**TUGAS BESAR PRAKTIKUM ALGORITMA PEMOGRAMAN**

**DJEPRET**



Disusun oleh:

|  |  |
| --- | --- |
| Ega Fauzan Sani | 1101223218 |
| Nabila Hatami Putri | 1101220111 |
| Panji Wijaya | 1101223413 |
| Romi Hasan | 1101223085 |
|  |  |

**ADAPTIVE NETWORK LABORATORY**

**FAKULTAS TEKNIK ELEKTRO**

**UNIVERSITAS TELKOM BANDUNG**

**2023**

# 

# **ABSTRAK**

Dalam praktikum ini, kami membuat sebuah program untuk membantu manajemen sebuah studio foto untuk jadwal foto dan pemesanan frame mencetak foto dalam bahasa c.

**Kata Kunci** : (bahasa c, program, studio foto)

# **DAFTAR ISI**

[**ABSTRAK** ii](#_Toc135771561)

[**DAFTAR ISI** iii](#_Toc135771562)

[**1.** **BAB I** 1](#_Toc135771563)

[**PENDAHULUAN** 1](#_Toc135771564)

[1.1. Latar Belakang 1](#_Toc135771565)

[1.2. Rumusan Masalah 1](#_Toc135771566)

[1.3. Tujuan 1](#_Toc135771567)

[1.4. Manfaat 1](#_Toc135771568)

[**2.** **BAB II** 2](#_Toc135771569)

[**LANDASAN TEORI** 2](#_Toc135771570)

[2.1. Magnet 2](#_Toc135771571)

[2.2. Induksi Magnetik 2](#_Toc135771572)

[2.3. Motor DC 2](#_Toc135771573)

[2.4. Gerak Melingkar 3](#_Toc135771574)

[**PYTHON** 4](#_Toc135771575)

[**1.** **Python Secara Umum** 4](#_Toc135771576)

[**Instalasi Python** 4](#_Toc135771577)

[**Fundamental Python** 4](#_Toc135771578)

[**KUBERNETES** 2](#_Toc135771579)

[**1.** **Kubernetes** 2](#_Toc135771580)

[**Arsitektur Kubenernetes** 2](#_Toc135771581)

[**Konsep Dasar Kubernetes** 3](#_Toc135771582)

[**DOCKER** 4](#_Toc135771583)

[**1.** **Docker Secara umum** 4](#_Toc135771584)

[**Arsitektur Docker** 4](#_Toc135771585)

[**NAMED DATA NETWORKING (NDN)** 6](#_Toc135771586)

[**1.** **Named Data Netwok (NDN)** 6](#_Toc135771587)

[**Arsitektur** 6](#_Toc135771588)

[**Cara Kerja** 7](#_Toc135771589)

[**OPENSTACK** 8](#_Toc135771590)

[**1.** **Definisi** 8](#_Toc135771591)

[**Arsitektur** 8](#_Toc135771592)

[**Layanan OpenStack** 8](#_Toc135771593)

[**DAFTAR PUSTAKA** 10](#_Toc135771594)

# **BAB I**

# **PENDAHULUAN**

* 1. Latar Belakang

Pemanasan global dan perubahan iklim telah menjadi masalah global yang mempengaruhi kehidupan di bumi. Salah satu penyebab utama dari masalah ini adalah penggunaan energi fosil yang tidak ramah lingkungan. Oleh karena itu, diperlukan upaya untuk mengurangi penggunaan energi fosil dan beralih ke sumber energi yang lebih ramah lingkungan. Dalam penelitian ini, kami mengajukan pembuatan alat aplikasi induksi magnetik dan gerak melingkar berupa kipas angin sederhana yang bersumber dari medan magnet sebagai salah satu upaya untuk mengurangi penggunaan energi fosil.

Penelitian ini bertujuan untuk mengeksplorasi kemungkinan penerapan prinsip induksi magnetik dan gerak melingkar dalam pembuatan alat yang dapat menghasilkan angin secara efisien dan efektif. Dengan demikian, penelitian ini diharapkan dapat menghasilkan konsep baru dalam pembuatan alat yang dapat memanfaatkan medan magnet sebagai sumber energi.

Saat ini, penggunaan energi fosil masih sangat dominan dalam kehidupan sehari-hari. Namun, kondisi ini diharapkan dapat berubah dengan adanya penelitian dan pengembangan teknologi yang dapat memanfaatkan sumber energi alternatif yang lebih ramah lingkungan. Dengan adanya alat aplikasi induksi magnetik dan gerak melingkar berupa kipas angin sederhana yang bersumber dari medan magnet, diharapkan dapat terjadi pergeseran dari penggunaan energi fosil ke penggunaan sumber energi alternatif yang lebih ramah lingkungan.

* 1. Rumusan Masalah

Bagaimana cara memanfaatkan prinsip induksi magnetik dan gerak melingkar dalam pembuatan kipas angin secara efisien dan efektif? Serta apakah kipas angin sederhana yang bersumber dari medan magnet dapat menjadi alternatif yang ramah lingkungan dan hemat energi dalam menghasilkan angin?

* 1. Tujuan

Menjelaskan cara memanfaatkan prinsip induksi magnetik dan gerak melingkar dalam pembuatan kipas angin secara efisien dan efektif serta membuktikan bahwa kipas angin sederhana yang bersumber dari medan magnet dapat menjadi alternatif yang ramah lingkungan dan hemat energi dalam menghasilkan angin.

* 1. Manfaat

Harapannya dapat memberikan wawasan baru bagi praktikan serta peneliti lain yang ingin mengembangkan teknologi serupa. Serta dapat meningkatkan kesadaran masyarakat akan pentingnya penggunaan sumber energi alternatif yang lebih ramah lingkungan.

# **BAB II**

# **LANDASAN TEORI**

* 1. Magnet

Magnet atau magnit adalah suatu obyek yang mempunyai suatu medan magnet. Magnet dapat menarik benda lain, beberapa benda bahkan tertarik lebih kuat dari yang lain, yaitu bahan logam. Namun tidak semua logam mempunyai daya tarik yang sama terhadap magnet. [1]

Medan magnet terdapat ruangan disekitar kutub magnet,yang gayanya bisa tarik-menarik dan tolak-menolaknya masih dirasakan oleh magnet lain. Pada logam yang bukan magnet, magnet elementernya mempunyai arah sembarangan (tidak teratur) sehingga efeknya saling meniadakan yang mengakibatkan tidak adanya kutub-kutub magnet pada ujung logam. [1]

Setiap magnet memiliki dua kutub, yaitu: utara (N) dan selatan (S). Kutub magnet adalah daerah yang berada pada ujung-ujung magnet dengan kekuatan magnet yang paling besar berada pada kutub- kutubnya. Jika kutub utara bertemu dengan kutub selatan maka kedua kutub akan saling tarik-menarik. Dan jika kutub beremu dengan kutub sejenisnya (utara bertemu utara atau selatan bertemu selatan) maka keduanya akan saling tolak menolak. [1]

* 1. Induksi Magnetik

Suatu medan magnet yang konstan tidak dapat menghasilkan arus listrik, namun perubahan fluks medan magnet dapat menghasilkan arus listrik (Hukum induksi Faraday). Arus listrik ini dinamakan arus induksi. [2]

Induksi elektromagnetik atau induksi adalah gaya gerak listrik oleh gerakan konduktor melintasi medan magnet atau oleh perubahan fluks magnet dalam medan magnet. Fluks magnetik adalah banyaknya garis gaya magnet yang menembus suatu bidang. [3]

Hukum Lenz menjelaskan bahwa ggl (gaya gerak listrik) induksi dan arus induksi memiliki arah sedemikian rupa, sehingga melawan muatan yang menghasilkan ggl dan arus induksi tersebut. [2]

Hukum Bio-Savart telah membuktikan bahwa arus listrik dapat menghasilkan medan magnet. Menjelaskan secara sistematis hubungan antara kuat arus, titik pengamatan dan kuat medan magnet. Terdapat pengaruh dari jumlah lilitan terhadap jarak yaitu semakin banyak jumlah lilitan maka jaraknya semakin jauh, dan sebaliknya. [2]

* 1. Motor DC

Salah satu penerapan dari konsep induksi magnetik adalah motor dc. Motor dc adalah perangkat yang mengubah energi listrik menjadi energi mekanik dengan menggunakan prinsip induksi magnetik.

Ketika arus listrik dialirkan pada kumparan akan menghasilkan medan magnet yang menghasilkan torsi dan akan memutar motor. Terdapat komponen bernama komutator yang akan menjaga putaran motor listrik agar tetap menghasilkan arus yang searah. [4]

* 1. Gerak Melingkar

Harapannya dapat memberikan wawasan baru bagi praktikan serta peneliti lain yang ingin mengembangkan teknologi serupa. Serta dapat meningkatkan kesadaran masyarakat akan pentingnya penggunaan sumber energi alternatif yang lebih ramah lingkungan.

# **BAB II**

# **METODE PENELITIAN**

* 1. Identifikasi Maslaha dan Solusi

Penggunaan energi fosil yang tidak ramah lingkungan telah menjadi masalah global yang memengaruhi kehidupan di bumi. Salah satu cara untuk mengurangi penggunaan energi fosil adalah dengan mengembangkan sumber energi alternatif yang ramah lingkungan.

Salah satu masalah tersebut adlaah dengan pembuatan kipas angin sederhana ynag disumbertenagai oleh medan magnet. Dengan menggunakan prinsip induksi magnetik dan gerak melingkar, alat ini dapat menghasilkan angin secara efisien dan efektif tanpa menggunakan energi fosil. Hal ini dapat alternatif yang ramah lingkungan dan hemat energi dalam menghasilkan angin. Dengan demikian, pembuatan kipas angin sederhana yang disumbertanagai oleh medan magnet dapat memberikan kontribusi dalam upaya mengurangi penggunaan energi fosil dan beralih ke sumber energi yang lebih ramah lingkungan.

* 1. Studi Pustaka

Induksi magnetik adalah fenomena yang berkaitan dengan kuat medan magnet yang ditimbulkan oleh arus listrik atau perubahan fluks magnetik. Induksi elektromagnetik adalah salah satu contoh penerapan induksi magnetik dalam bidang listrik.

Gerak melingkar adalah gerak benda yang mengelilingi suatu titik pusat dengan jarak tetap. Dalam hal ini, benda tersebut bergerak dengan kecepatan sudut yang konstan dan mengalami percepatan sentripetal yang mengarah ke pusat lingkaran.

Dalam pembuatan kipas angin sederhana yang disumbertenagai oleh medan magnet, prinsip induksi magnetik dan gerak melingkar digunakan secara bersamaan. Arus listrik yang mengalir pada kumparan akan menimbulkan medan magnet yang dapat mempengaruhi magnet yang lain. Dengan memanfaatkan prinsip gerak melingkar, benda tersebut dapat bergerak memutari poros dan menghasilkan angin.

* 1. Desain dan Rancangan Alat

Python menggunakan indentasi untuk me-*hightlight* sebuah baris kode. Indentasi yang digunakan python ialah spasi. Semua pernyataan dengan jarak yang sama ke kanan adalah bagian dari blok kode yang sama. Jika sebuah kode bersarang pada sebuah kode maka blok tersebut harus terindensasi lebih dalam. [12]

* 1. Pengukuran dan Pengujian

Python menggunakan indentasi untuk me-*hightlight* sebuah baris kode. Indentasi yang digunakan python ialah spasi. Semua pernyataan dengan jarak yang sama ke kanan adalah bagian dari blok kode yang sama. Jika sebuah kode bersarang pada sebuah kode maka blok tersebut harus terindensasi lebih dalam. [12]

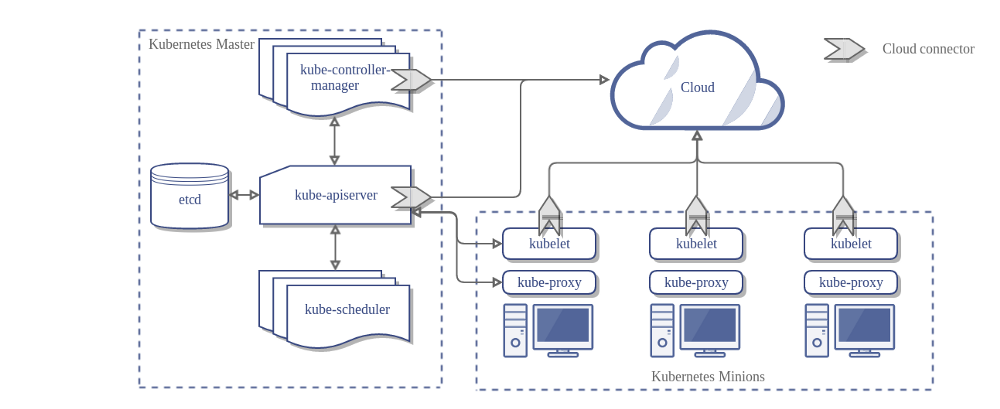
# **KUBERNETES**

## **Kubernetes**

Kubernetes adalah suatu *platform* yang bersifat *open source, portable* dan ektensible yang berfungsi memanajemen kontainer workload dan services. Kubernetes merupakan *container orchestration* yang mengelola sebuah kontainer yang biasanya berupa *docker containers* atau dapat juga berupa teknologi kontainer lainnya. Ini berarti kubenernetes membantu kita dalam mengatur aplikasi berbasis kontainer dalam jumlah yang banyak di berbagai environment. [13]

Dahulu, sofware dibuat dengan cara monolith. Suatu arsitektur monolith memiliki berbagai macam unit dalam sebuah aplikasi. Seiring perkebangan zaman, arsiterktur monolith berganti ke arsitektur microservices yang memecah unit unit pada arsitektur monolith agar lebih *scalable* dan *reliable*. Agar lebih mudah untuk dijalankan semua aplikasi microservices ini dibungkus sebagai kontainer. Hal ini mengakibatkan banyaknya kontainer yang harus dijalankan dan diatur di atas banyaknya server. Hal inilah membuat *tools* orkestrasi kontainer dibutuhkan untuk mengelola kontainer-kontainer tersebut. [14][15]

## **Arsitektur Kubenernetes**



Gambar Klaster Kubernetes [16]

Gambar 5 adalah gambaran suatu klaster kubernetes secara garis besar. Terdapat dua komponen besar yaitu *kubernetes master* dan *kubernetes nodes* atau biasa disebut *worker nodes*. *Kubernetes master* adalah otak dari arsitektur kubernetes, yang memanajemen aplikasi. Sedangkan *worker nodes* adalah bagian yang mengeksekusi aplikasi kita. [17]

Sistem kubernetes berjalan di *kubernetes master*. Dalam *kubernetes master* terdapat beberapa komponen. API server dapat berjalan sebagai linux proses atau sebagai kontainer. API server berfungsi sebagai gerbang masuk ke klaster kubernetes. API server menerima panggilan dari CLI kubernetes yang bernama kubectl, UI kubernetes yaitu kubernetes dashboard serta melalui API melalui kubelet atau dari aplikasi buatan sendiri maupun scripting. *Controller manager* bertugas mengatur state di dalam suatu klaster. *Controller manager* juga bertugas memastikan jumlah kontainer sesuai dengan kebutuhan, dapat diakses dengan ip tertentu dan mengatur hal-hal lainnya yang berhubungan dengan state *workload* aplikasi komponen kontainer kita. *Scheduler* bertugas untuk menempatkan Pod ke tempat yang tepat. Komponen terakhir ialah ETCD. ETCD befungsi sebagai tempat penyimpanan *state* atau *statefull*. [17]

*Worker node* berfungsi untuk memberi pemeliharaan terhadap pod serta menyediakan *environment runtime* bagi kubernetes. Tiap worker node memiliki memiliki komponen kubelet. Kubelet bertugas semacam agen yang bertugas selalu berkomunikasi dengan kubernetes master. Kubelet akan memastikan bahwa tiap aplikasi berjalan di node yang nantinya akan dikontrol oleh *controller manager*. Kube-proxy berjalan pada setiap node dan bertugas sebagai *proxy server* terhadap arus *network* yang masuk ke aplikasi serta menjadi *load balancer*. Komponen terakhir ialah *container manager*. Kubernetes mendukung beberapa container manager seperti docker, containerd, cri-o, rktlet, dan sebagainya. [17]

*Virtual network* berfungsi sebagai penyambung semua kontainer dan komponen yang tersebar di berbagai *worker nodes*. *Virtual network* memungkinkan setiap kubelet dapat berkomunikasi dengan *kubernetes master*. Ini menjadikan mesin-mesin *worker nodes* dan *kubernetes master* yang terpisah seolah menjadi satu mesin yang besar, sehingga dapat memiliki *resource* gabungan yang besar. [17]

## **Konsep Dasar Kubernetes**

Di dalam kubernetes, pod merupakan unit terkecil yang dapat di-*deploy* atau dijalankan di dalam klaster. Tiap *worker node* dapat menjalankan satu atau lebih pod, dan satu pod dapat terdiri dari satu kontainer atau lebih. Biasanya satu pod berisi satu kontainer. Namun beberapa kontainer dapat dibungkus sekaligus dalam satu pod. [18]

Semua *worker nodes* disambungkan oleh *virtual network* sehingga tiap pod dapat memiliki ip address lokal di dalam klaster dan bisa saling berkomunikasi melalui ip address tersebut. Sehingga tiap pod bisa menjadi server tersendiri, dapay diakses melalui ip dan portnya masing masing. [18]

Kita tidak dapat menjalankan kontainer secara langsung, tapi harus menjalankan pod yang didalamnya terdapat kontainer aplikasi kita. Jadi, pod adalah abstraksi kubernetes untuk kontainer kita. Melalui abstraksi ini, kubernetes dapat me-*restart* kontainer jika terdapat masalah. Pod bersifat sementara, yang berarti ip dari pod berganti-ganti setiap kali dibuat oleh kubernetes. Jadi jika terdapat pod yang crash atau dimatikan secara sengaja lalu di-*restart*, ia akan mendapat ip yang baru. Sehingga kita tidak dapat mengandalkan ip pod untuk berkomunikasi. [18]

*Service* memungkinkan kita untuk mendapat suatu ip permanen yang dapat diakses dari *node* manapun dalam klaster. Jadi aplikasi kita tetap dapat berkomunikasi walaupun pod mati atau termuat ulang melalui ip dari service. Service juga berfungsi sebagai *load balancer* aplikasi kita yang berjalan sebagai pod. [19]

# **DOCKER**

## **Docker Secara umum**

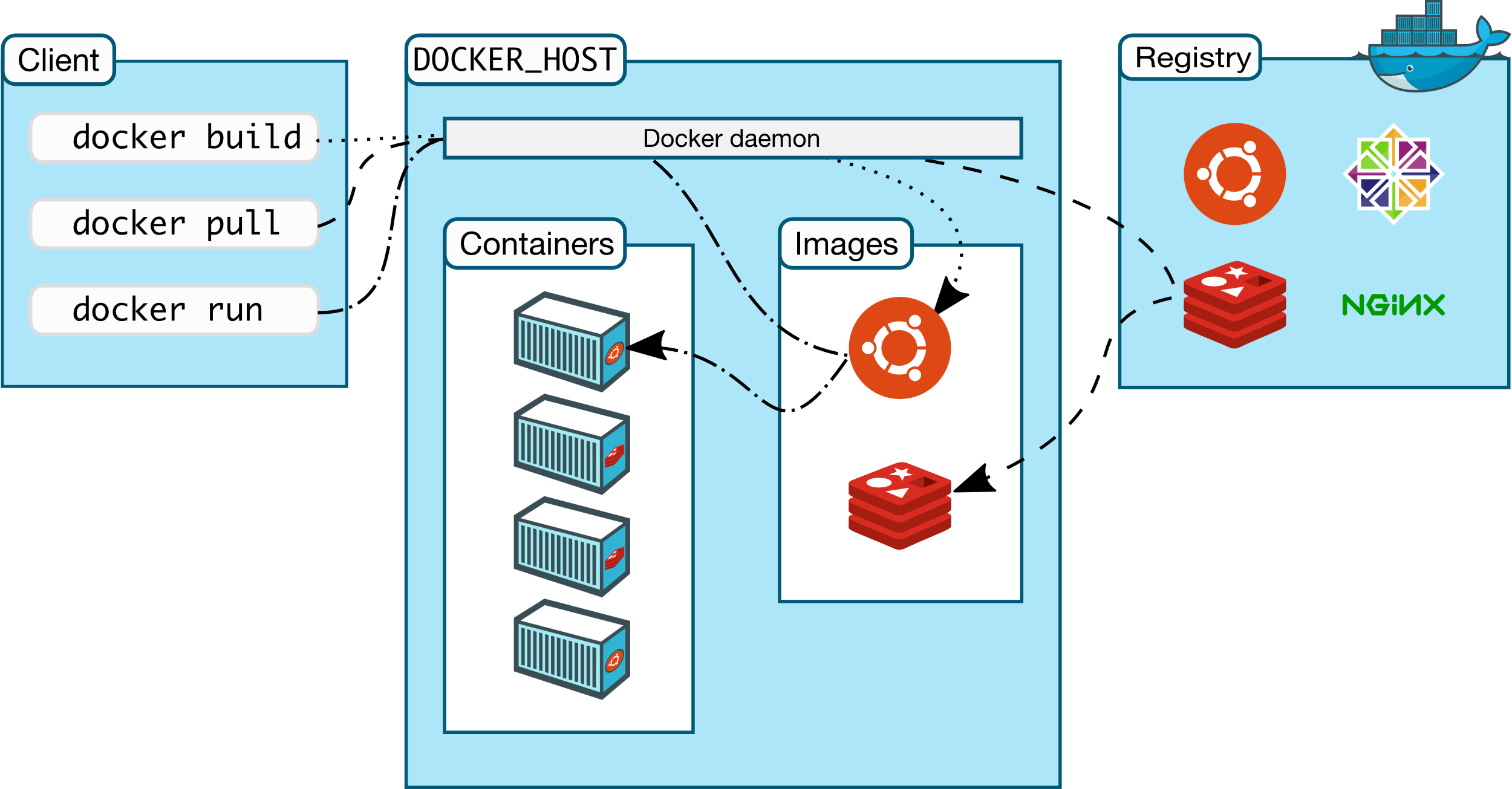
*Container* adalah sebuah standar untuk membungkus kode dan semua layanan pendukungnya menjadi suatu kesatuan yang dapat dijalankan dalam perangkat apapun dan di mana pun. Kontainer menjadi sebuah wadah untuk menjalankan suatu aplikasi atau simultan kernel sistem operasi dengan bantuan Operasi sistem host. [20][21]

Docker adalah sebuah platform terbuka untuk mengembangkan, mengirim, dan menjalankan aplikasi. Docker adalah implementasi dari *container manager*. Docker memiliki kemampuan untuk mem-*package* aplikasi men-*deploy* dan mendistribusikan aplikasi secara konsisten di *environment* manapun. Dengan adanya docker, aplikasi akan di-*package* dengan semua depensi dan konfigurasinya. Akibatnya artifak kode atau aplikasi akan menjadi lebih *portable*, lebih mudah dibagikan dan dijalankan ke siapa saja dan di mana saja. Ini akan menjadikan development dan deployment menjadi lebih mudah. [22]

Docker container berperan penting agar aplikasi kita dapat di-*package* di-*deploy* dan didistribusikan. Di dalam kontainer, biasanya terdapat konfigurasi sistem, software yang diperlukan, serta *startup script* untuk mengatur pengaturan tertentu atau menginisiasi tabel tertentu di database. Sehingga developer hanya perlu men-*download* kontainer agar pekerjaan menjadi lebih mudah. Docker container membuat kita untuk cukup memberi satu perintah untuk menginstall aplikasi. Docker container juga dapat membuat kita menjalankan aplikasi yang sama namun berbeda versi di komputer yang sama. [22]

## **Arsitektur Docker**

Basis arsitektur docker adalah *client* dan *server*. Pada saat kita meng-*install* docker, terdapat dua aplikasi yaitu *client*, yang akan kita gunakan, dan *server* digunakan untuk memanage docker. Docker client akan berkomunikasi dengan server (docker daemon). Seluruh perintah yang dimasukan ke dalam docker client akan dikirimkan ke docker daemon. Docker client dan docker daemon dapat dijalankan di satu sistem yang sama dan dapat berkomunikasi menggunakan REST API. [22]



Gambar Ilustrasi arsitektur Docker [22]

Docker Registry adalah tempat di mana kita menyimpan docker image. Dengan menggunakan docker registry, kita dapat menyimpan image yang kita buat dan bisa digunakan di docker daemon di manapun selama bisa terkoneksi ke docker registry. Salah satu fungsi registry adalah images terpusat. Sistem registry di docker hampir mirip dengan sistem git, kita bisa pull (mengambil image dari sumber lain) atau juga push (menambahkan image kita sendiri). [23][22]

Docker image mirip seperti installer aplikasi. Sebelum menjalankan aplikasi di docker, kita harus mendownload docker imagenya terlebih dahulu. Dalam docker image terdapat aplikasi dan dependensi. Docker image seperti aplikasi yang sudah siap digunakan. Sehingga docker image ibarat isi yang akan dijalankan pada sebuah container. [23][22]

Docker container merupakan kumpulan docker image. Docker container mirip seperti aplikasi hasil installernya (docker image). Docker image dapat digunakan untuk membuat beberapa docker container asalkan namanya berbeda. Jika telah membuat docker container, maka docker image yang sudah digunakan tidak dapat dihapus. Hal tersebut dikarenakan docker container hanya menggunakan isi dari docker image, tidak di-*copy*. [21]

Docker engine merupakan bagian dari docker yang berfungsi untuk menjalankan docker container. Docker engine akan menggunakan sistem operasi Linux untuk menjalankan perintahnya. [23]

Saat container dibuat di docker, secara default container akan saling terisolasi satu sama lain, sehingga kita tidak bisa memanggil antar container. Docker memiliki fitur Network yang dapat digunakan untuk membuat jaringan dalam docker. Dengan adanya Docker network kita dapat menghubungkan container lain dalam satu jaringan yang sama. Jika container berada pada jaringan yang sama, otomatis container dapat berkomunikasi satu sama lain. [24]

# **NAMED DATA NETWORKING (NDN)**

## **Named Data Netwok (NDN)**

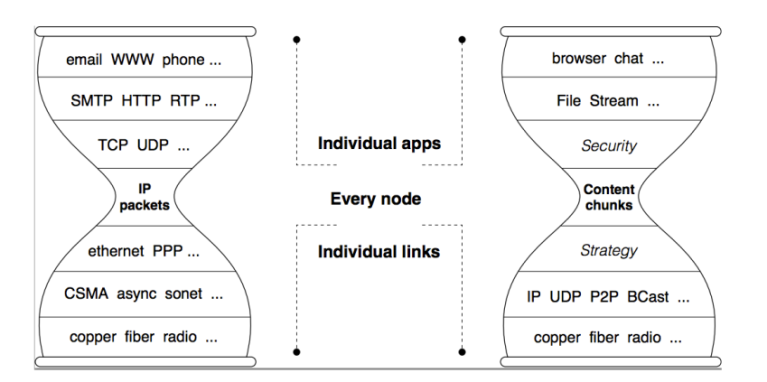
Jaringan dengan arsitektur ip adress dianggap sudah tidak efisien untuk memecahkan masalah kebutuhan komunikasi yang semakin kompleks karena masih menggunakan sistem infrastruktur end-to-end. Dengan menggunakan ip, permintaan user terhadap suatu data akan ditujukan kepada alamat sumber informasi tertent secara spesifik. Paket akan berlalu-lalang dalam jaringan server menuju user yang meminta data sehingga menimbulkan beban yang tidak perlu pada jaringan. [25][26]

Saat ini kita membutuhkan paradigma komunikasi baru yang mecakup komunikasi intensif terkait konten (*content lookup* dan *content caching*), *mobility*, *cloud*, serta *low delay* untuk mendukung jaringan 5g serta 6g. [25]

Named Data Networking (NDN) adalah arsitektur jaringan baru yang mana user tidak memerlukan ip address untuk mengakses sebuah konten pada internet, melainkan menggunakan nama konten. NDN memberikan nama pada data bukan pada hostnya. NDN disebut sebagai *content-centric network* atau jaringan yang berbasis konten. Request data pada NDN dilakukan berdasarkan konten bukan kepada server tertentu, sehingga respon terhadap request dapat dilakukan oleh router NDN manapun tidak hanya server (*producer*). [25]

## **Arsitektur**

Arsitektur internet saat ini menggunakan hourglass architecture center pada lapisan jaringan yang universal yaitu ip yang mengimplementasikan minimal fungsional yang diperlukan untuk interkoneksi global. Arsitektur NDN tetap mempertahankan bentuk hourglass yang sama pada bentuk sebelumnya namun, diubah sedikit untuk fokus pada data secara langsung bukan pada lokasi. [27][28]

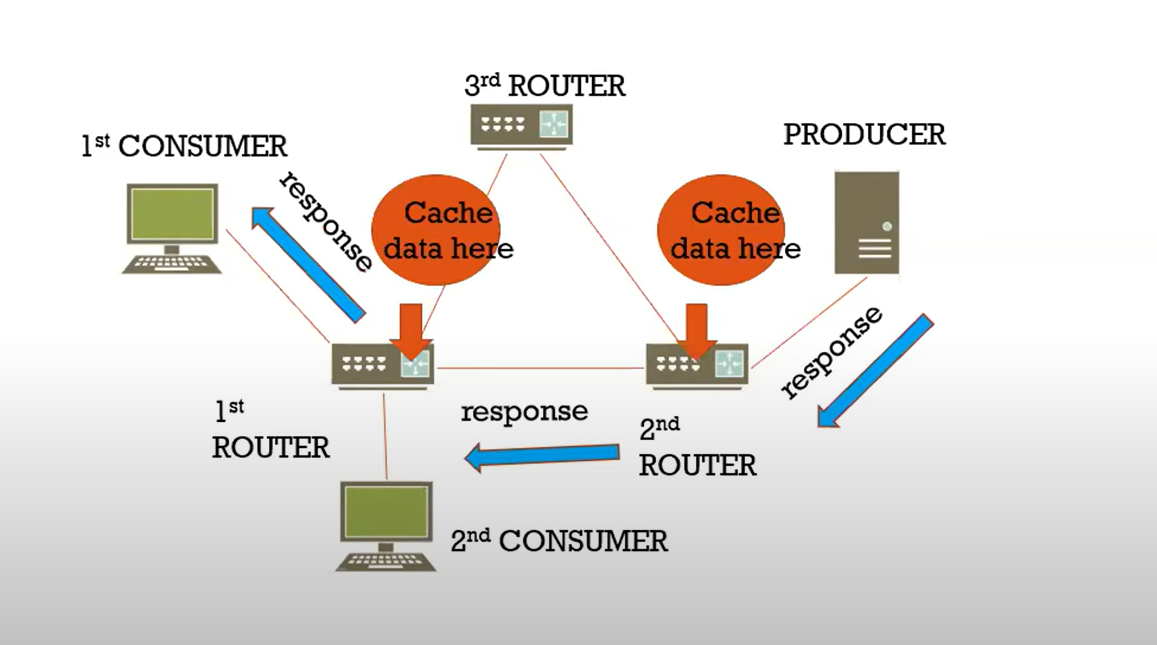


Gambar Arsitektur hourglass internet dan NDN [29]

Sama dengan arsitektur ip, Gambar 7 adalah pusat arsitektur dari NDN. Namun, NDN menggunakan named data bukan alamat ip untuk melakukan pengiriman. [27][28]

## **Cara Kerja**

Ketika user pertama meminta data, maka permintaan tersebut akan diteruskan ke router pertama. Setelah men-*generate* *table routing*, request akan dikirim melalui router kedua, ketiga dan seterusnya hingga ke producer. Producer akan menerima permintaan tersebut dan merespon data yang diminta. Kemudian, selama perjalanan kembali melalui router tiga, dua hingga pertama data akan di-cache (disimpan datanya) kemudian akan diforward ke router selanjutnya dan akan mengalami proses penyimpanan data yang serupa hingga menuju user.



Gambar ilustrasi mekanisme kerja NDN [25]

Saat user kedua meminta data yang sama dengan user pertama tadi, router pertama akan menerima request dari user kedua dan mengecek apakah router tersebut memiliki data yang diminta. Router pertama sudah memiliki data yang diminta user karena telah men-cache data saat melakukan forwarding respons data user pertama. Router pertama akan merespon permintaan user kedua dan mengirim data ke user kedua tanpa harus melalui producer lagi.

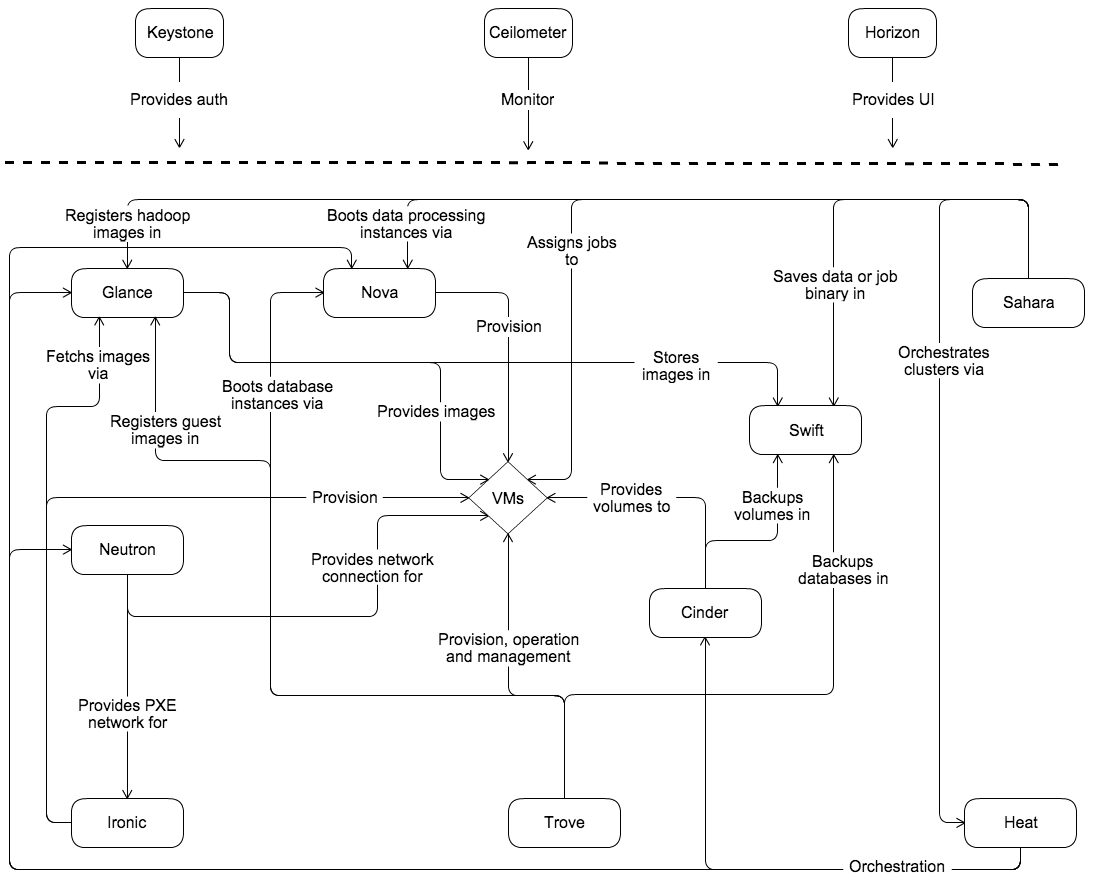
# **OPENSTACK**

## **Definisi**

OpenStack adalah sebuah layanan sistem operasi cloud terbuka yang dapat digunakan untuk memanajemen sumber daya komputasi yang besar, penyimpanan dan jaringan di semua data center dalam satu tampilan dashboard. Administrator atau pengguna dapat mengendalikan dan melakukan *provisioning* atas sumber daya ini melalu antarmuka. [30][31]

## **Arsitektur**

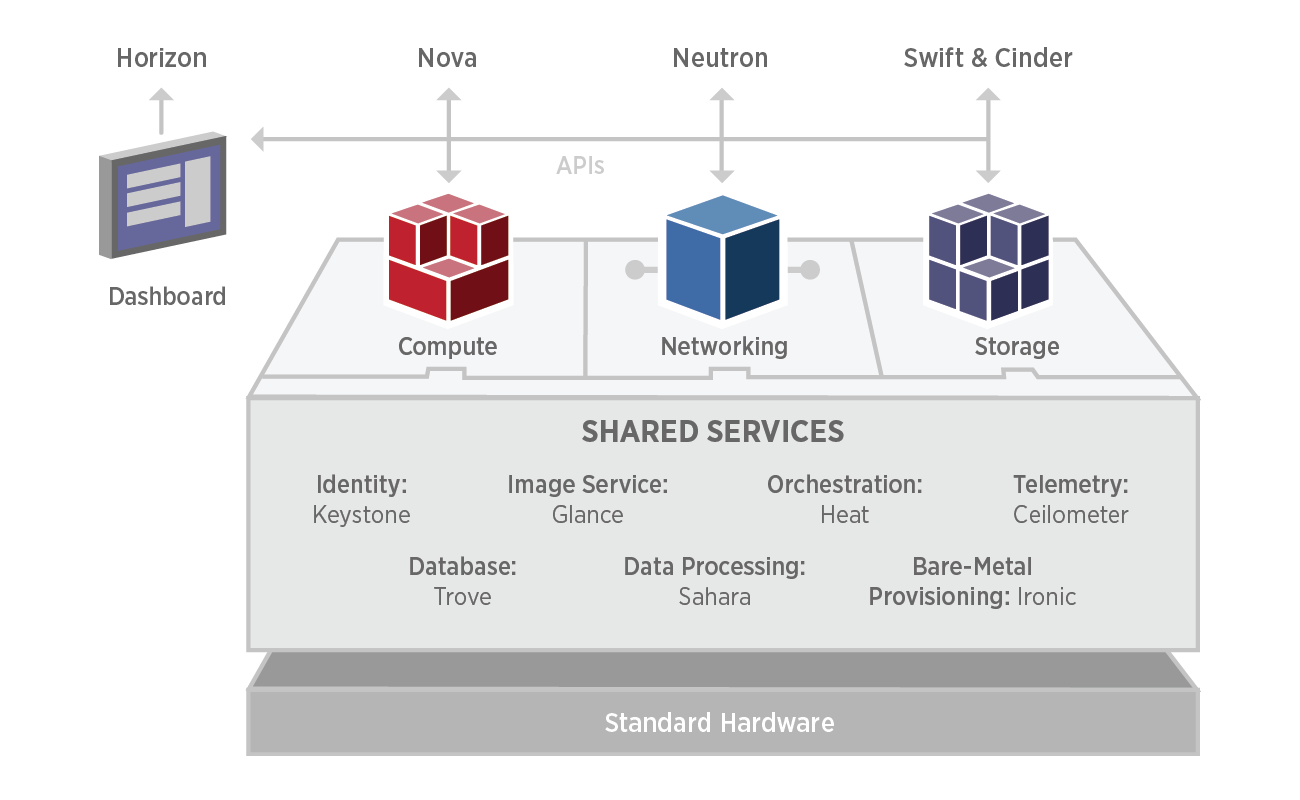
OpenStack menganut arsitektur modular untuk menyediakan satu set layanan inti yang memfasilitasi skalabilitas dan elastisitas sebagai prinsip desain inti. OpenStack terdiri dari bagian bagian independen yang disebut OpenStack Services (layanan). Tiap layanan akan saling berinteraksi melalui API (publik atau via auth). [31]



Gambar Hubungan antara layanan OpenStack [31]

## **Layanan OpenStack**

OpenStack menyediakan solusi dari IaaS (Infrastructure ad a Service) dengan berbagai layanan yang berhubungan. Masing-masing layanan menyediakan API yang memfasilitasi integrasi. [32]



Gambar Gambaran layanan pada OpenStack [32]

1. **Dashboard/Horizon**

Menyediakan tampilan web yang berinteraksi dengan layanan OpenStack seperti meluncurkan instance, memberikan ip address dan mengkonfigurasi akses kontrol untuk Nova, Neutron, Swift, dll.

1. **Compute/Nova**

Nova merupakan kontroler sumber-daya compute yg menjadi komponen utama dari sistem IaaS, karena entitas ini yg mengatur proses dan alokasi CPU untuk setiap VM.

1. **Networking/Neutron**

Fungsi utama Neutron adalah untuk menyediakan Network connectivity as a service untuk servis OpenStack lain seperti compute (Nova).

1. **Object Storage/Swift**

Swift bertugas menympan dan menambil object data unstructured via API.

1. **Block Storage/Cinder**

Cinder menyediakan persistent block storage untuk menjalankan instance.

1. **Identity Service/Keystone**

Menyediakan manajemen otorisasi layanan.

1. **Image Service/Glance**

Glance bertugas untuk menyimpan dan mengambil image VM. Komponen compute menggunakan servis ini pada saat provisioning instance.

1. **Telementery/Ceilometer**

Ceilometer akan memantau billing, benchmarking, scalability dan tujuan statistik.

1. **Orchestration/Heat**

Berfngsi untuk menggabungkan beberapa aplikasi cloud.

1. **Database Service/Trove**

Menyediakan layanan cloud database-as-a-service.

1. **Data Processing Service**

Menyediakan kemampuan provision untuk Hadoop clusters.

# **DAFTAR PUSTAKA**

[1] Sumarno, “ANALISA RANCANG BANGUN TURBIN TENAGA MAGNET SEDERHANA SEBAGAI SUMBER LISTRIK SKALA RUMAH TANGGA Sumarno Program Studi Teknik Mesin , Fakultas Teknik Universitas Muhammadiyah Tangerang Jl . Perintis Kemerdekaan I / 33 , Cikokol , Kota Tangerang,” 2012.

[2] S. B. Kartika, “Modul Praktikum Fisika Dasar,” *Modul Prakt. Fis. Dasar*, 2016, doi: 10.21070/2016/978-979-3401-82-9.

[3] A. Kurniawan, “Induksi adalah Munculnya Arus Listrik Akibat Perubahan Medan Magnet, Ini Ulasannya,” *www.merdeka.com*, 2022. https://www.merdeka.com/jabar/induksi-adalah-munculnya-arus-listrik-akibat-perubahan-medan-magnet-ini-ulasannya-kln.html (accessed May 23, 2023).

[4] J. Owen, “How does an Electric Motor work? (DC Motor),” *Youtube*, 2021. https://youtu.be/CWulQ1ZSE3c

[5] A. Muhardian, “Belajar Pemrograman Python: Pengenalan Dasar Python dan Persiapan Awal,” *Petanikode*, 2018. https://www.petanikode.com/python-linux/ (accessed Jan. 25, 2023).

[6] S. Nugroho, B. Pujiarto, U. M. Magelang, and P. Korespondensi, “Network Automation Pada Beberapa Perangkat Router Network Automation in Some Router Devices,” *J. Teknol. Inf. dan Ilmu Komput.*, vol. 9, no. 1, pp. 79–86, 2022, doi: 10.25126/jtiik.202293947.

[7] Fitri, K. R. R, A. Rahmansyah, and W. Darwin, “Penggunaan Bahasa Pemrograman Python Sebagai Pusat Kendali Pada Robot 10-D,” *5th Indones. Symp. Robot. Syst. Control*, pp. 23–26, 2017.

[8] R. M. R. Clinton and S. Sengkey, “Purwarupa Sistem Daftar Pelanggaran Lalulintas,” *J. Tek. Elektro dan Komput. Vol.8*, vol. 8, no. 3, pp. 181–192, 2019.

[9] S. Miftah, “Library Python Kenali Perbedaan Module, Package, dan Library Pada Python,” *DQLab.id*, 2021. https://www.dqlab.id/library-python-kenali-perbedaan-module-package-dan-library-pada-python (accessed Jan. 25, 2023).

[10] “Python Tutorial,” *GeeksforGeeks*, 2022. https://www.geeksforgeeks.org/python-programming-language/learn-python-tutorial/ (accessed Jan. 26, 2023).

[11] “#7 Software Pendukung,” *SMK 1 Lubuk sikaping*. https://smkn1lubuksikaping.sch.id/learning/courses-lesson/pemrograman-dasar/9/17#:~:text=IDE ( Integrated Development Environment ) merupakan,program aplikasi yang sudah dibuat. (accessed Jan. 26, 2023).

[12] “Indentation in Python,” *GeeksforGeeks*, 2022. https://www.geeksforgeeks.org/indentation-in-python/ (accessed Jan. 26, 2023).

[13] Smarvin, “Kubenernetes Documentation - Concepts,” *kubernetes.io*, 2019. https://kubernetes.io/docs/concepts/overview/ (accessed Jan. 28, 2023).

[14] M. Beznos, “Microservices vs monolith: Which architecture is the best choice for your business?,” *N-iX*, 2023. https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business/#:~:text=While a monolithic application is,well as perform specific functions. (accessed Jan. 28, 2023).

[15] Veerash Ayyagari, “Orkestrasi kontainer untuk layanan mikro,” *Microsoft*. https://learn.microsoft.com/id-id/azure/architecture/microservices/design/orchestration (accessed Jan. 28, 2023).

[16] “Konsep-konsep di balik Controller Manager,” *kubernetes.io*, 2020. https://kubernetes.io/id/docs/concepts/architecture/cloud-controller/ (accessed Jan. 28, 2023).

[17] “Komponen-Komponen Kubernetes,” *kubernetes.io*, 2020. https://kubernetes.io/id/docs/concepts/overview/components/ (accessed Jan. 28, 2023).

[18] “Pod,” *kubernetes.io*, 2020. https://kubernetes.io/id/docs/concepts/workloads/pods/pod/ (accessed Jan. 28, 2023).

[19] “Service,” *kubernetes.io192*. https://kubernetes.io/id/docs/concepts/services-networking/service/ (accessed Jan. 29, 2023).

[20] “Use containers to Build, Share and Run your applications,” *docker.com*. https://www.docker.com/resources/what-container/ (accessed Jan. 31, 2023).

[21] A. D. Nugraha, “Belajar Docker — Mengenal Apa Itu Docker dan Container, Container vs Virtual Machine, Fungsi Container,” *Medium*, 2020. https://viandwi24.medium.com/belajar-docker-mengenal-apa-itu-docker-dan-container-container-vs-virtual-machine-3c20d7288224 (accessed Jan. 31, 2023).

[22] “Docker Overview,” *Docker Docs*. https://docs.docker.com/get-started/overview/ (accessed Jan. 31, 2023).

[23] A. D. Nugraha, “Belajar Docker — Mengenal Apa Itu Registry, Images, Docker Host, Docker Client, Docker Engine,” *Medium*, 2020. https://viandwi24.medium.com/belajar-docker-mengenal-apa-itu-registry-images-docker-host-docker-client-docker-engine-4e085baa22c4 (accessed Jan. 31, 2023).

[24] E. K. Khannedy, “Docker Dasar,” 2021. https://docs.google.com/presentation/d/1LoCIoqR68t-y7P7eOs\_TVoooZy4mq-tc2cwInQAtfy0/edit#slide=id.gcfdc6d4495\_0\_377 (accessed Jan. 31, 2023).

[25] L. V. Yovita, “Named Data Network as a Future Internet,” *Youtube : SG NDN Telkom University*, 2021. https://www.youtube.com/watch?v=\_s4OhMArH54&t=1437s (accessed Jan. 05, 2023).

[26] R. Novianti, “SIMULASI PERFORMANSI TOPOLOGI MENGGUNAKAN NAMED DATA NETWORK PADA DISTRIBUSI DATA DENGAN SIMULATOR NDN-SIM,” *Proy. Akhir Inst. Teknol. Telkom Jakarta*, 2022.

[27] S. Ahdan, “Mengenal Arsitektur Named Data Networking,” 2021. https://syaifulahdan.wordpress.com/2020/09/24/mengenal-arsitektur-named-data-networking/ (accessed Jan. 07, 2023).

[28] S. Ahdan, “Overview Keamanan pada Jaringan NDN (Named Data Networking),” *Tugas Akhir Mata Kuliah Keamanan Inf. dan Jar. Inst. Teknol. Bandung*, 2016.

[29] L. Zhang *et al.*, “Named Data Networking (NDN) Project 2013-2014 Report”.

[30] M. S. S. R, R. R. P, and I. Yusup, “OPENSTACK ESSENTIALS,” *Tek. Komput. Politek. Sukabumi*, 2018.

[31] E. Mulyana, “Pengantar OpenStack,” *Telematika.ORG*, 2017. https://www.telematika.org/post/pengantar-openstack/ (accessed Feb. 12, 2023).

[32] D. D. Prasetyo, “Menganal OpenStack Untuk Cloud Computing,” *Algostudio Community*, 2020. https://community.algostudio.net/menganal-openstack-untuk-cloud-computing/ (accessed Feb. 12, 2023).

*Halaman ini sengaja dikosongkan*