, storage lokal atau yang berbasis cloud seperti AWS dan lainnya.

- **Automated Rollouts and Rollbacks**

  Deployment dapat menggunakan jenis file YAML. Dengan demikian bisa mendeskripsikan dahulu deployment yang diinginkan. Kita dapat menggunakan jenis file YAML agar dapat melakukan deployment lain serta untuk merubah deployment yang sudah ada. Lebih dalam lagi kita bisa melaksanakan rollback dengan menggunakan fike YAML dahulu. Dengan demikian semuanya bisa terdokumentasi serta bisa dilakukan secara otomatis.

  Fitur automated rollouts dan rollback sangat penting untuk membantu kita terkait deployment. Pada Kubernetes, deployment bisa menggunakan file YAML yang didalamnya nanti akan berisi ReplicaSet. Apabila terjadi kendala pada deployment, dan kita masih memiliki ReplicaSet dari versi sebelumnya, kita dapat menggunakannya untuk melakukan rollback. Hal tersebut juga bisa Anda gunakan ketika Anda akan melakukan deployment lainnya.

- **Automatic Bin Packing**

  Menggunakan fitur Automatic Bin Packing, Anda bisa mengatur kapasitas CPU dan sumber daya tiap kontainer secara spesifik. Jadi saat limit kapasitas sudah ditentukan, aplikasi terhindar dari berebut sumber daya. Selain itu, sumber daya jadi lebih hemat.

- **Self Healing**

  Fitur self-healing sangat penting bagi kita yang ingin terus menjalankan aplikasi selama 24 jam. Dan Kubernetes memiliki kemampuan untuk memeriksa kontainer yang ada, apakah dalam keadaan running (berjalan) atau mengalami error.

  Jika terjadi kendala pada kontainer, Kubernetes akan menghentikan proses yang berjalan dan memberikan opsi untuk merestart atau mengganti secara otomatis kontainer yang error tersebut. Menariknya, trafik tidak akan diarahkan ke kontainer yang rusak sebelum siap untuk menerima request kembali.

- **Secret and Configuration Management**

  Kubernetes memungkinkan Kita menyimpan data-data sensitif seperti password, auth token hingga SSH keys ke dalam Kubernetes Secret. Hal ini tentu jauh lebih aman dibanding menyimpannya di dalam container image.

  Secret bisa dibuat oleh sistem atau oleh user, yaitu Anda sendiri. Nah, karena secret sendiri secara default disimpan tanpa terenkripsi, Anda bisa melakukan konfigurasi lebih lanjut sesuai dengan kebutuhan,antara lain opaque (secret umum/default), docker registry (untuk otentikasi docker registry), dan TLS (untuk penggunaan dengan public/private keys).

BAB 3
PENGGUNAAN TOOLS DEVOPS

3.1 Judul :

Pengembangan Infrastruktur Analisis Data Heart Rate berbasis Microservices menggunakan Kubernetes

3.2 Penulis :

- Abd. Jahiduddin
- Eko Sakti Pramukantoro
- Fariz Andri Bakhtiar

3.3 Nama Jurnal :

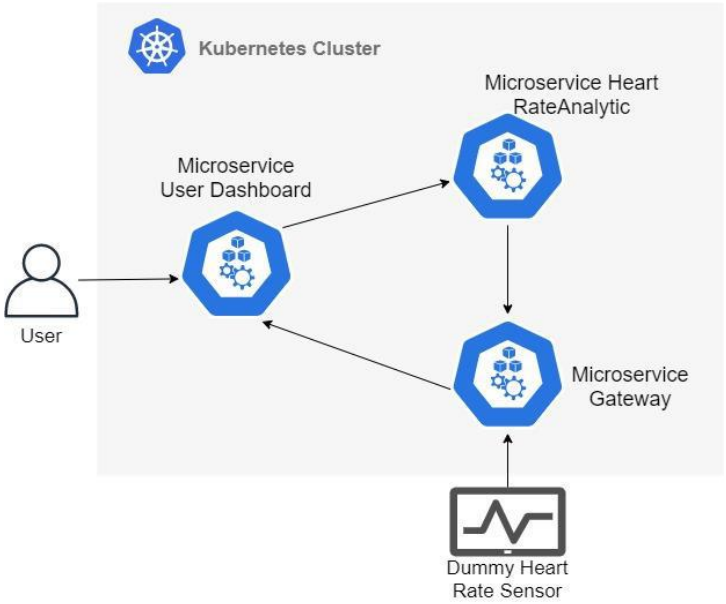
Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer

3.4 **Tahun, Halaman :** Vol. 4, No. 1, Januari 2020, hlm. 100-108

3.5 Tujuan :

Tujuan pada penelitian ini adalah untuk Mengembangkan infrastruktur analisis data heart rate menggunakan arsitektur microservices, Docker container dan Kubernetes guna menyelesaikan permasalahan meningkatnya jumlah perangkat Internet of Things yang menyebabkan peningkatan volume data dalam jumlah yang sangat besar, dan data tersebut tidak akan berguna jika tidak dilakukan pemrosesan data, lalu kesulitan dalam hal penggunaan alat untuk analisis, serta proses pemeliharaan dan pengembangan yang rumit kebutuhan sumber daya dalam pemrosesan data, proses instalasi yang kompleks.

3.6 Perancangan Sistem :



Keterangan :

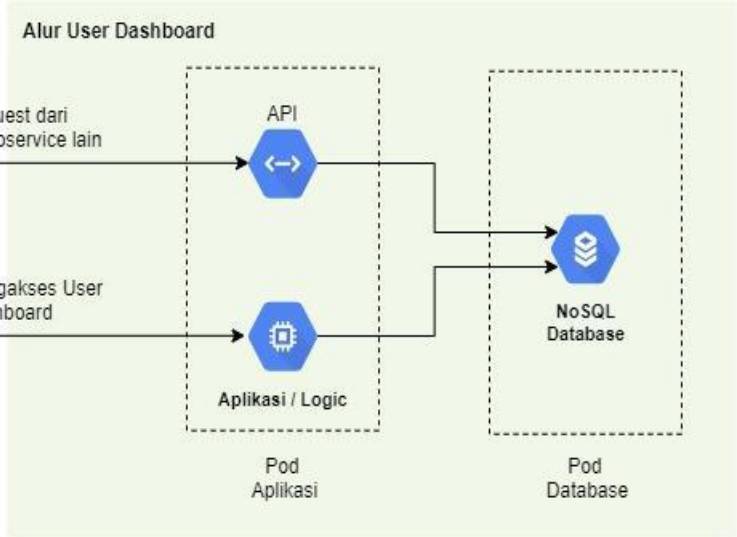
1) Terdapat tiga microservices yaitu user dashboard, heart rate analytic, dan gateway. Setiap microservices dideploy ke pod yang berbeda.
2) Pengguna mengakses microservice user dashboard untuk mengakses keseluruhan sistem melalui web.
3) Sensor mengirimkan data ke microservice gateway.
4) Setiap microservices dapat saling berkomunikasi melalui API, untuk penjelasan lebih lengkap mengenai alur komunikasi dapat dilihat pada

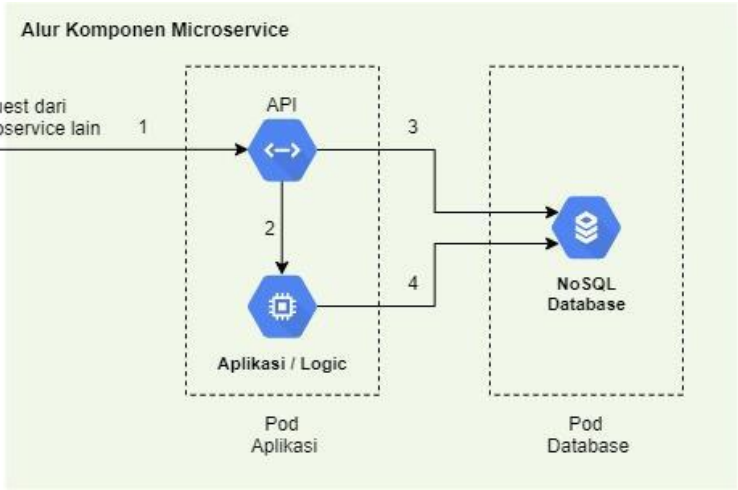
3.7 Alur Sistem

Alur sistem pada penelitian ini dibagi menjadi dua bagian yaitu alur sistem antar microservices dan alur sistem antara komponen yang terdapat pada setiap microservices. Sehingga memberikan gambaran bagaimana sistem bekerja pada level cluster Kubernetes dan pada level microservices.

1) Alur Sistem Komponen Microservices

Alur sistem pada komponen microservice heart rate analytic dan gateway adalah sama sehingga dijelaskan ke dalam satu bagian, sedangkan alur sistem pada komponen microservice user dashboard berbeda dengan komponen microservices lainnya

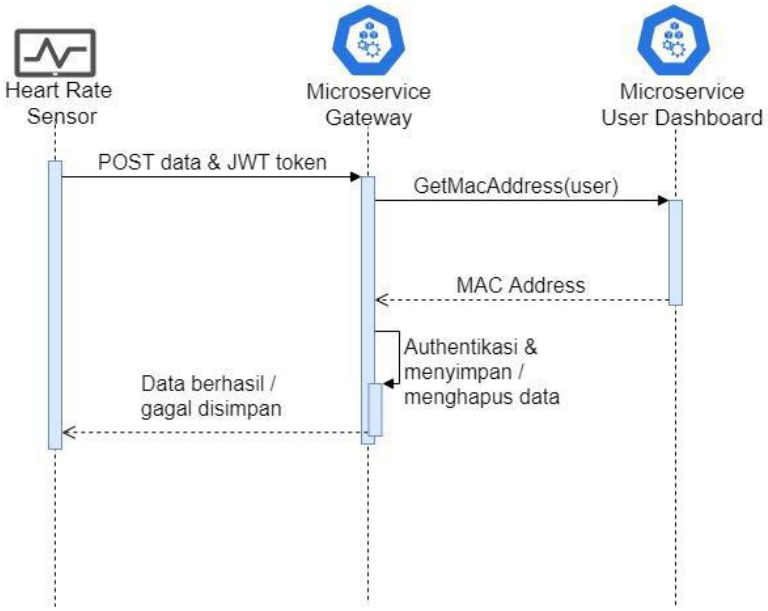




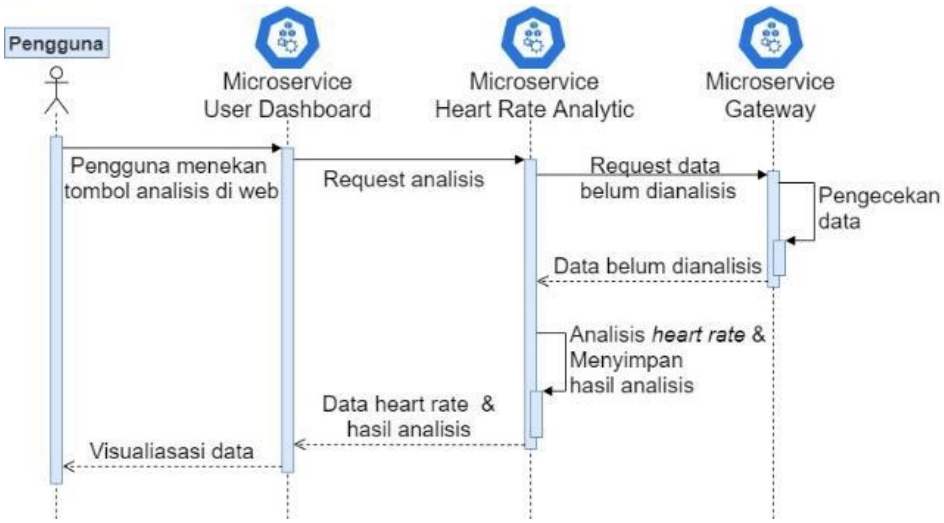
Pada gambar diatas dapat dilihat pengguna dapat mengakses langsung komponen aplikasi melalui user interface berupa aplikasi web, hal ini yang membedakan alur komunikasi komponen microservices user dashboard dan lainnya. Alur komunikasi dibagi menjadi dua. Pertama, pengguna mengakses komponen aplikasi melalui web untuk menjalankan fungsionalitas yang dimiliki oleh microservices user dashboard. Komponen aplikasi mengakses database untuk mendapatkan informasi yang dibutuhkan. Alur komunikasi kedua, microservices lain mengakses API dari microservice user dashboard kemudian API melakukan query ke database untuk mendapatkan data yang dibutuhkan kemudian dikembalikan oleh API berupa response.

2) Alur Sistem Microservices

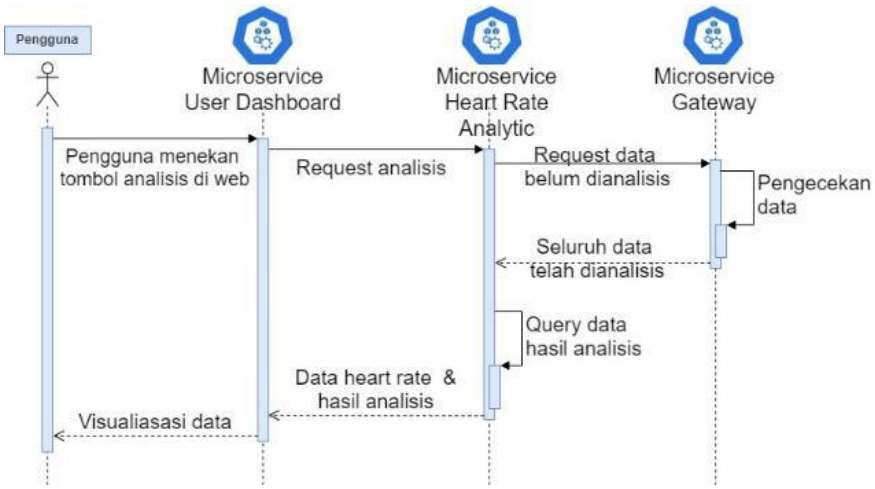
Perancangan alur sistem microservices merupakan rancangan alur komunikasi antara microservices yang dikembangkan pada penelitian ini. Secara umum ada tiga alur komunikasi utama yang dibangun pada penelitian ini yaitu alur pengiriman data ke gateway, alur analisis data dan alur visualisasi data.



Sequence Diagram Alur Pengiriman Data Ke Gateway



Sequence Diagram Alur Analisis Data



Sequence Diagram Alur Visualisasi Data

3.8 Perancangan dan Pengujian

Perancangan pengujian yang dirumuskan terdiri dari perancangan skalabilitas, selfhealing, dan persistent pod database.

1) Perancangan Pengujian Skalabilitas

Pengujian ini bertujuan untuk mengetahui kemampuan dan batasan dari infrastruktur yang dibangun dalam melakukan analisis data. Pengujian skalabilitas ini menggunakan 5 skenario dengan pola berdasarkan penelitian Faathin dkk (Faathin, Pramukantoro, & Bakhtiar, 2019). Pola yang digunakan untuk melakukan pengujian adalah analisis 1.000, 5.000, 10.000, 50.000 dan 100.000 data. Pengujian dilakukan dengan cara melakukan request analisis menggunakan Postman.

2) Perancangan Pengujian Self-Healing

Pengujian bertujuan untuk mengetahui apakah cluster selalu dapat kembali ke keadaan normal jika terdapat pod yang mati. Penelitian ini dilakukan dengan cara menghapus salah satu pod yang sedang berjalan. Kemudian peneliti mengamati sistem untuk mengetahui apakah cluster mampu kembali ke keadaan normal setelah pod dihapus.

3) Perancangan Pengujian Persistent Pod Database

Pengujian ini bertujuan untuk mengetahui apakah data yang telah disimpan pada database tidak terhapus meskipun pod database mati. Pengujian ini dilakukan karena pod memiliki siklus hidup yang menyebabkan pod dapat mati dan pod yang telah mati tidak dapat dijalankan lagi, sehingga penyimpanan dan data yang ada di dalam pod juga terhapus. Pengujian dilakukan dengan cara menghapus pod dan dilakukan pengamatan data yang tersimpan pada pod database sebelum dan setelah pod dihapus. Pengujian ini dilakukan pada pod database setiap microservices.

3.9 Pengujian

Pengujian dilakukan dengan berbagai 3 tahap yaitu Skalabilitas, Self-Healing, dan Persistent Pod Database. Pengujian skalablitas dilakukan untuk mengetahui batasan infrastruktur yang telah dibangun dalam melakukan analisis data. Pengujian dilakukan menggunakan tool Postman untuk melakukan request analisis ke microservice heart rate analytic. Skenario uji yang dilakukan adalah dengan melakukan analisis 1.000, 5.000, 10.0000, 50.000, dan 100.000 data.

Sementara Pengujian dilakukan dengan cara menghapus salah satu pod, kemudian dilakukan pengamatan, seluruh proses yang terjadi dimulai pada saat sebelum pod dihapus hingga pod baru berhasil berjalan untuk menggantikan pod yang mati.

Dan yang terakhir Pod database dikatakan persistent jika data yang tersimpan pada database tidak terhapus meskipun pod database mati dan cluster menjalankan pod database baru. Pengujian ini dilakukan pada setiap pod

database yang dimiliki masing-masing microservices. Pengujian dilakukan dengan cara melakukan perbandingan data yang tersimpan pada database setelah dan sebelum pod dihapus.

3.10 Hasil dan Kesimpulan

Berdasarkan hasil dan pengujian, dapat ditarik kesimpulan bahwa infrastruktur analisis data heart rate dapat diimplementasikan menggunakan arsitektur microservices, berbasis docker container dan kubernetes untuk melakukan manajemen cluster. Pengujian self healing menunjukkan bahwa sistem mampu kembali ke keadaan normal ketika terdapat pod yang mati. Pada pengujian self-healing juga didapatkan hasil bahwa ketika pod mati, deployment membuat pod baru terlebih dahulu dan menunggu hingga pod baru dalam keadaan berjalan, kemudian pod lama benar-benar dihapus dari cluster. Hal tersebut menunjukkan bahwa sistem akan selalu tersedia untuk diakses.

Penelitian berhasil dengan menunjukkan bahwa sistem akan selalu tersedia untuk diakses. Pengujian persistent pod database menunjukkan bahwa seluruh pods database dari setiap microservices mampu menyimpan data secara persistent.

BAB 4
PENUTUP

Perbandingan dengan metode Waterfall, Prototype, RAD, Scrum, Agile, DevOps. Berikut ini adalah perbandingan metode tersebut berdasarkan kelebihan dan kekurangannya

Metode	Kelebihan	Kekurangan

Waterfall	● Mudah digunakan Mudah dimengerti dan mudah dikelola ● Requirement dari sistem bersifat stabil ● Lebih mengutamakan hasil daripada waktu dan biaya ● Alur dari sistem yang jelas sehingga pengerjaan proyek akan semakin detail ● Memiliki gambaran akhir yang jelas ● Baik dalam pendokumentasian Karena hal tersebut, setiap progres dan informasi bisa tercatat dan dapat diakses oleh pengembang yang lain	● Waktu yang relatif lama ● Sistem yang sudah baku sehingga sulit untuk melakukan improvisasi ● Biaya yang bisa saja bertambah jika ada perubahan mendadak ● Tidak siap dengan adanya perubahan mendadak karena akan mengubah keseluruhan proses yang sudah ada ● Kurang cocok digunakan untuk mengembangkan perangkat lunak atau sistem yang berskala besar
Prototype	● Hemat waktu dan biaya ● Dapat meminimalisir kesalahan sistem dari awal karena adanya keterlibatan pemilik sistem ● Implementasi dan penggunaan sistem lebih mudah karena klien sudah memiliki gambaran sistem sebelumnya ● Mempermudah dalam memperkirakan pengembangan sistem selanjutnya ● Klien dapat terpuaskan jika pengembang berhasil memenuhi kebutuhan	● Masalah utama pada metode ini adalah pada klien, jika klien tidak puas di tahap awal pengembangan maka akan menghabiskan banyak waktu ● Komunikasi antara pengembang dan klien harus berjalan secara efektif dan tidak boleh ada Miskomunikasi karena akan menghambat pengembangan sistem ● Klien yang terus menambah requirement dari akan menambah kompleks pembuatan dari sistem
RAD	● Kebutuhan aplikasi bisa berubah sewaktu-waktu. ● Aplikasi dikembangkan berdasarkan kebutuhan dan keinginan user. ● Mudah mengakomodasi perubahan sistem ● Mudah dalam menentukan dasar sistem ● Waktu pengembangan aplikasi bisa lebih cepat dan efektif. ● Mempermudah proses integrasi.	● Metode ini sangat bergantung pada skill individu dan memerlukan pengembang ahli dan kerjasama yang baik, karena tanpa itu tidak mungkin pengembang bisa menyelesaikan sistem dengan metode RAD, terlebih jika mengerjakan proyek yang besar, namun jika ada pengembang yang ahli dan kerjasama yang baik metode ini akan sangat hemat waktu dan biaya ● Tidak cocok untuk proyek dengan dana terbatas ● Hanya cocok untuk proyek yang waktunya singkat.
Scrum	● Metode yang hemat waktu dan biaya ● Siap jika sewaktu waktu harus mengalami berubah mendadak ● Dokumentasi dan pengujian terus dilakukan setelah software dibangun	● Rawan terjadinya scope creep ● Setiap tugas harus didefinisikan dengan baik, karena hal ini dapat mempengaruhi perkiraan biaya dan waktu pengerjaan proyek. ● Metode ini sangat memerlukan orang-orang yang sudah

	● Proses scrum mampu menyatakan bahwa produk selesai saat diperlukan	berpengalaman ● Developer harus selalu siap dengan perubahan karena tiap perubahan akan selalu diterima
Agile	● Testing dilakukan setiap saat. ● Rilis bisa lebih cepat ● Proses Iterative dan Incremental. ● Requirement dapat berubah sewaktu-waktu. ● Pelacakan requirement dengan melihat Backlog produk. ● Desain dan implementasi yang sederhana ● Lebih Dinamis Serta Mendukung Real-time tracking	● Membutuhkan manajemen tim yang terlatih. ● Agile Tidak cocok digunakan dalam proyek yang berskala besar. ● Perkiraan waktu rilis dan harga software sulit ditentukan
DevOps	● Reaksi cepat terhadap perubahan pasar dan permintaan pelanggan ● Meningkatkan stabilitas dan kualitas ● DevOps lebih cocok untuk proses end-to-end.	● Kurang cocok untuk tim yang berskala kecil