PENGEMBANGAN SISTEM MONITORING SISTEM MICROSERVICES UNTUK MENINGKATKAN OBSERVABILITY

SKRIPSI

Disusun oleh:
HARITS RIZKAL ALIAMDY
19/439814/TK/48544

JURUSAN TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI
FAKULTAS TEKNIK UNIVERSITAS GADJAH MADA
YOGYAKARTA

2023

HALAMAN PENGESAHAN

PENGEMBANGAN SISTEM MONITORING SISTEM MICROSERVICES UNTUK MENINGKATKAN OBSERVABILITY

SKRIPSI

Diajukan Sebagai Salah Satu Syarat untuk Memperoleh
Gelar Sarjana Teknik Program S-1
Pada Jurusan Teknik Elektro dan Teknologi Informasi Fakultas Teknik
Universitas Gadjah Mada

Disusun oleh:

HARITS RIZKAL ALIAMDY
19/439814/TK/48544

Telah disetujui dan disahkan
pada tanggal 3 Februari 2014

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Sujoko Sumaryono, M.T	**Bimo Sunarfri Hantono, S.T., M.Eng.**
NIP. 196104181988031001	**NIP. 1977 0131 2002 12 1 003**

HALAMAN PERSEMBAHAN

Untuk Ibu, Bapak,
dan Adik-adikku tercinta.

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Puji syukur penulis panjatkan ke hadirat Allah SWT karena hanya dengan rahmat dan hidayah-Nya, Tugas Akhir ini dapat terselesaikan tanpa halangan berarti. Keberhasilan dalam menyusun laporan Tugas Akhir ini tidak lepas dari bantuan berbagai pihak yang mana dengan tulus dan ikhlas memberikan masukan guna sempurnanya Tugas Akhir ini. Oleh karena itu dalam kesempatan ini, dengan kerendahan hati penulis mengucapkan terima kasih kepada:

1. Bapak Sarjiya, S.T., M.T., Ph.D., selaku Ketua Jurusan Teknik Elektro dan Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada,
2. Bapak Sigit Basuki Wibowo, S.T., M.Eng. selaku dosen pembimbing pertama yang telah memberikan banyak bantuan, bimbingan, serta arahan dalam Tugas Akhir ini,
3. Bapak Bimo Sunarfri Hantono, S.T., M.Eng. selaku dosen pembimbing kedua yang juga telah memberikan banyak bantuan, bimbingan, serta arahan dalam Tugas Akhir dan kegiatan-kegiatan yang lain,
4. Bapak Warsun Najib, S.T., M.Sc. selaku dosen pembimbing akademis penulis dan juga dosen pembimbing lapangan penulis pada KKN-PPM UGM 2013 Unit SLM07,
5. Seluruh Dosen di Jurusan Teknik Elektro dan Teknologi Informasi FT UGM, yang tidak bisa disebutkan satu-satu, atas ilmu dan bimbingannya selama penulis berkuliah di JTETI,
6. Ibu dan Bapak yang selama ini telah sabar membimbing, mengarahkan, dan mendoakan penulis tanpa kenal lelah untuk selama-lamanya, dan
7. Cantumkan pihak-pihak lain yang ingin anda berikan ucapan terimakasih.

Penulis menyadari bahwa penyusunan Tugas Akhir ini jauh dari sempurna. Kritik dan saran dapat ditujukan langsung pada e-mail atau *mention* langsung pada akun *twitter* saya. Akhir kata penulis mohon maaf yang sebesar-besarnya apabila ada kekeliruan di dalam penulisan Tugas Akhir ini.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, 15 Januari 2014

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SINGKATAN

A

AJAX	Asynchronous JavaScript and XML
AP	Access Point
API	Application Programming Interface

C

CLI	Command Line Interface

C

DFM	Discovered Full Mesh

E

ERD	Entity Relationship Diagram

F

FTDI	Future Technology Devices International
FUSE	Filesystem in Userspace

I

IP	Internet Protocol

J

JTETI	Jurusan Teknik Elektro dan Teknologi Informasi

L

LAN	Local Area Network

O

OSI	Open Systems Interconnection

R

RF	Radio Frequency

S

SDLC	Software Development Life Cycle
SFTP	Secure Shell File Transfer Protocol
SSHFS	Secure Shell Filesystem

U

UGM	Universitas Gadjah Mada
USB	Universal Serial Bus

V

VRS	Virtual Routing Structure

W

WAP	Wireless Access Point
WIT	Western Indonesian Time
WLAN	Wireless Local Area Network
WSN	Wireless Sensor Network

Intisari

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt.

Kata kunci : *wireless sensor network*, *Internet Protocol*, WiFi, interoperabilitas.

Abstract

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt.

Keywords : *wireless sensor network, Internet Protokol, WiFi, interoperability*.

BAB I

LATAR BELAKANG

1.1 Latar Belakang Masalah

Saat ini, perkembangan teknologi sangatlah cepat, terlebih dengan adanya revolusi industri 4.0. Dengan perkembangan yang cepat itu kebutuhan akan perangkat lunak pun semakin banyak dan semakin kompleks. Menurut State of the Developer Nation report pada edisi 20 terdapat 24.3 juta pengembang perangkat lunak di seluruh dunia yang akan meningkat 20% setiap tahunnya dan diperkirakan pada tahun 2030 akan ada 45 juta pengembang perangkat lunak[1]. Hal tersebut menggambarkan betapa tingginya demand untuk pembuatan perangkat lunak. Perkembangan seperti ini membuat beberapa konsep dan cara pengembangan perangkat lunak yang sebelumnya dipakai menjadi tidak relevan karena kebutuhan perangkat lunak yang semakin banyak dan kompleks. Salah satunya adalah arsitektur monolith pada perangkat lunak, dimana sebuah perangkat lunak dibangun sebagai satu kesatuan utuh. Pada arsitektur monolith tiap bagian perangkat lunak terikat dengan erat (*tightly coupled*) sehingga ketika aplikasi sudah begitu besar akan sangat sulit untuk melakukan perubahan karena ada kemungkinan perubahan tersebut berdampak pada bagian lain di perangkat lunak. Hal ini berdampak pada waktu pengembangan yang lebih lambat, karena para pengembang harus menguji dan *deploy* seluruh aplikasi setiap kali terjadi perubahan.

Permasalahan yang umum terjadi pada arsitektur monolith coba diselesaikan oleh arsitektur microservices. Microservices merupakan salah satu jenis arsitektur perangkat lunak yang sedang populer belakangan ini. microservices membagi perangkat lunak menjadi beberapa bagian yang lebih kecil yang disebut servis. Masing-masing servis berjalan secara terpisah independen berdasarkan domain bisnisnya. Arsitektur microservices hadir untuk menyelesaikan masalah atau limitasi pada arsitektur tradisional monolith yang diantaranya kemudahan dalam proses-proses maintainability, reusability, scalability, availability, dan automated deployment.

Meskipun arsitektur miroservices menyelesaikan banyak permasalahan yang ada pada arsitektur monolith, terdapat masalah dan tantangan baru pada bagian monitoring dan memahami keseluruhan sistem secara utuh. Dikarenakan jumlah servis yang banyak dan tiap servis terdistribusi secara independen, cukup sulit untuk me-

lacak aliran data dan mengetahui bagian yang bermasalah atau menjadi *bottle-neck* pada sistem. Karena hal tersebutlah diibutuhkan suatu cara untuk menyelesaikan permasalahan tersebut. Sebagai gambaran, perusahaan streaming online Netflix memiliki lebih dari 1000 microservice[2], sebuah request yang masuk dapat melewati banyak servis sebelum memberikan respon. Jika terjadi masalah pada salah satu servis, maka akan sulit untuk mengetahui bagian mana yang bermasalah karena tidak mungkin seorang pengembang dapat mengetahui perilaku sistem secara keseluruhan. Oleh karena itu, perlu adanya distributed tracing pada sistem microservices.

Distributed tracing merupakan suatu teknik pada sistem perangkat lunak yang digunakan untuk memantau dan menganalisa aliran sebuah request yang melewati beberapa komponen dalam hal microservices merupakan servis. Distributed tracing membuat pengembang untuk memahami perilaku dan kinerja sistem secara keseluruhan.

Pada penelitian kali ini, akan dilakukan implementasi dan analisis distributed tracing pada sistem microservices. Hasil dari penelitian ini diharapkan dapat dijadikan acuan dan pertimbangan dalam melakukan implementasi distributed tracing pada sistem microservices.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dituliskan sebelumnya, terdapat masalah dan tantangan dalam upaya mengetahui keadaan internal suatu sistem perangkat lunak berbasis microservices. Masalah tersebut muncul karena arsitektur microservices membagi sistem menjadi bagian-bagian yang lebih kecil yang secara alamiah akan membuat sistem perangkat lunak menjadi lebih kompleks. Permasalahan yang muncul pada arsitektur microservices terkait upaya mengetahui keadaan internal sistem terebut diantaranya adalah:

1. Apakah metode distributed tracing dapat mengetahui aliran request pada sebuah sistem microservices?
2. Apakah metode distributed tracing dapat membantu dalam deteksi error dan bottle-neck pada sebuah sistem microservices?
3. Bagaimana menerapkan distributed tracing pada sistem microservices yang memiliki teknologi yang berbeda-beda?

4. Bagaimana distributed tracing dapat membantu dalam proses monitoring dan debugging pada sistem microservices?
5. Bagaimana dampak dari penerapan distributed tracing terhadap performa dari sistem microservices?

1.3 Tujuan Penelitian

Penelitian ini bertujuan untuk mengetahui efektivitas dari implementasi distributed tracing dalam deteksi error dan bottle-neck pada sistem microservices serta dampak implementasi tersebut terhadap kesluruhan performa sistem.

1.4 Batasan Penelitian

Batasan penelitian ini adalah:

1. Fokus penelitian ini adalah pada penerapan distributed tracing pada sistem microservices dalam mendeteksi error yang terjadi pada sistem microservices serta dampak dari penerapan distributed tracing terhadap performa dari sistem microservices.
2. Sistem microservices yang digunakan menggunakan dua bahasa pemrograman yang berbeda yaitu Javascript dan Go.
3. Sistem microservices yang digunakan menggunakan dua sistem basis data yang berbeda yaitu Postgresql dan MySQL.
4. Semua tools yang digunakan pada penelitian ini merupakan open source.
5. Sistem microservices yang digunakan pada penelitian ini berjalan pada kubernetes dengan 1 node master dan 2 node worker.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah:

1. Bagi praktisi pengembang perangkat lunak, penelitian ini dapat dijadikan dasar untuk pembuatan tools monitoring untuk mengetahui keadaan internal dari sistem microservices.

2. Bagi penulis, penelitian ini dapan menambah wawasan, ilmu dan pengetahuan dalam pembuatan tulisan ilmiah, khususnya pada topik terkait monitoring pada sistem perangkat lunak modern.

3. Bagi pelaku bisnis, penerapan observability dapat meningkatkan customer experience.

1.6 Sistematika Penulisan

BAB I : PENDAHULUAN

Pada bab ini dijelaskan latar belakang, rumusan masalah, batasan, tujuan, manfaat, keaslian penelitian, dan sistematika penulisan.

BAB II : TINJAUAN PUSTAKA DAN LANDASAN TEORI Bab tersebut berisi tentang hasil-hasil penelitian yang telah dilakukan oleh para peneliti sebelumnya yang dapat ditemukan dalam tinjauan pustaka, serta teori-teori yang mendukung penelitian yang terdapat pada dasar teori dan analisis perbandingan metode dari penelitian sebelumnya.

BAB III : METODOLOGI PENELITIAN

Bab tersebut membahas mengenai rincian mengenai alat dan bahan yang digunakan dalam penelitian, seperti perangkat keras dan perangkat lunak yang digunakan. Selain itu, dijelaskan pula mengenai metode yang digunakan dalam penelitian. Bab tersebut juga membahas mengenai alur penelitian, yaitu tahapan-tahapan yang dilakukan dalam penelitian untuk mencapai tujuan yang telah ditetapkan.

BAB IV : HASIL DAN PEMBAHASAN

Pada bab ini dijelaskan hasil penelitian dan pembahasannya.

BAB V : KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan yang diperoleh dari penelitian yang telah dilakukan. Selain itu, bab ini juga memuat saran-saran untuk pengembangan penelitian selanjutnya berdasarkan hasil temuan yang didapatkan dari penelitian yang telah dilakukan.

BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Dalam beberapa tahun terakhir perkembangan teknologi sangat lah pesat, terutama teknologi perangkat lunak.Perkembangan yang pesat ini disebabkan oleh berbagai faktor, termasuk peningkatan kebutuhan akan aplikasi yang lebih canggih dan kompleks, serta kemajuan dalam metode pengembangan perangkat lunak itu sendiri.Dalam menghadapi kompleksitas dan kebutuhan yang semakin tinggi, banyak organisasi atau perusahaan yang beralih dan menggunakan pendekatan microservices dalam pengembangan perangkat lunak.

Salah satu tantangan yang dihadapi dalam penggunaan microservices adalah kesulitan dalam mengetahui keadaan sistem secara menyeluruh. Error yang terjadi pada suatu sistem microservices dapat disebabkan oleh banyak faktor dikarenakan tiap request yang masuk akan melewati banyak layanan yang saling terkait. Oleh karena itu, dibutuhkan sebuah metode yang dapat membantu dalam memahami dan menganalisa perilaku sebuah sistem microservices. Salah satu metode yang dapat digunakan adalah distributed tracing.

Terdapat beberapa penelitian yang dilakukan untuk mengatasi permasalahan tersebut. diantaranya, penelitian yang dilakukan Benjamin H. Sigelman et al yang mengembangkan sebuah tracing sistem bernama Dapper untuk mengatasi permasalahan dan tantangann dalam memonitoring dan memahami perilaku sebuah sistem terdistribusi berskala besar di sistem Google. Penelitian ini membahas betapa sulit dalam mendiagnosis permasalahan performa pada sistem terdistribusi dimana sebuah request akan melewati banyak servis dan komponen sehingga dibutuhkan sebuah infrastruktur tracing untuk mengambil dan menganalisa data terkait alur request tersebut. Dapper adalah solusi yang dikembangkan oleh Google, sebuah infrastruktur pelacakan terdistribusi yang dapat mengumpulkan informasi dari berbagai servis yang terlibat dalam proses sebuah request. Sebuah request yang masuk akan menghasilkan sebuah pohon trace dimana setiap node pada pohon trace tersebut disebut span yang merepresentasikan unit kerja yang dilakukan. Pada setiap pohon trace akan terdapat annotation yang berisi informasi yang memberikan konteks mengenai kejadian yang terjadi pada span tersebut.Dapper menggunakan metode sampling dalam mengum-

pulkan trace alih-alih mengumpulkan semua trace dari request yang terjadi, hal ini dilakukan untuk mengurangi dampak terhadap performa pada microservices. Berdasarkan penelitian tersebut penerapan tracing pada sistem terdistribusi dapat membantu dalam memahami dan menganalisa perilaku sebuah sistem terdistribusi.

Clement Casse et al melakukan penelitian dengan menggunakan distributed tracing untuk mengetahui penggunaan resource yang tidak efisien di aplikasi cloud. Penelitian ini fokus pada pengumpulan informasi pada microservices dan memodelkannya menjadi sebuah hierarchical model graph. Penelitian ini melakukan implementasi pada sebuah microservices aplikasi perbelanjaan yang dideploy pada sebuah cluster kubernetes yang terdiri dari 4 nodes dan 2 zones dan menggunakan opentelemetry untuk mengumpulkan data tracing. Hasil dari penelitian ini menunjukan bahwa model yang digunakan dapat mengetahui penggunaan resource cloud yang tidak efisien.

Menurut penelitian yang dilakukan oleh Putu Napoleon Krishna Bayu, log yang dihasilkan pada server yang memiliki lokasi secara terpisah tidak dapat lagi dimanfaatkan untuk mengetahui keadaan sistem secara efisien dikarenakan karakteristiknya yang semakin menyerupai big data dalam hal kecepatan pembuatan log, besaran volume log, dan variasi struktur log. Oleh karena itu dibutuhkan suatu solusi yaitu log management yang dapat menyimpan dan mengelola log secara terpusat. Log management terdiri dari beberapa tahapan yaitu pembuatan, pengumpulan, transformasi, penyimpanan, analysis dan archival.Penelitian ini menggunakan Elastic Stack yang merupakan produk open-source yang dikembangkan oleh elastic yaitu Elasticsearch, Logstash, dan Kibana. Elasticsearch digunakan sebagai database untuk menyimpan log, Logstash digunakan sebagai agen pengumpul data log, dan Kibana digunakan sebagai platform untuk visualisasi data log yang telah disimpan. Hasil dari penelitian yang dilakukan adalah berhasil mengimplementasikan log monitoring system yang menjalankan tahapan-tahapan log management dengan menggunakan Elastic Stack sehingga keadaan dua buah server dapat dikelola dan dimonitor secara terpusat.

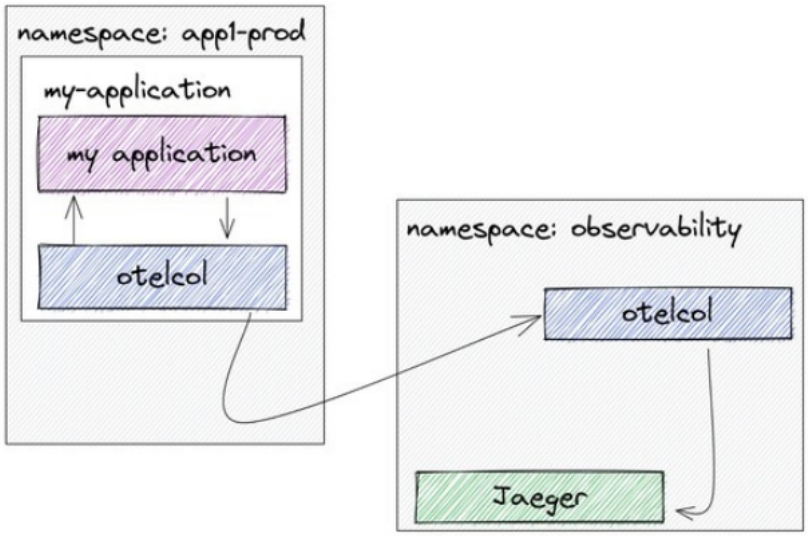
Kemudian, penelitian yang dilakukan oleh Guntoro Yudhy Kusuma et al yang melakukan perancangan sistem monitoring performa aplikasi menggunakan Opentelemetry dan Grafana Stack. Aplikasi monitoring yang dirancang berperan untuk mengumpulkan data telemetri berupa metrics dan traces dari sebuah sistem REST API yang diinstrumentasi menggunakan sebuah kerangka kerja sumber terbuka Opentelemetry. Data telemetri metrics yang telah dikumpulkan akan dikirim menuju sebuah alat alerting dan monitoring Prometheus sedangkan data telemetri traces akan dikirim

menuju sistem backend Jaeger. Prometheus dan Jaeger kemudian diintegrasikan ke Grafana sebagai alat visualisasi data telemetri. Hasil dari penelitian ini menunjukan bahwa rancangan sistem monitoring berjalan dengan baik dengan beberapa kelemahan diantaranya penurunan jumlah throughput secara rata-rata sebesar 23,32% dan kenaikan secara rata-rata sebesar 22,80% pada request latency.

Pada Kubecon North America 2021 yang merupakan sebuah konferensi untuk pengembang Kubernetes, Juraci Paixao Kroehling menjelaskan bagaimana saja deployment pattern untuk Opentelemetry Collector. Terdapat dua pattern untuk melakukan deployment Opentelemetry Collector pada Kubernetes yaitu:

1. Deployment menggunakan Sidecar

   Sidecar pada Kubernetes merupakan sebuah container yang berjalan bersama container utama pada sebuah pod. pada pattern ini Opentelemetry Collector akan dijalankan sebagai sebuah sidecar bersamaan dengan kontainer utama (aplikasi utama). Kontainer utama akan mengirimkan data telemetri menuju sidecar Opentelemetry Collector yang kemudian sidecar tersebut akan mengirimkan data telemetri tersebut ke external collector seperti pada gambar 2.2.1.

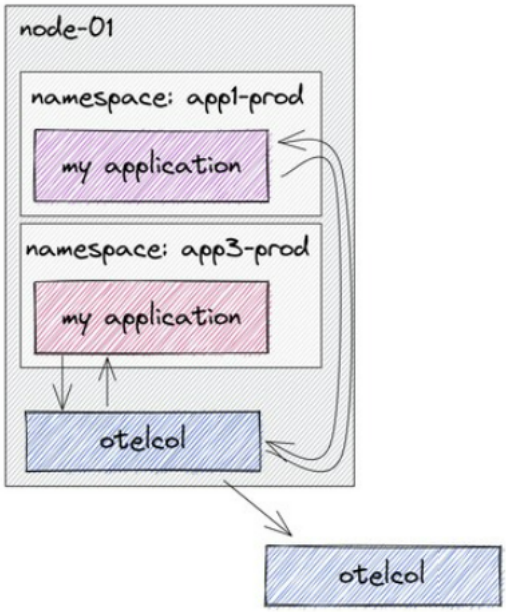


Gambar 2.1: Deployment Opentelemetry Collector dengan sidecar.

2. Deployment menggunakan DaemonSet

   Berbeda dengan sidecar, DaemonSet menjalankan Opentelemetry Collector pada setiap node. Cara kerja dari pattern ini adalah setiap pod yang berada pada

satu node akan mengirimkan data telemetri menuju OpenTelemetry Collector pada node tersebut yang kemudian akan mengirimkan data telemetri tersebut ke external collector seperti pada gambar 2.2



Gambar 2.2: Deployment Opentelemetry Collector dengan daemonset.

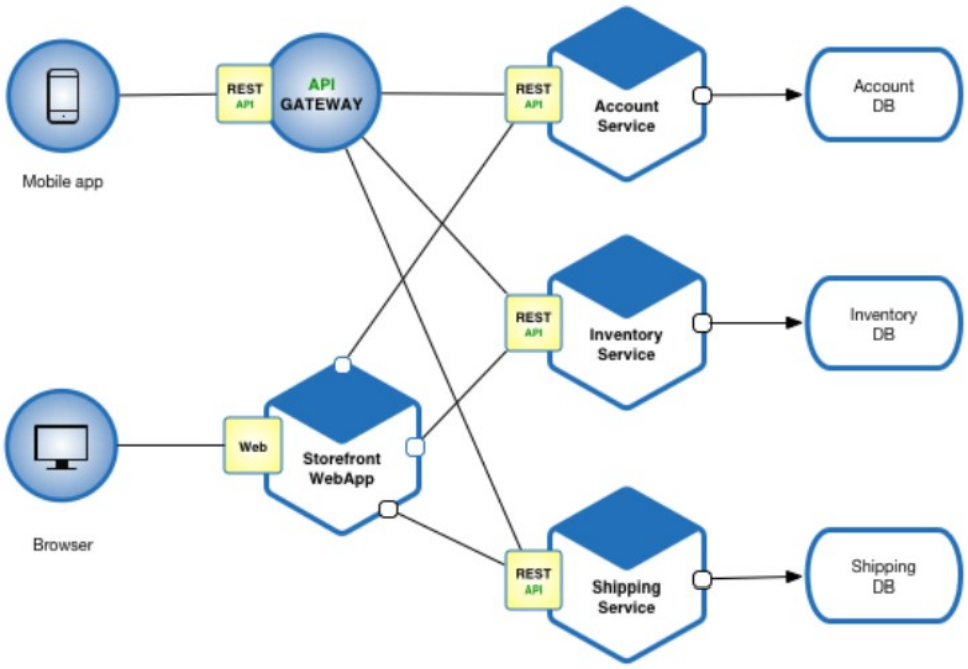
Rolando Brondolin et al melakukan penelitian dengan mengembangkan sebuah pendekatan Black-box monitoring untuk mengukur runtime performa pada microservices. Penelitian ini menggunakan teknologi extended Berkeley Packet Filter (eBPF) yang digunakan untuk mengumpulkan data terkait performa aplikasi, penggunaan power, dan aktivitas network pada setiap docker container, kubernetes host dan physical host. eBPF merupakan sebuah teknologi yang memungkinkan penulis dapat membuat program yang berjalan di level kernel dengan aman, Dengan eBPF penulis dapat membuat sebuah program yang bertujuan untuk mengekstrak data metrics yang dibutuhkan untuk pengukuran performa aplikasi. Peneliti membuat dua eBPF program berbeda dengan tujuan masing-masing untuk mengumpulkan data performa dan data network. Kedua program tersebut bereaksi pada Linux tracepoint dan Kprobe yang khusus pada data yang ingin dikumpulkan. Ketika eBPF VM menerima event dari linux kernel, event tersebut akan dikirimkan menuju program eBPF yang telah dibuat sesuai dengan jenis event kemudian hasil dari program tersebut akan dikirim dan disimpan pada hash map yang dapat diakses oleh user-space agen. kemudian,

user-space agen akan mengirimkan data yang telah dikumpulkan menuju remote backend.

2.2 Landasan Teori

2.2.1 Microservices

Microservices merupakan sebuah aristektur perangkat lunak yang terdiri dari banyak servis yang independen. Setiap servis mengenkapsulasi fungsi sebuah proses tertentu dan saling berkomunikasi melalui jaringan[3]. Sebagai contoh sebuah aplikasi toko online yang terdiri dari beberapa servis yaitu inventory, account, dan shipping seperti pada gambar 2.1.

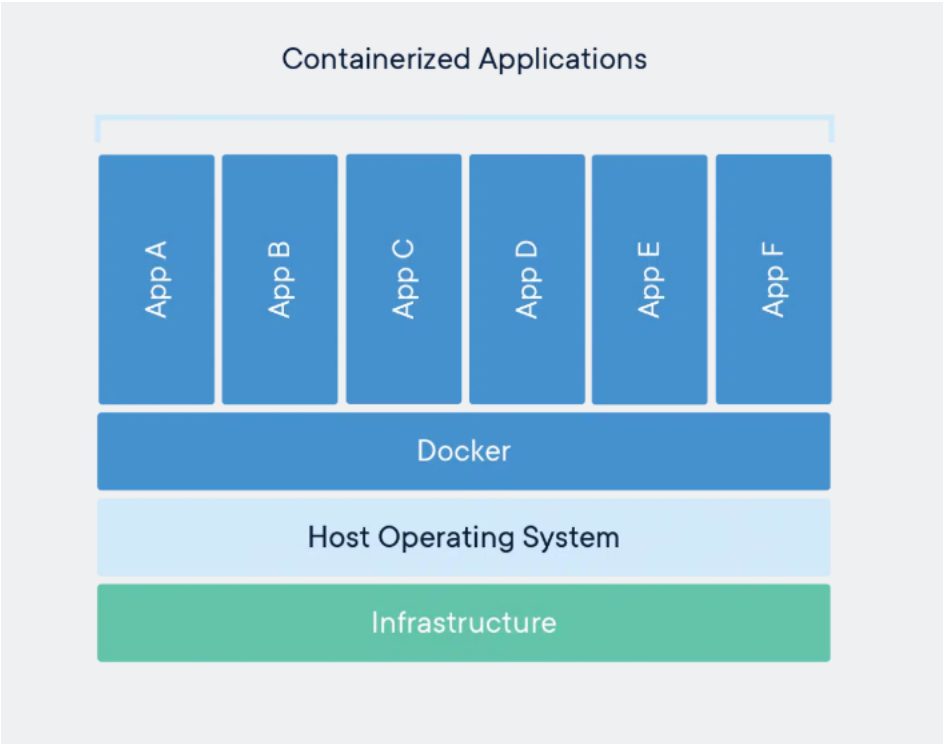


Gambar 2.3: Arsitektur microservices toko online.

2.2.2 Docker

Docker merupakan sebuah teknologi sumber terbuka yang berjalan di windows atau linux yang membantu dalam membuat, mengatur, dan melakukan orkestrasi kontainer. Kontainer merupakan sebuah paket ringan yang berisi kode perangkat lunak, environment serta dependensi lainnya. Kontainer memungkinkan sebuah aplika-

si untuk berjalan di lingkungan yang terisolasi pada sebuah host seperti pada gambar 2.2.

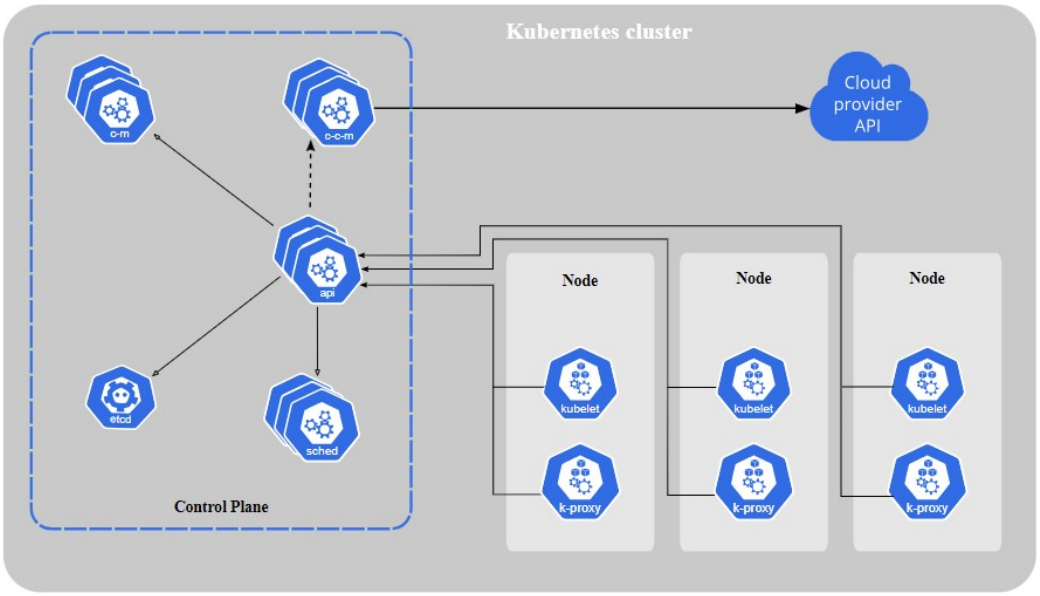


Gambar 2.4: Kontainer yang berjalan pada sebuah host.

2.2.3 Kubernetes

Kubernetes merupakan sebuah platform sumber terbuka yang merupakan bagian dari Cloud Native Computing Foundation (CNCF). Kubernetes digunakan untuk melakukan manajemen container secara otomatis. Kubernetes memiliki kemampuan diantaranya load balancing, manajemen storage, otomasi rollouts dan rollback, pembatasan resource container, self-healing, dan manajemen konfigurasi[4].

Sebuah kubernetes cluster terdiri dari beberapa node. Sebuah node dapat merupakan sebuah linux host yang berjalan baik pada mesin fisik atau virtual. Terdapat dua jenis node pada kubernetes yaitu control plane node dan worker node[5]. Sebuah control plane node beperan untuk mengatur dan mengontrol worker node sedangkan worker node berperan untuk menjalankan aplikasi yang berjalan pada sebuah kontainer sperti pada gambar.



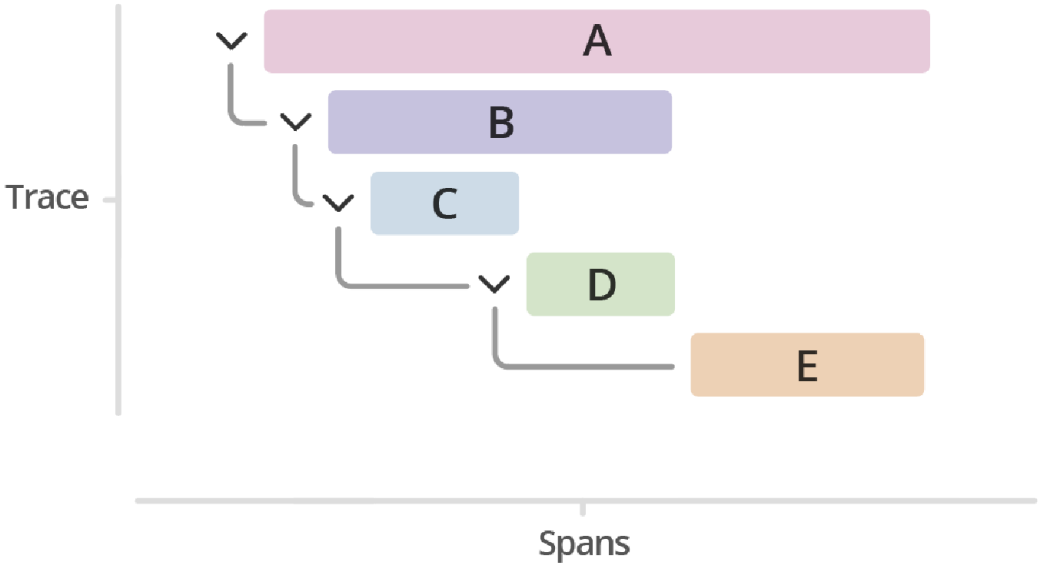
Gambar 2.5: Komponen-komponen pada kubernetes cluster.

2.2.4 Minikube

Minikube merupakan sebuah tool yang digunakan untuk menjalankan sebuah kubernetes cluster pada sebuah host lokal yang tersedia untuk Linux, macOS, dan Windows. Minikube bekerja dengan cara membuat sebuah virtual machine yang berisi sebuah single-node kubernetes cluster. Minikube dapat digunakan untuk melakukan eksperimen pada kubernetes atau untuk melakukan development aplikasi yang akan di deploy pada sebuah cluster kubernetes[6].

2.2.5 Distributed Tracing

Tracing merupakan suatu metode untuk mengetahui serangkaian kejadian atau event yang terjadi pada sebuah request aplikasi. Sebuah tracing dikatakan distributed apabila dalam request tersebut melibatkan atau melewati beberapa proses atau mesin atau jaringan yang berbeda. Sebuah request digambarkan sebagai trace yang terdiri dari beberapa span, span tersebut merupakan unit terkecil kejadian yang terjadi pada sebuah trace. Gambar sebuah trace dapat dilihat pada gambar 2.4.



Gambar 2.6: Trace.

Distributed tracing bekerja dengan cara menginstrumentasi aplikasi sehingga ketika sebuah request masuk aplikasi akan membuat sebuah traceID kemudian traceID tersebut akan dikirimkan ke setiap proses yang dilalui oleh request tersebut. Setiap proses akan membentuk sebuah span dengan spanID. Span akan menyimpan informasi seperti waktu mulai dan selesai, durasi, dan metadata terkait. Kemudian, trace dan span yang telah dibuat akan digabungkan sesuai urutannya oleh sebuah sistem atau platform distributed tracing.

2.2.6 OpenTelemetry

Opentelemetry merupakan observability framework yang bersifat sumber terbuka dan merupakan projek dari CNCF. Opentelemetry menyediakan standar, tools, APIs, dan SDKs yang dapat digunakan untuk instrumentasi, menghasilkan, mengumpulkan, dan mengeskport data telemetri (metrics, logs, traces) untuk membantu dalam analisis performance pada aplikasi. Opentelemetry besifat vendor-agnostic yang berarti dapat digunakan di bahasa dan teknologi banyak teknologi[7].

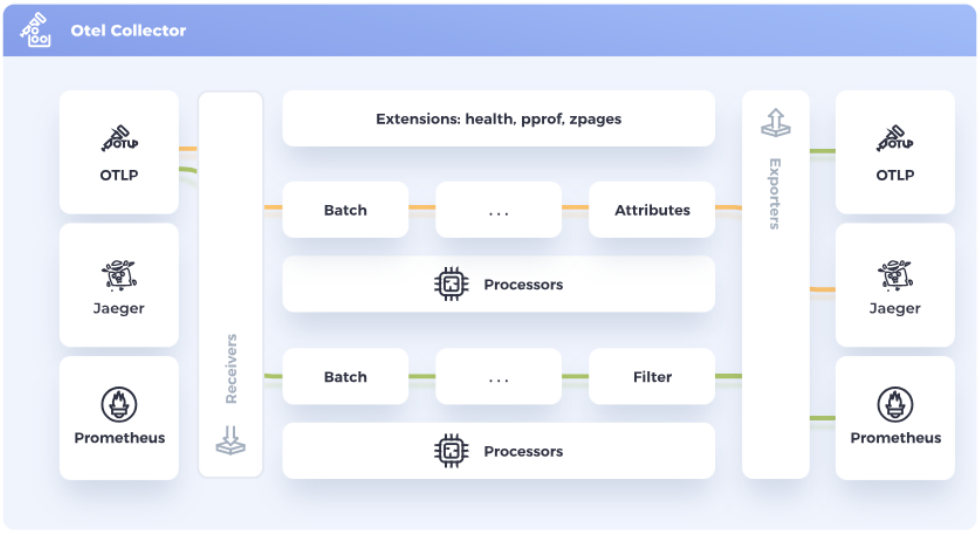
Opentelemetry terdiri dari beberapa komponen, yaitu

1. API and SDK

   Opentelemetry sudah menyediakan API (application programming interface)

dan SDK (Software Development Kit) untuk berbagai macam bahasa pemrograman sehingga dapat membantu untuk melakukan instrumentasi pada aplikasi dengan berbagai bahasa untuk menghasilkan metrics dan tracing telemetri.

2. OpenTelemetry Collector
   Opentelemetry collector merupakan sebuah implementasi yang bersifat vendor-agnostic untuk melakukan penerimaan, pemrosesan, dan mengirimkan data telemetri. dengan opentelemetry collector data telemetri dapat dikirimkan ke berbagai backend tools seperti Jaeger, Prometheus, Zipkin, dan lain-lain[8]. Arsitektur dari opentelemetry collector dapat dilihat pada gambar 2.5.



Gambar 2.7: Arsitektur Opentelemetry collector.

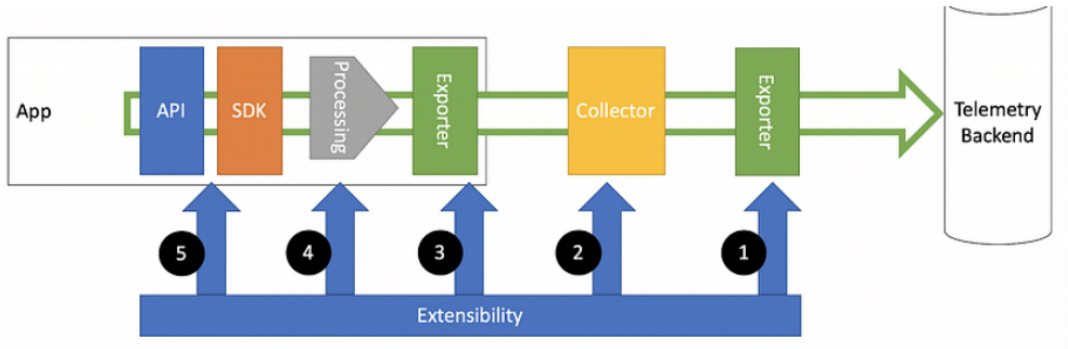
Gambar .... merupakan contoh data telemetri traces dengan standar Opentelemetry.

```
{
  "name": "hello",
  "context": {
    "trace_id": "0x5b8aa5a2d2c872e8321cf37308d69df2",
    "span_id": "0x051581bf3cb55c13"
  },
  "parent_id": null,
  "start_time": "2022-04-29T18:52:58.114201Z",
  "end_time": "2022-04-29T18:52:58.114687Z",
  "attributes": {
    "http.route": "some_route3"
  },
  "events": [
    {
      "name": "Guten Tag!",
      "timestamp": "2022-04-29T18:52:58.114561Z",
      "attributes": {
        "event_attributes": 1
      }
    }
  ]
}
```


Gambar 2.8: OpenTelemetry data trace.

3. OpenTelemetry Protocol
   OTLP merupakan protokol untuk mentransmisikan traces, metrics, dan logs telemetri. OTLP membantu Opentelemetry untuk mengirimkan data 13 ke banyak backend tools. OTLP mengatur bagaimana encoding, transport, dan mekanisme pengiriman data.

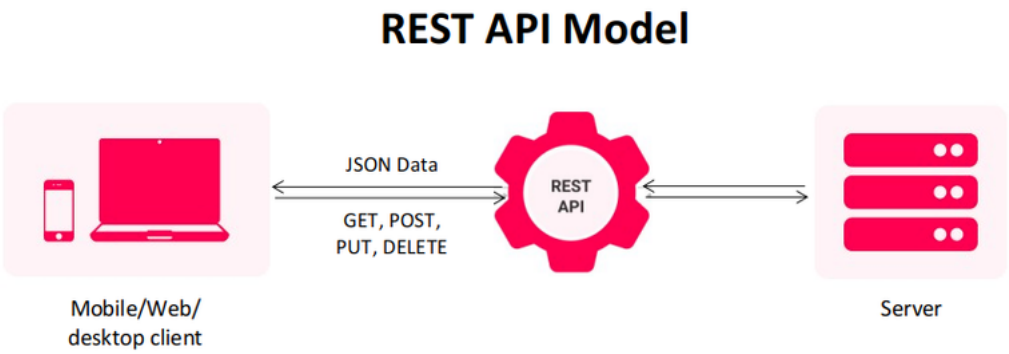
Bagaimana Opentelemetry bekerja dapat dilihat pada gambar Sebuah aplikasi akan menggunakan API dan SDK Opentelemetry untuk menginstrumentasi data telemetri kemudian data tersebut di proses sebelum di export/kirim menuju collector dan dari collector akan diexport menuju backend tools.



Gambar 2.9: Alur kerja Opentelemetry

2.2.7 REST API

Application Programming Interface atau API merupakan sekumpulan kode atau aturan yang mengizinkan dua aplikasi saling berkomunikasi. Salah satu jenis API ada REST API. REST (Representational State Transfer) API merupakan arsitektur API dimana komunikasi yang terjadi antar dua aplikasi bersifat stateless dan menggunakan protokol HTTP atau HTTPS. REST API menggunakan HTTP method seperti GET, POST, PUT, dan DELETE dan menggunakan JSON sebagai format penerimaan atau pengiriman data.



Gambar 2.10: Alur kerja REST API.

Gambar 2.7 merupakan alur kerja dari sebuah REST API dimana sebuah klien melakukan request dengan method GET/POST/PUT/DELETE kemudian request tersebut akan diterima oleh server dan server akan mengirimkan response sesuai dengan request yang diterima dalam bentuk JSON seperti pada gambar .

```
{
    "message": "list of merchants",
    "data": {
        "id": 2,
        "name": "Toko asik bangets",
        "userId": "74a52497-a782-48bc-b65a-393e50136c6a",
        "levelId": 1,
        "user": {
            "id": "74a52497-a782-48bc-b65a-393e50136c6a",
            "email": "rizkal@gmail.com",
            "password": "$2b$10$YmG2DCpKvhjqUE.hSWFiyeP58Cuq4.aIoBkI/rKPtqKcMek82VsQm",
            "createdAt": "2023-03-22T08:42:43.251Z",
            "updatedAt": "2023-03-22T08:42:43.251Z",
            "roleId": 2
        },
        "createdAt": "2023-03-22T10:44:16.733Z",
        "updatedAt": "2023-03-22T10:44:16.733Z",
        "level": {
            "id": 1,
            "name": "FRESH",
            "createdAt": "2023-03-22T09:15:39.950Z",
            "updatedAt": "2023-03-22T09:15:39.950Z"
        }
    }
}
```


Gambar 2.11: Format data JSON.

2.2.8 Load Testing

Load testing merupakan salah satu jenis pengujian pada aplikasi perangkat lunak. Load testing menguji perangkat lunak dengan memberikan beban yang biasanya berupa request ke aplikasi untuk melihat apakah aplikasi tersebut mampu mengatasi beban tersebut dengan baik. Ada banyak tools sumber terbuka untuk melakukan load testing seperti k6, JMeter, Locust, dan lain-lain.

BAB III
METODOLOGI PENELITIAN

3.1 Alat dan Bahan

Alat dan bahan yang digunakan pada penelitian ini terbagi atas perangkat keras dan perangkat lunak yang akan dijelaskan seperti berikut.

3.1.1 Perangkat Keras

Pro omnium incorrupte ea. Elitr eirmod ei qui, ex partem causae disputationi nec. Amet dicant no vis, eum modo omnes quaeque ad, antiopam evertitur reprehendunt pro ut. Nulla inermis est ne. Choro insolens mel ne, eos labitur nusquam eu, nec deserunt reformidans ut. His etiam copiosae principes te, sit brute atqui definiebas id.

a. Kit pancar-rima IQRF TR-53B (3 unit),
b. Kit pengunduh program CK-USB-04 (1 unit),
c. Kit pengembangan DK-EVAL-03 (2 unit),
d. Kit pengembangan CK-EVAL-04 (1 unit),
e. *XBee 802.15.4 Radios (Series 1)* (3 unit),
f. *XBee Explorer USB Board* (1 unit),
g. *2 channel Relay Shield For Arduino (With XBee/BTBee interface)* (2 unit),
h. Arduino Uno (2 unit),
i. TP-LINK MR3020 (1 unit),
j. Kabel USB ke Serial Prolific (1 unit).

3.1.2 Perangkat Lunak

Pro omnium incorrupte ea. Elitr eirmod ei qui, ex partem causae disputationi nec. Amet dicant no vis, eum modo omnes quaeque ad, antiopam evertitur reprehendunt pro ut. Nulla inermis est ne. Choro insolens mel ne, eos labitur nusquam eu, nec deserunt reformidans ut. His etiam copiosae principes te, sit brute atqui definiebas id.

a. Arduino for Mac OS X,
b. CoolTerm,
c. Driver FTDI for Mac OS X,

d. PHP, MySQL, dan uHTTPd,
e. Python dan pustaka PySerial,
f. IQRF IDE v 2.08 for TR-53B,
g. SSHFS,
h. Sublime Text 3.

3.2 Alur Penelitian

Consul graeco signiferumque qui id, usu eu summo dicunt voluptatum, nec ne simul perpetua posidonium. Eos ea saepe prodesset signiferumque. No dolore possit est. Mei no justo intellegebat definitiones, vis ferri lorem eripuit ad. Solum tritani scribentur duo ei, his an adipisci intellegat.

3.3 Tahapan Pelaksanaan

Consul graeco signiferumque qui id, usu eu summo dicunt voluptatum, nec ne simul perpetua posidonium. Eos ea saepe prodesset signiferumque. No dolore possit est. Mei no justo intellegebat definitiones, vis ferri lorem eripuit ad. Solum tritani scribentur duo ei, his an adipisci intellegat.

3.4 Jadwal Kegiatan

Quo no atqui omnesque intellegat, ne nominavi argumentum quo. Eum ei purto oporteat dissentiet, soleat utamur an sit. Et assum dicam interpretaris quo. Cetero alterum ea vel, no possit alterum utroque nec. His fuisset quaestio ad. Has eu tritani incorrupte consequuntur, esse aliquip nec ne 3.1.

Tabel 3.1: Jadwal Penelitian.

No	Keterangan	Bulan					
		1	2	3	4	5	6
1	Studi literatur						
2	Desain						
3	Pembelian bahan						
4	Pembuatan prototipe						
5	Uji coba dan perbaikan						
6	Penulisan skripsi						

BAB IV

HASIL DAN PEMBAHASAN

4.1 Subbab 1

Habeo perfecto in sea. Ea deleniti gloriatur pri, paulo mediocrem incorrupte sea ei. Ad mollis scripta per. Incorrupte sadipscing ne mel. Mel ex nonumy malorum epicurei.

Ne per tota mollis suscipit. Ullum labitur vim ut, ea dicit eleifend dissentias sit. Duis praesent expetenda ne sed. Sit et labitur albucius elaboraret. Ceteros efficiantur mei ad. Hendrerit vulputate democritum est at, quem veniam ne has, mea te malis ignota volumus.

Eros reprimique vim no. Alii legendos volutpat in sed, sit enim nemore labores no. No odio decore causae has. Vim te falli libris neglegentur, eam in tempor delectus dignissim, nam hinc dictas an.

4.2 Subbab 2

Habeo perfecto in sea. Ea deleniti gloriatur pri, paulo mediocrem incorrupte sea ei. Ad mollis scripta per. Incorrupte sadipscing ne mel. Mel ex nonumy malorum epicurei.

4.2.1 Subsubbab 2 1

Ne per tota mollis suscipit. Ullum labitur vim ut, ea dicit eleifend dissentias sit. Duis praesent expetenda ne sed. Sit et labitur albucius elaboraret. Ceteros efficiantur mei ad. Hendrerit vulputate democritum est at, quem veniam ne has, mea te malis ignota volumus.

```
config mount
        option target        /mnt
        option device        /dev/sda1
        option fstype        ext3
        option options       rw,sync
        option enabled       1
        option enabled_fsck  0
        option is_rootfs     1
```

```
# opkg update
# opkg install python pyserial
```

4.2.2 Subsubbab 2 2

Consul graeco signiferumque qui id, usu eu summo dicunt voluptatum, nec ne simul perpetua posidonium. Eos ea saepe prodesset signiferumque. No dolore possit est. Mei no justo intellegebat definitiones, vis ferri lorem eripuit ad. Solum tritani scribentur duo ei, his an adipisci intellegat.

4.3 Subab 3

Consul graeco signiferumque qui id, usu eu summo dicunt voluptatum, nec ne simul perpetua posidonium. Eos ea saepe prodesset signiferumque. No dolore possit est. Mei no justo intellegebat definitiones, vis ferri lorem eripuit ad. Solum tritani scribentur duo ei, his an adipisci intellegat.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil analisis dan pengujian fungsional aplikasi ini, didapat kesimpulan sebagai berikut:

1. Lorem ipsum is a pseudo-Latin text used in web design, typography, layout, and printing in place of English to emphasise design elements over content.
2. It's also called placeholder (or filler) text. It's a convenient tool for mock-ups.
3. It helps to outline the visual elements of a document or presentation, eg typography, font, or layout. Lorem ipsum is mostly a part of a Latin text by the classical author and philospher Cicero.
4. Its words and letters have been changed by addition or removal, so to deliberately render its content nonsensical; it's not genuine, correct, or comprehensible Latin anymore.

5.2 Saran

1. Lorem ipsum is a pseudo-Latin text used in web design, typography, layout, and printing in place of English to emphasise design elements over content.
2. It's also called placeholder (or filler) text. It's a convenient tool for mock-ups.
3. It helps to outline the visual elements of a document or presentation, eg typography, font, or layout. Lorem ipsum is mostly a part of a Latin text by the classical author and philospher Cicero.
4. Its words and letters have been changed by addition or removal, so to deliberately render its content nonsensical; it's not genuine, correct, or comprehensible Latin anymore.

DAFTAR PUSTAKA

[1] Anna Mleczko, "How many developers are there in the world in 2023?" 2022, [Online]. Available: https://www.future-processing.com/blog/how-many-developers-are-there-in-the-world-in-2019/. [Accessed: June 09, 2023].

[2] shriramvenugopal, "The Story of Netflix and Microservices," 2020, [Online]. Available: https://www.geeksforgeeks.org/the-story-of-netflix-and-microservices/. [Accessed: June 09, 2023].

[3] S. Newman, *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, Inc., 2021.

[4] Kubernetes.io, "Overview of Kubernetes," 2023, [Online]. Available: https://kubernetes.io/docs/concepts/overview/. [Accessed: June 09, 2023].

[5] N. Poulton, *The Kubernetes Book*. Leanpub, 2023.

[6] Kubernetes.io, "Using Minikube to Create a Cluster," 2023, [Online]. Available: https://kubernetes.io/docs/tutorials/hello-minikube/. [Accessed: June 09, 2023].

[7] opentelemetry.io, "What is OpenTelemetry?" 2023, [Online]. Available: https://opentelemetry.io/docs/concepts/what-is-opentelemetry/. [Accessed: June 09, 2023].

[8] ——, "Collector," 2023, [Online]. Available: https://opentelemetry.io/docs/collector/. [Accessed: June 09, 2023].