UG-AI-CoE

Melakukan Deployment Model (Pipelining, Teknologi Deployment, Cara Deployment)

PERTEMUAN KE - 5

Course Definition

- Melakukan Deployment Model
- Unit Kompetensi:

J.62DMI00.016.1 - Membuat Rencana Deployment Model

- Melakukan strategi deployment
- Menyusun instruksi deployment

J.62DMI00.017.1 - Melakukan Deployment Model

- Melakukan langkah deployment sesuai instruksi
- Membuat laporan hasil deployment

Learning Objective

Dalam pelatihan ini diharapkan:

Peserta mampu melakukan deployment model menjadi sistem yang dapat dioperasikan (website)

Course Sub-topic

- Pendahuluan
- Penerapan model berbasis pola
- Strategi deployment model
- Deployment model (Regresi Linier)
- Deployment model (ANN)

Referensi: SKKNI Data Science

KODE UNIT : **J.62DMI00.016.1**
JUDUL UNIT : **Membuat Rencana *Deployment Model***
DESKRIPSI UNIT : Unit kompetensi ini berhubungan dengan pengetahuan, keterampilan, dan sikap kerja yang dibutuhkan dalam membuat perencanaan *deployment model*.

ELEMEN KOMPETENSI	KRITERIA UNJUK KERJA
1. Menentukan strategi *deployment*	1.1 ***Existing system*** terkait yang sudah dimiliki organisasi diidentifikasi. 1.2 Parameter keberhasilan *deployment* disusun berdasarkan standar yang berlaku. 1.3 Diskrepansi/perbedaan **konfigurasi** antara **lingkungan pengembangan** dengan **lingkungan *deployment*** dianalisis sesuai standar yang berlaku. 1.4 **Strategi *deployment*** ditentukan berdasarkan **parameter keberhasilan** dan hasil analisis diskrepansi/perbedaan. 1.5 *Tools* untuk *deployment* diidentifikasi.
2. Menyusun instruksi *deployment*	2.1 **Instruksi *deployment*** dibuat sesuai strategi *deployment*. 2.2 **Mekanisme pemantauan deployment** disusun sesuai strategi *deployment*. 2.3 Instruksi *deployment* didokumentasikan sesuai standar yang berlaku.

1. Konteks variabel
   1.1 *Existing system* adalah sistem yang akan menggunakan hasil dari model yang sudah dibangun.
   1.2 Konfigurasi adalah serangkaian parameter teknis yang menyusun sebuah lingkungan sistem berbasis *Information Technology* (IT). Contoh konfigurasi adalah sistem operasi yang digunakan, jenis manajemen basis data yang digunakan, bahasa pemrograman yang digunakan.
   1.3 Lingkungan pengembangan adalah sebuah lokasi sistem berbasis IT yang dikhususkan untuk pembuatan aplikasi atau sistem berbasis IT.
   1.4 Lingkungan *deployment* adalah sebuah lokasi sistem berbasis IT yang dikhususkan untuk penempatan aplikasi atau sistem berbasis IT yang sudah siap pakai.
   1.5 Strategi *deployment* adalah serangkaian arahan dalam pelaksanaan *deployment* untuk memastikan terpenuhinya parameter keberhasilan *deployment*.
   1.6 Instruksi *deployment* adalah serangkaian tata cara pelaksanaan *deployment* dan alternatifnya yang sesuai dengan strategi *deployment* serta sudah mempertimbangkan kemungkinan-kemungkinan yang menghalangi terpenuhinya parameter keberhasilan *deployment*.
   1.7 Parameter keberhasilan adalah ukuran kuantitatif yang dapat diukur secara objektif yang harus dipenuhi dalam pelaksanaan *deployment*. Contoh dari parameter keberhasilan adalah tidak adanya penurunan kinerja sistem, waktu respons sistem tetap stabil selama *deployment*, tidak ada kegagalan yang merembet ke sistem lain selama atau karena efek *deployment*, dan lain-lain.
   1.8 Mekanisme pemantauan *deployment* adalah tata cara terurut dan terkuantifikasi yang mendefinisikan hal-hal yang perlu dipantau dari sistem.

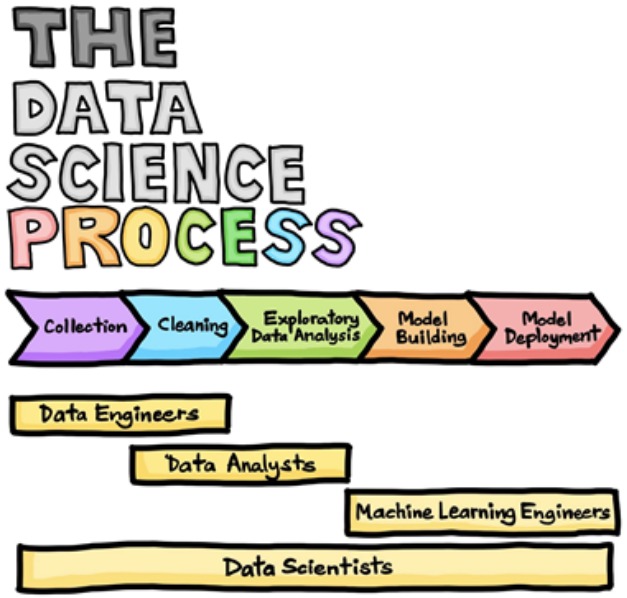
Referensi: SKKNI Data Science

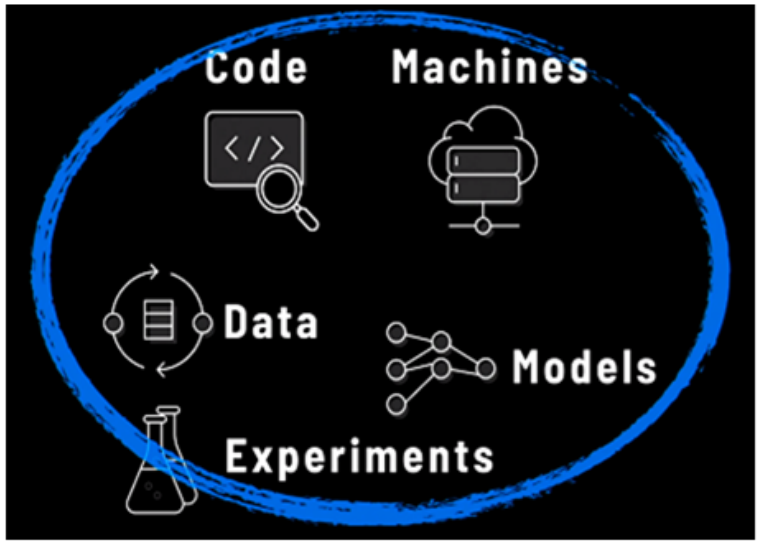
KODE UNIT : **J.62DMI00.017.1**
JUDUL UNIT : **Melakukan *Deployment Model***
DESKRIPSI UNIT : Unit kompetensi ini berhubungan dengan pengetahuan, keterampilan, dan sikap kerja yang dibutuhkan dalam melakukan *deployment model*.

ELEMEN KOMPETENSI	KRITERIA UNJUK KERJA
1. Melakukan langkah *deployment* sesuai instruksi	1.1 Prasyarat **lingkungan *deployment*** disiapkan sesuai strategi *deployment*. 1.2 *Deployment* dilaksanakan sesuai **instruksi *deployment*.**
2. Membuat laporan hasil *deployment*	2.1 **Keluaran sistem** sebagai efek dari *deployment* didokumentasikan. 2.2 Masalah yang timbul setelah pemodelan karena efek *deployment* didokumentasikan. 2.3 Hasil proses *deployment* didokumentasikan sesuai dengan standar yang berlaku.

1. Konteks variabel
   1.1 Lingkungan *deployment* adalah sistem yang memiliki konfigurasi tertentu yang sesuai dengan kebutuhan bisnis.
   1.2 Instruksi *deployment* adalah serangkaian tata cara pelaksanaan *deployment* dan alternatifnya yang sesuai dengan strategi *deployment* serta sudah mempertimbangkan kemungkinan-kemungkinan yang menghalangi terpenuhinya parameter keberhasilan *deployment*.
   1.3 Keluaran sistem adalah sebuah ukuran kuantitatif yang dapat diamati secara objektif sebagai hasil kerja sistem.

Pertimbangan Penting





Cases
& Value

Data
Ecosystems

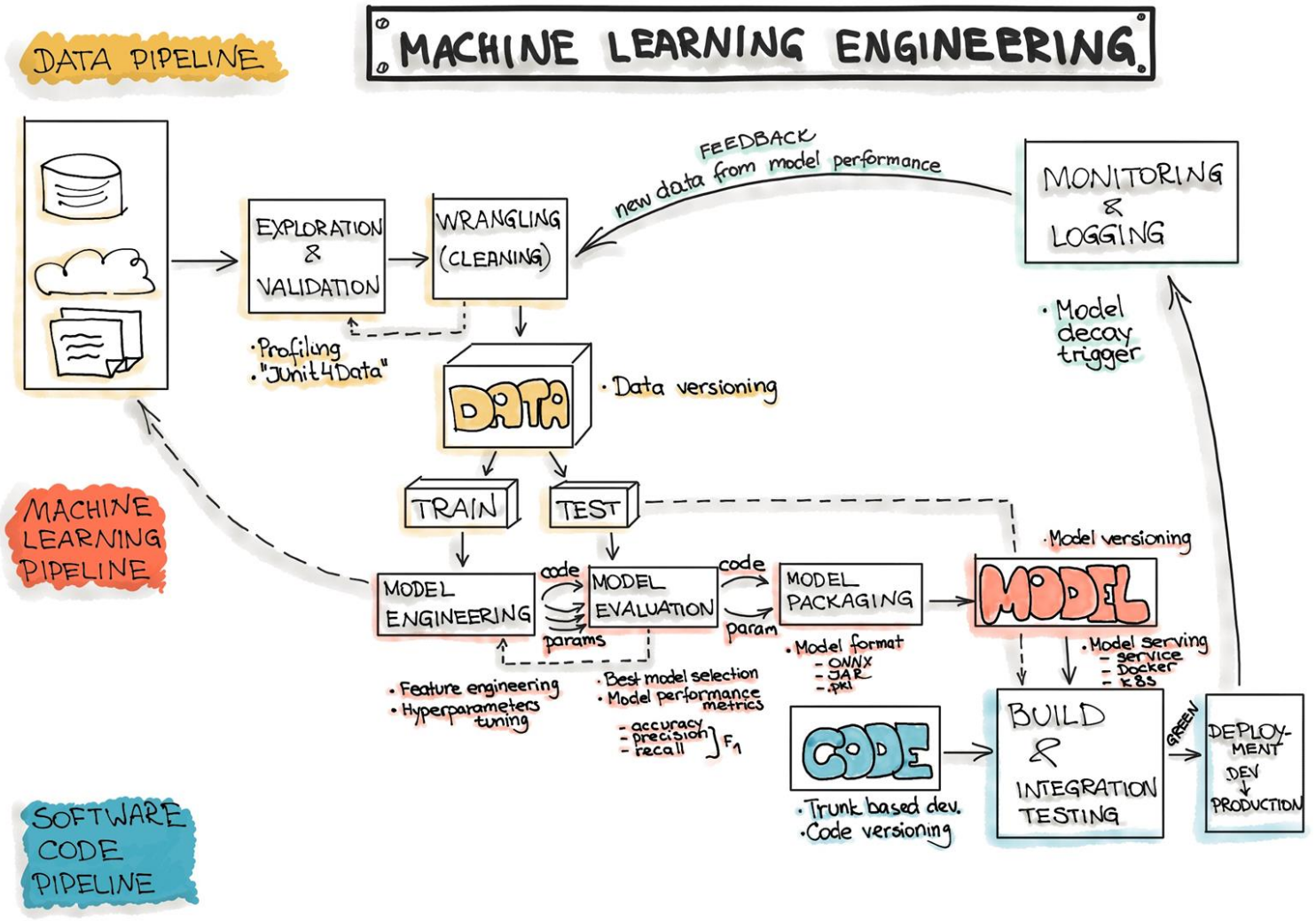
Techniques
& Tools

Workflow
Integration

Open
Culture

Deployments

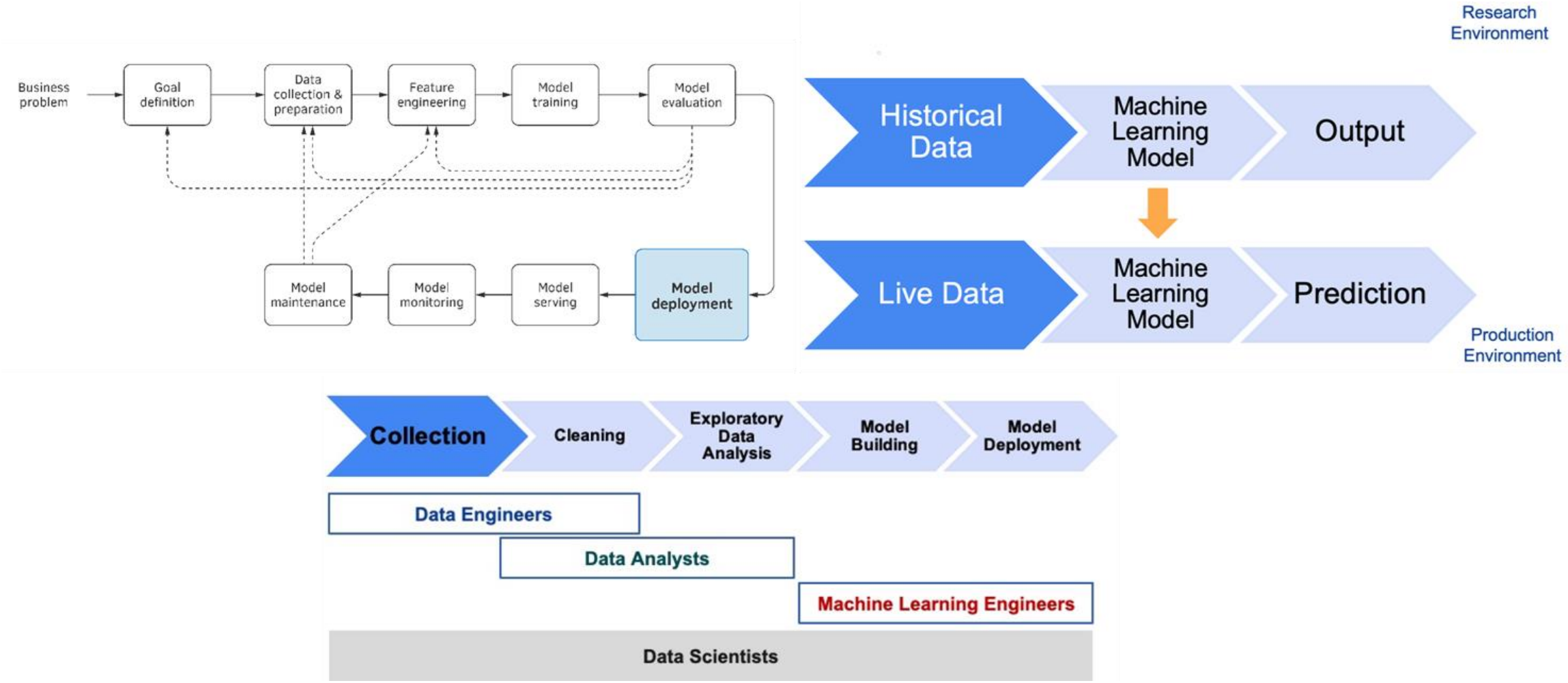
Pendahuluan (1/6)



Pendahuluan (2/6)

- **Deployment model** adalah suatu proses untuk membuat **model** (model machine learning) tersebut **tersedia** pada **lingkungan produksi**, dimana model tersebut **dapat memberikan prediksi ke sistem** perangkat lunak yang **lain**.
- Deployment model merupakan **tahapan terakhir** pada Machine Learning lifecycle dan merupakan tahapan yang paling menantang.

Pendahuluan (3/6)



Pendahuluan (4/6)

Deployment model **sangat penting**:

- Untuk menggunakan model Machine Learning, **penerapan secara efektif** model tersebut ke dalam lingkungan produksi sangat diperlukan. Hal ini ditujukan agar model tersebut dapat memberikan prediksi ke sistem perangkat lunak yang lain.
- Mengekstraksi prediksi dengan handal dan membagikan ke dalam sistem yang lain sangat penting dilakukan untuk **memaksimalkan nilai model** machine learning yang telah dibuat.

Pendahuluan (5/6)

Tantangan pada implementasi Deployment model:

- Tantangan pada perangkat lunak biasa:
  - Reliability
  - Reusability
  - Maintainability
  - Flexibility
- Tantangan spesifik pada Machine Learning:
  - Reproducibility

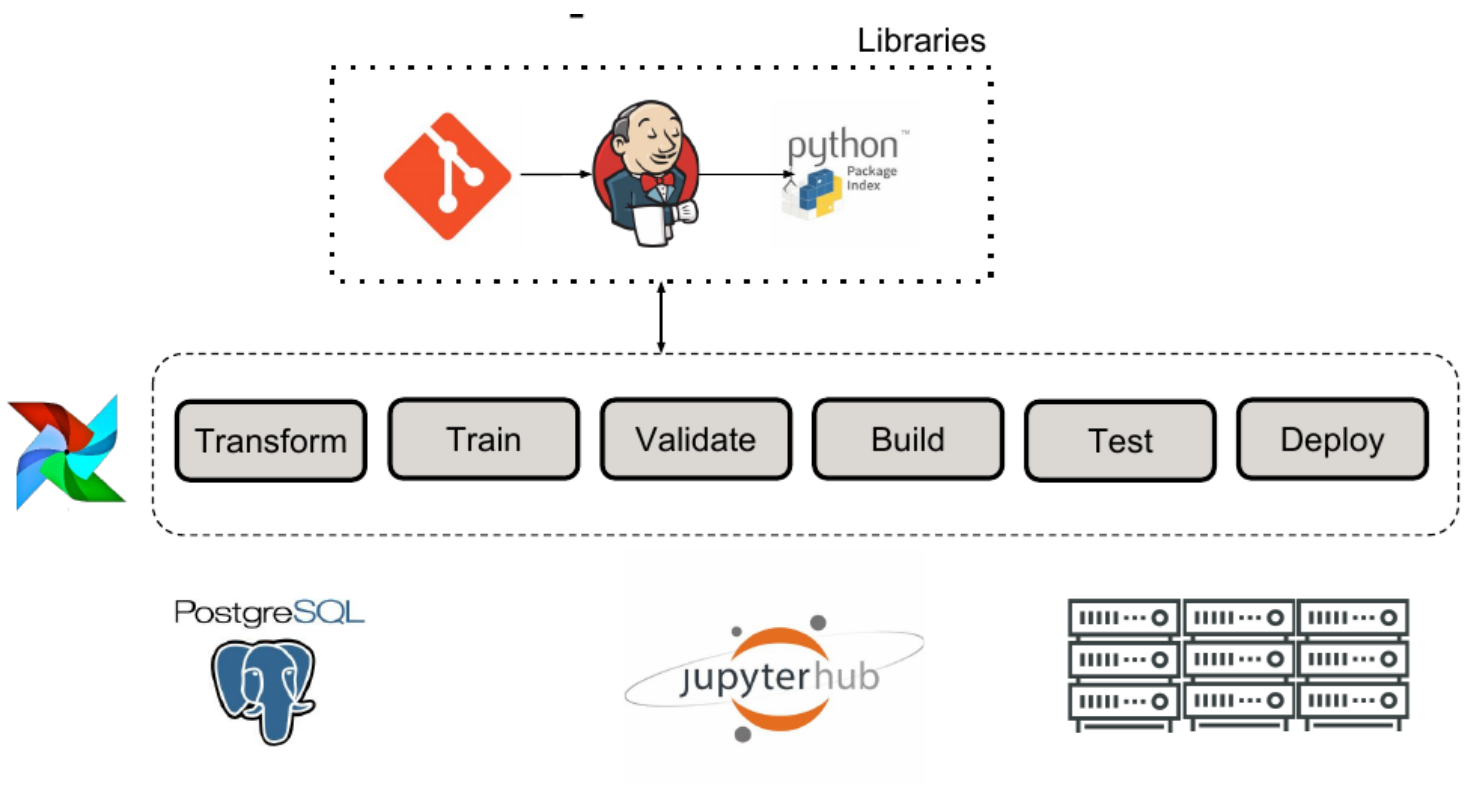
Pendahuluan (6/6)

Tantangan pada implementasi Deployment model:

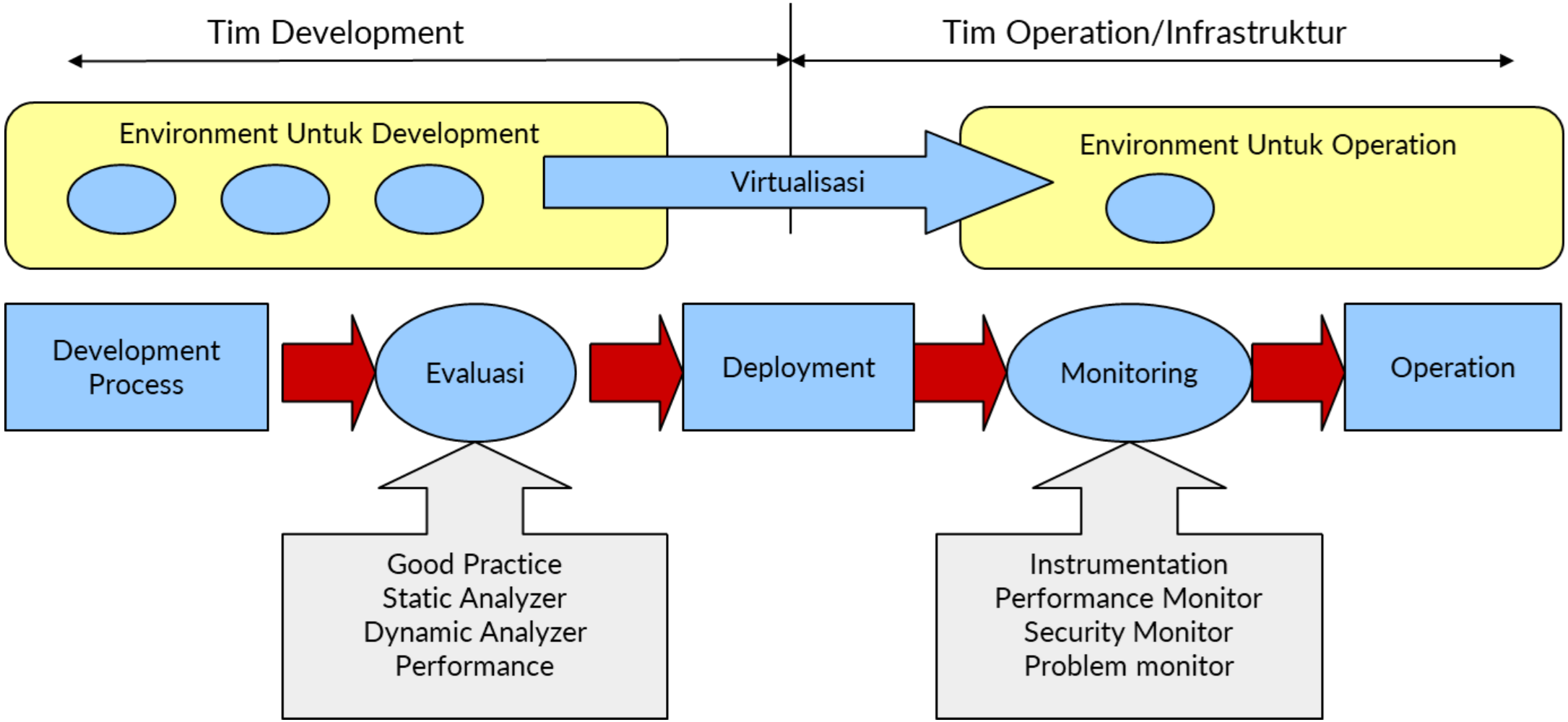
- Memerlukan koordinasi antara Data Scientists, IT Teams, software developers, dan profesional bisnis:
  - Memastikan model bekerja secara reliable
  - Memastikan model mendeliver keluaran yang dimaksudkan.
- Potensi perbedaan antara bahasa pemrograman yang digunakan dalam mengembangkan model dengan bahasa yang digunakan pada sistem produksi.
  - Melakukan coding ulang akan memperpanjang durasi pengerjaan proyek dan mengurangi reproduktifitas.

Tools utk Data Science

- Perangkat database digunakan untuk menyuplai data, integrasi data dan sebagainya
- Sedangkan perangkat seperti Python dengan Jupyter Notebook digunakan untuk membangun model, mencoba algoritma
- Ketika akan di deploy maka dibutuhkan perangkat bantu tambahan untuk melakukan deployment

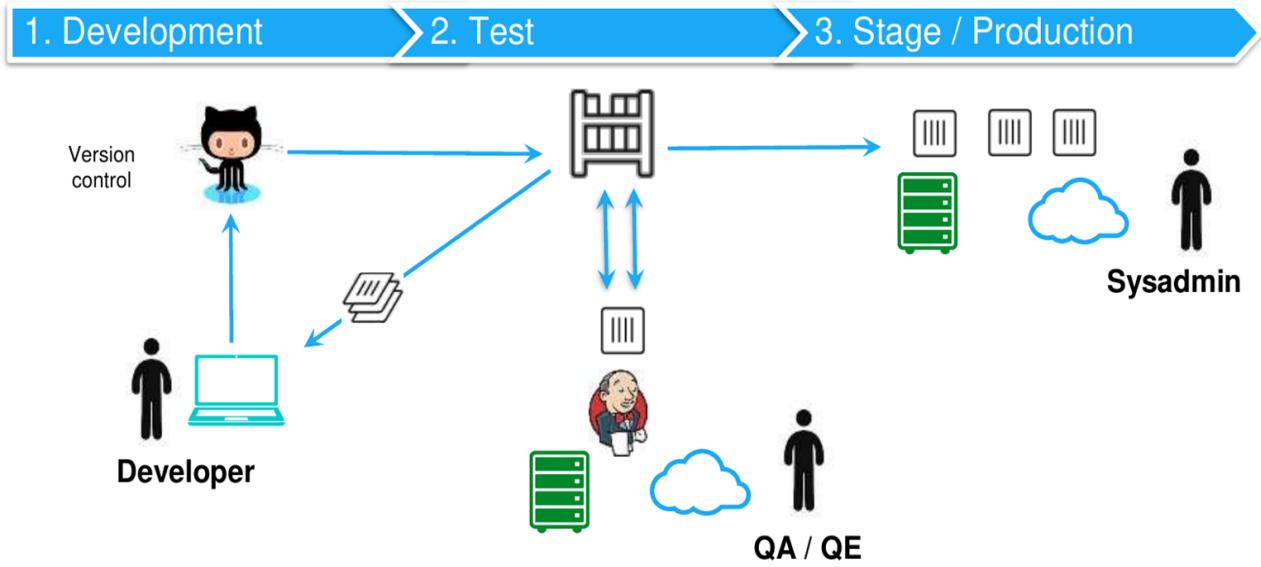


Continue Development

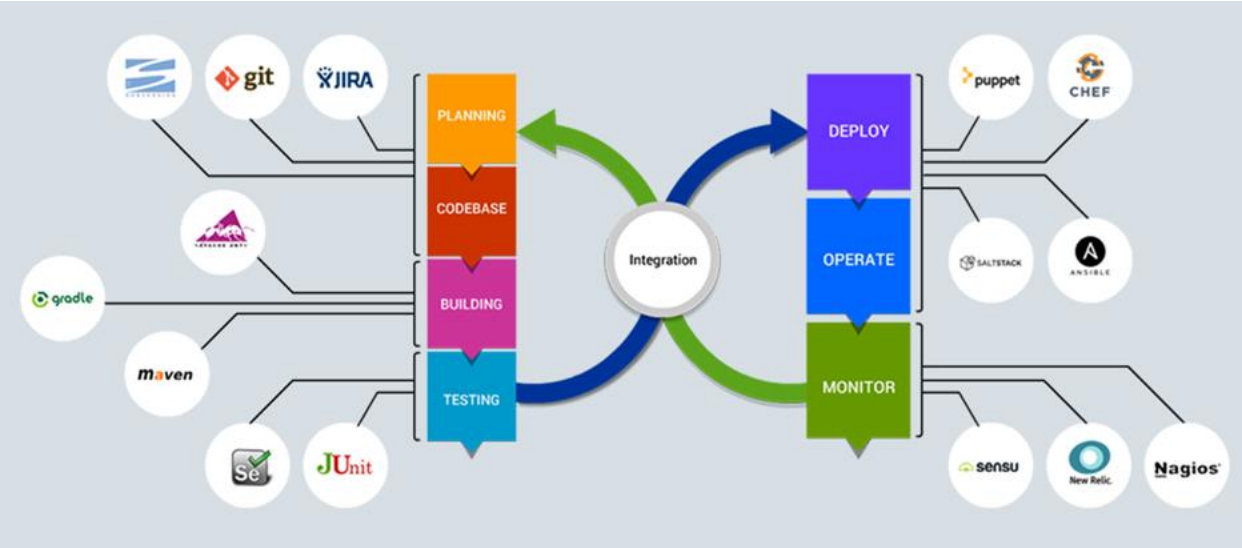


CI/CD untuk Data Science

1. Build the model
    1. Build the model artifacts
    2. Send the artifacts to long-term storage
    3. Run basic checks (smoke tests/sanity checks)
    4. Generate fairness and explainability reports
2. Deploy to a test environment
    1. Run tests to validate ML performance, computational performance
    2. Validate manually
3. Deploy to production environment
    1. Deploy the model as canary
    2. Fully deploy the model



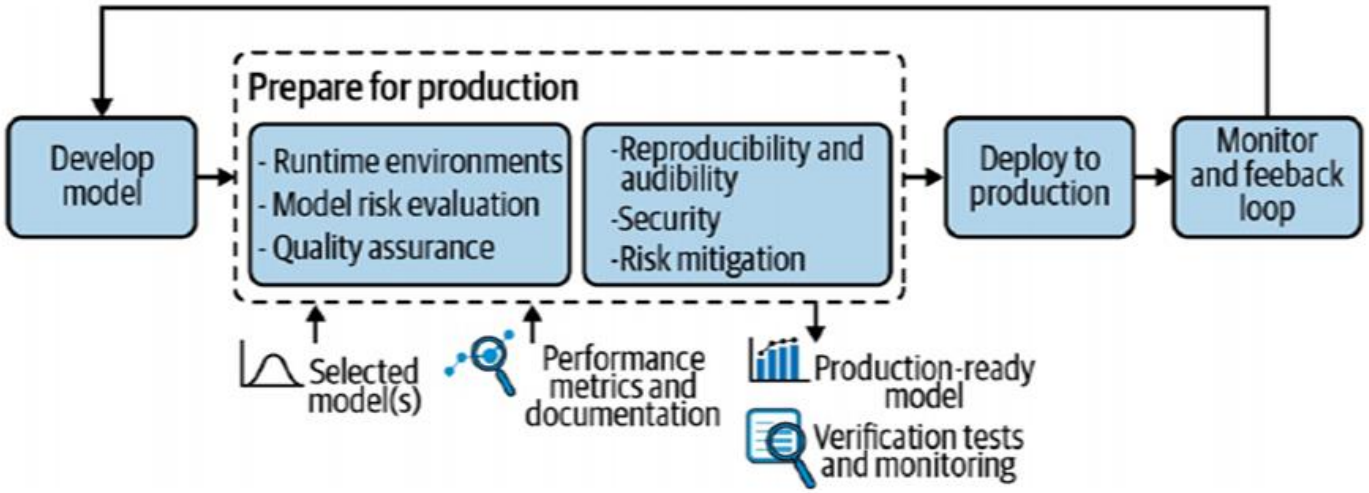
Tools untuk DevOps



- Untuk lingkungan deployment yang menerapkan DevOps maka digunakan beberapa perangkat bantu
- Untuk proses development digunakan beberapa perangkat bantu misal junit, git dsb
- Sedangkan untuk melakukan deployment dan operasi dilakukan perangkat bantu lainnya
- Perangkat bantu ini harus terintegrasi sehingga bisa melakukan deployment secara continues

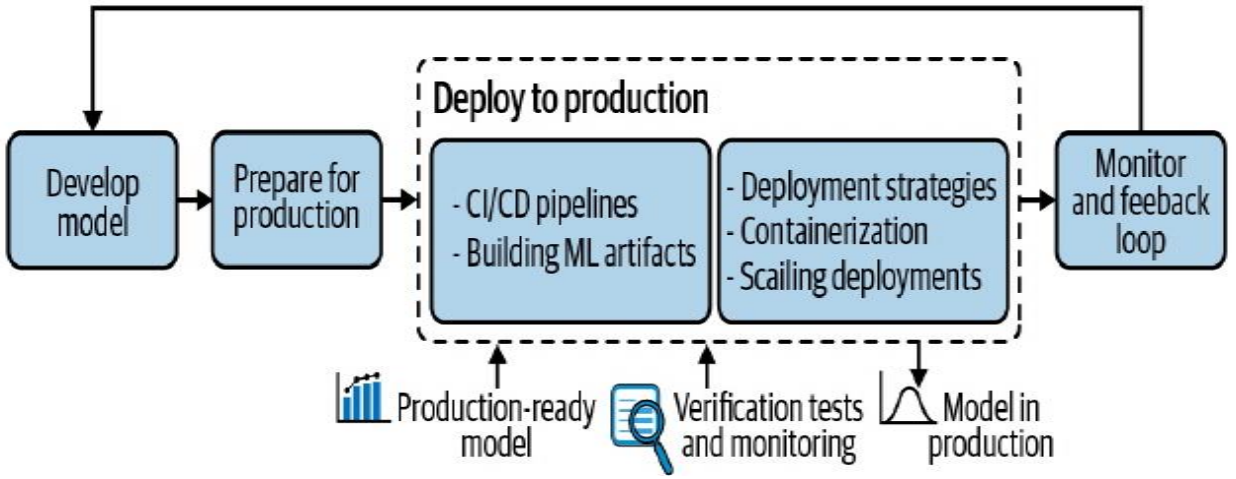
Preparing for production

- Model yang sudah jadi harus dipersiapkan untuk produksi
- Dokumentasi-dokumentasi saat membangun model merupakan hal yang membantu proses deployment

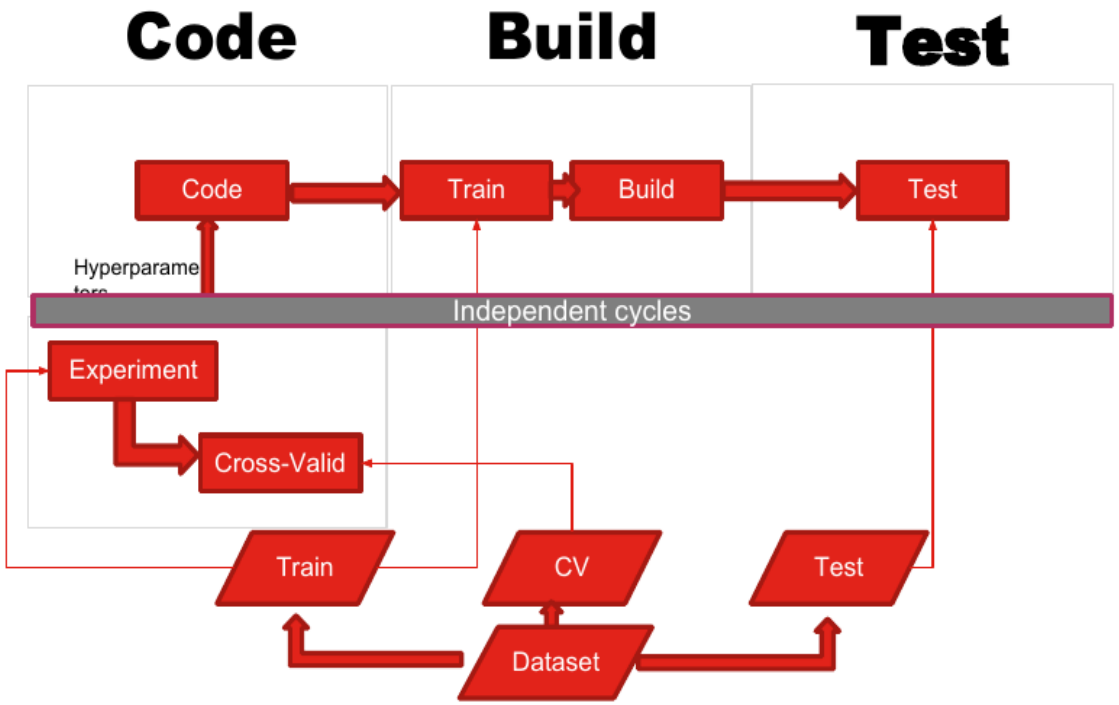


Deploy to production

- Pada saat di deploy maka dilakukan proses CI/CD
- Proses ini menggunakan model container sehingga memudahkan dalam deployment dan skalabilitas

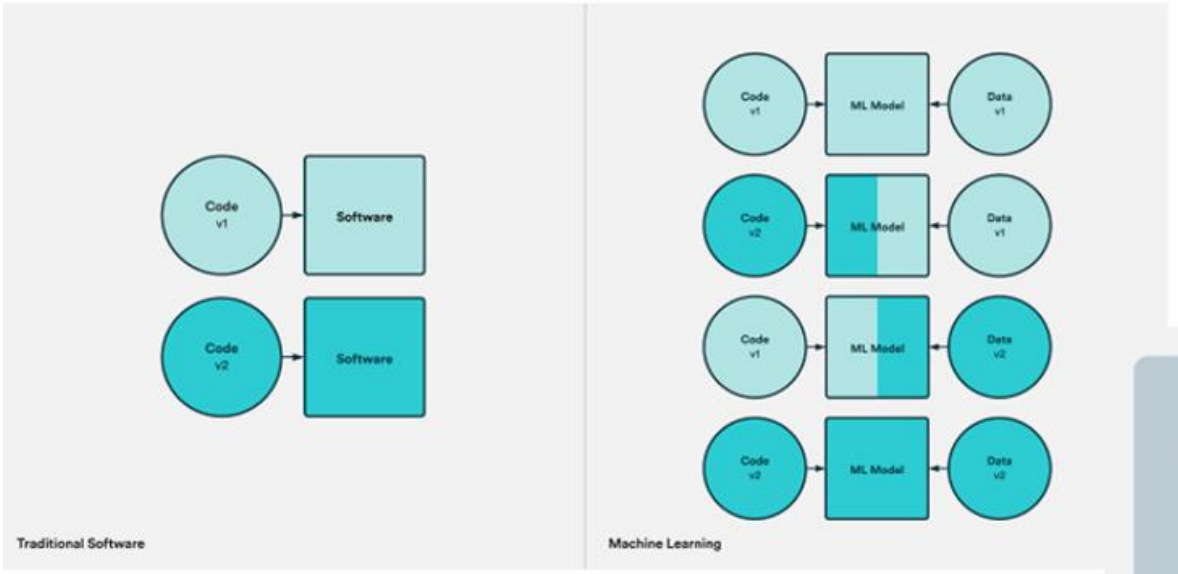


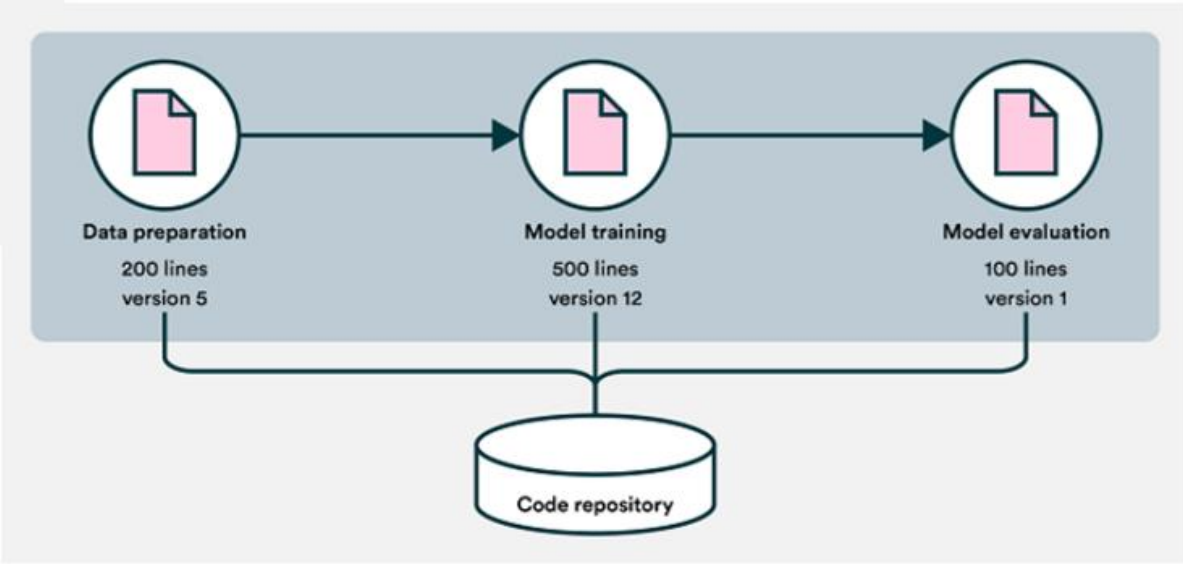
Siklus Pengembangan Solusi



- Yang membedakan deployment solusi Data Science Machine Learning, adalah perlunya memastikan deployment data set dengan baik
- Proses versioning, dan deployment data juga harus dilakukan secara tepat.
- Karena dataset ini akan menentukan bagaimana solusi bekerja dengan baik
- Sehingga ada 2 siklus yang berjalan yaitu untuk kode programnya dan untuk dataset

Perbedaan Software ML



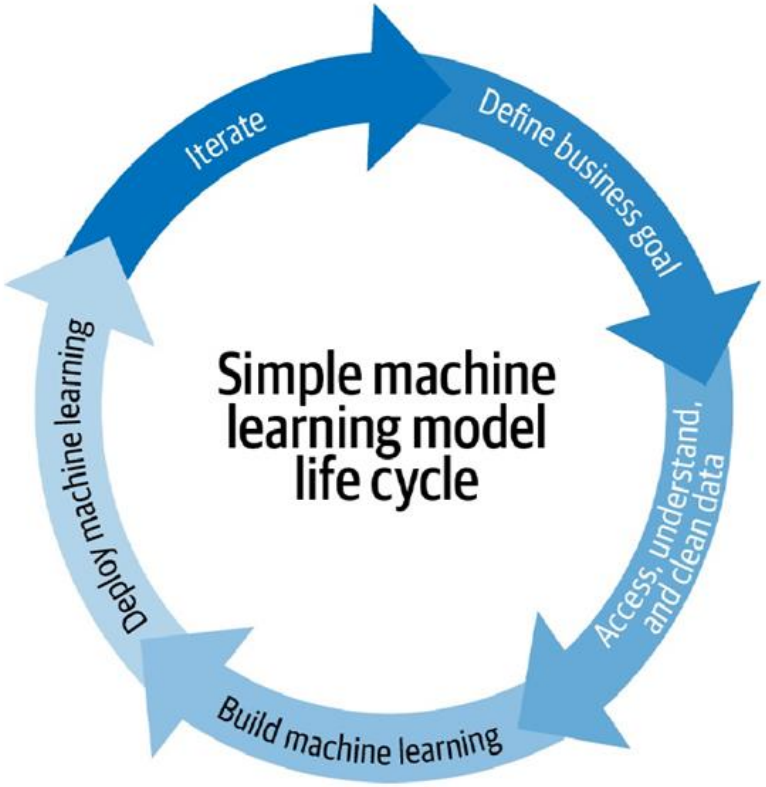


Pembuatan Artefak

- Bisa dilihat 2 siklus yang berbeda tersebut, satu terkait data hingga menjadi model (disimpan di model registry)
- Dan model tersebut digunakan oleh aplikasi yang di deploy.

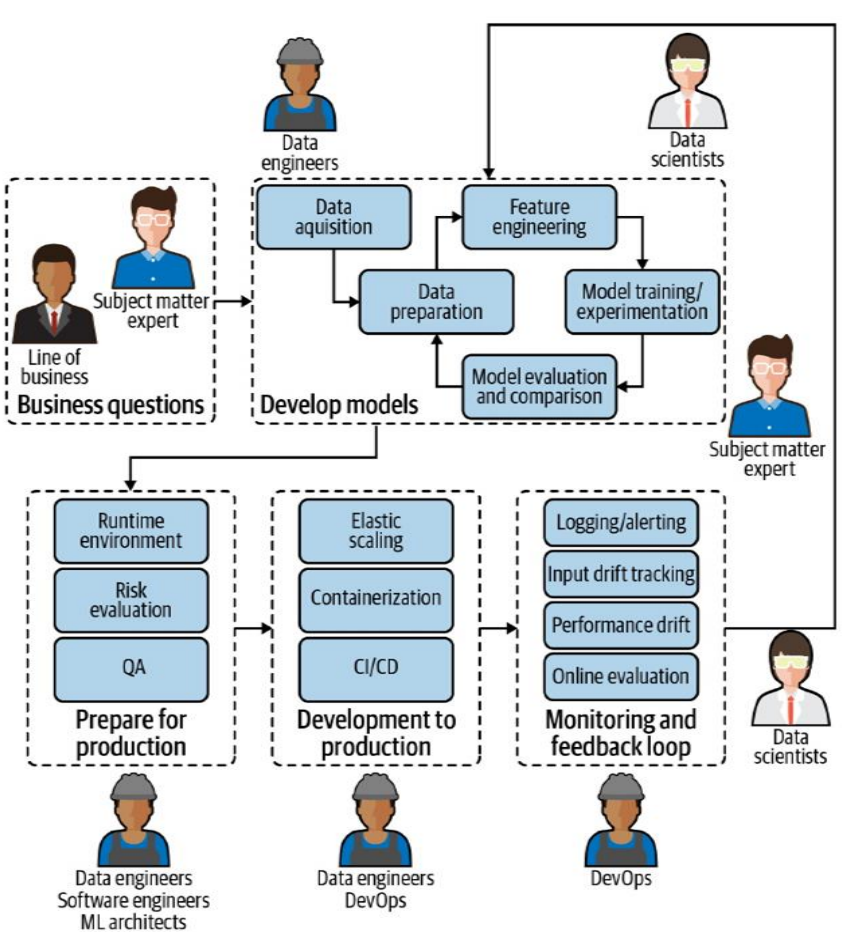
Data Scientists vs Soft. Engineer

- Data scientist sesungguhnya bukan software engineer. Mereka dikhususkan untuk membangun model dan mengevaluasinya, mereka bukan ahli di dalam membuat program aplikasi
- Walau begitu dalam pekerjaannya sering sekali hal ini menjadi tipis batasannya, pada saat ini seringkali data scientist diharapkan berperan dalam role yang beragam
- Sehingga diharapkan seorang Data Scientist memahami beberapa masalah teknis lainnya.
- Tentu saja hal ini makin kompleks ketika terjadi pergantian staf, seringkali seorang data scientist harus mengelola model yang dia tidak buat. Untuk itulah dokumentasi saat penyusunan model menjadi hal yang penting

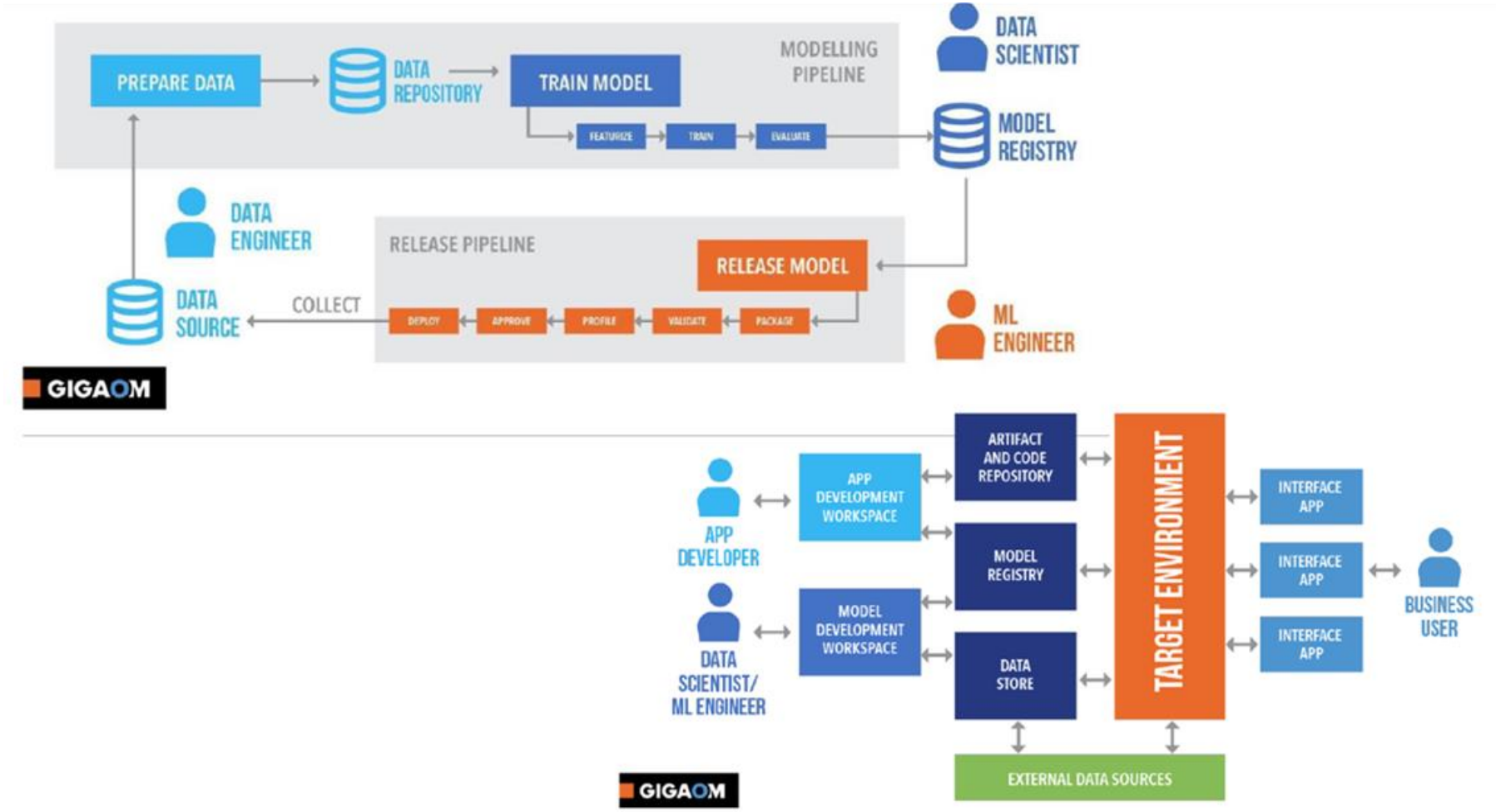


Personal

- Data scientist, Data Engineer, DevOps engineer bekerja sama untuk memberikan solusi
- Pada organisasi kecil seringkali dirangkap oleh orang yang sama
- Dengan mengetahui apa yang menjadi pertimbangan pihak lain maka, data scientist dapat mempersiapkan pekerjaannya dapat lebih tepat sehingga luarannya dapat dimanfaatkan oleh pihak lain, misal Data Engineer DevOps



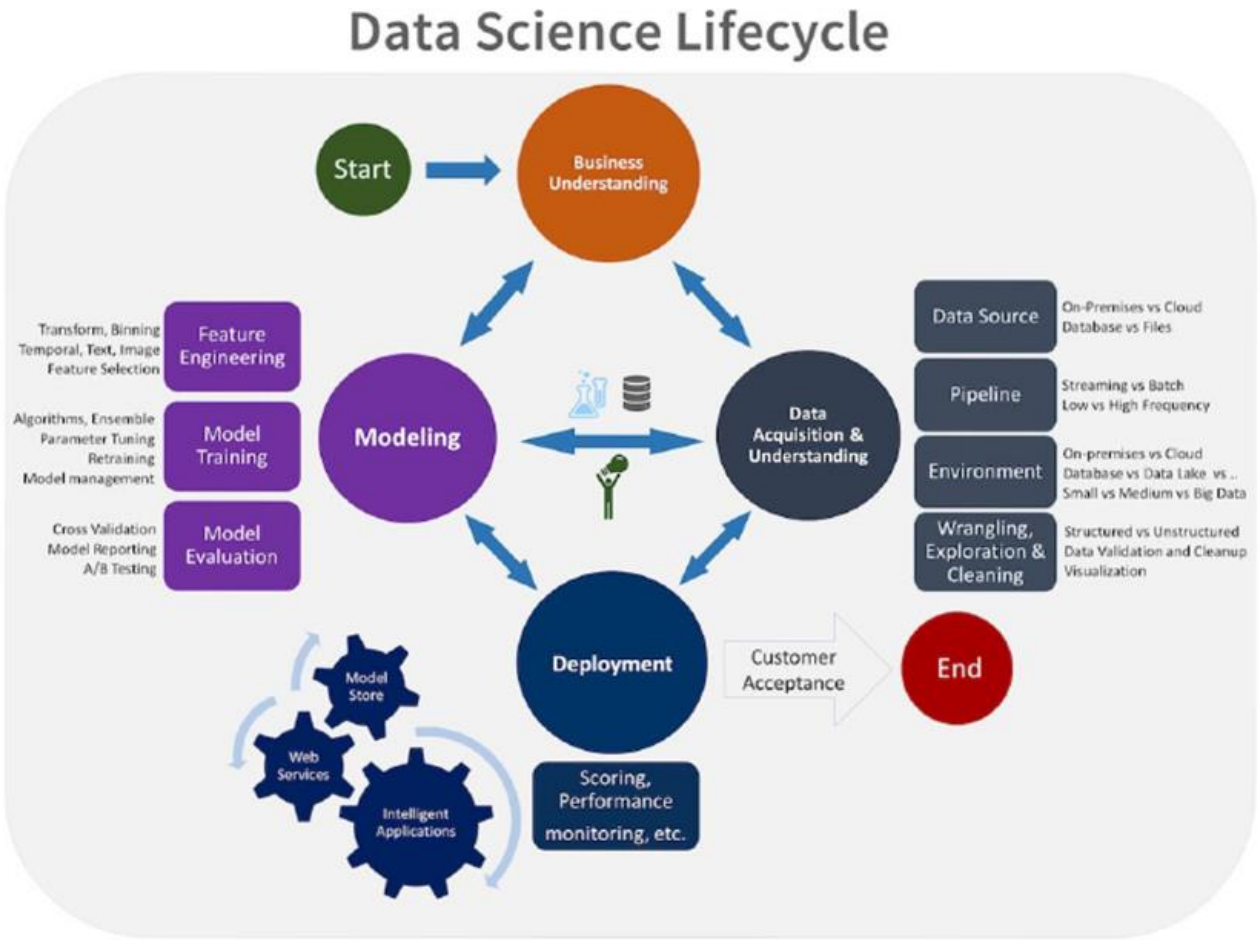
Data Scientist dan ML Engineer



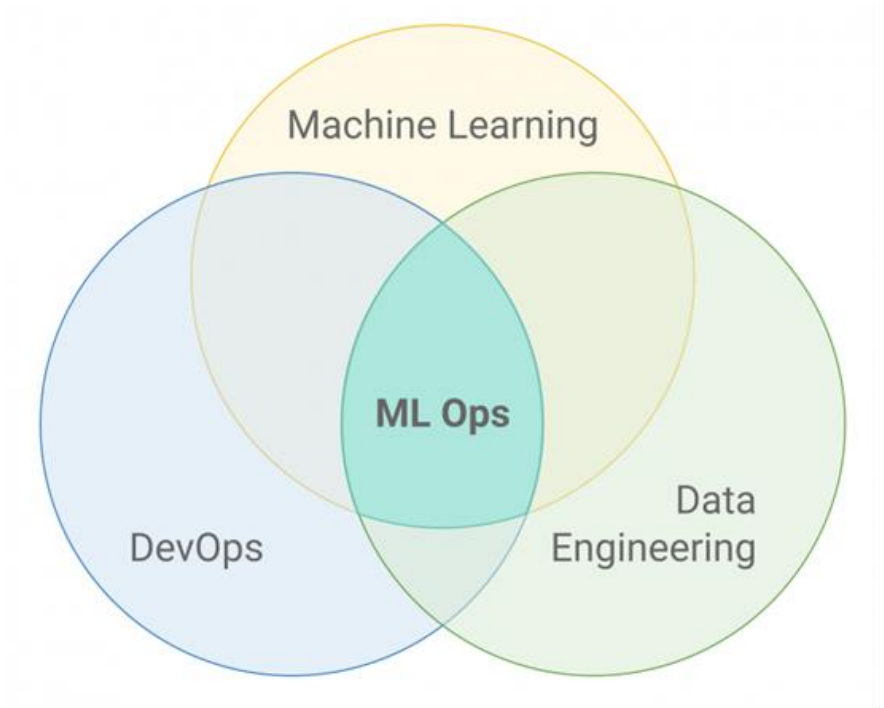
Deployments

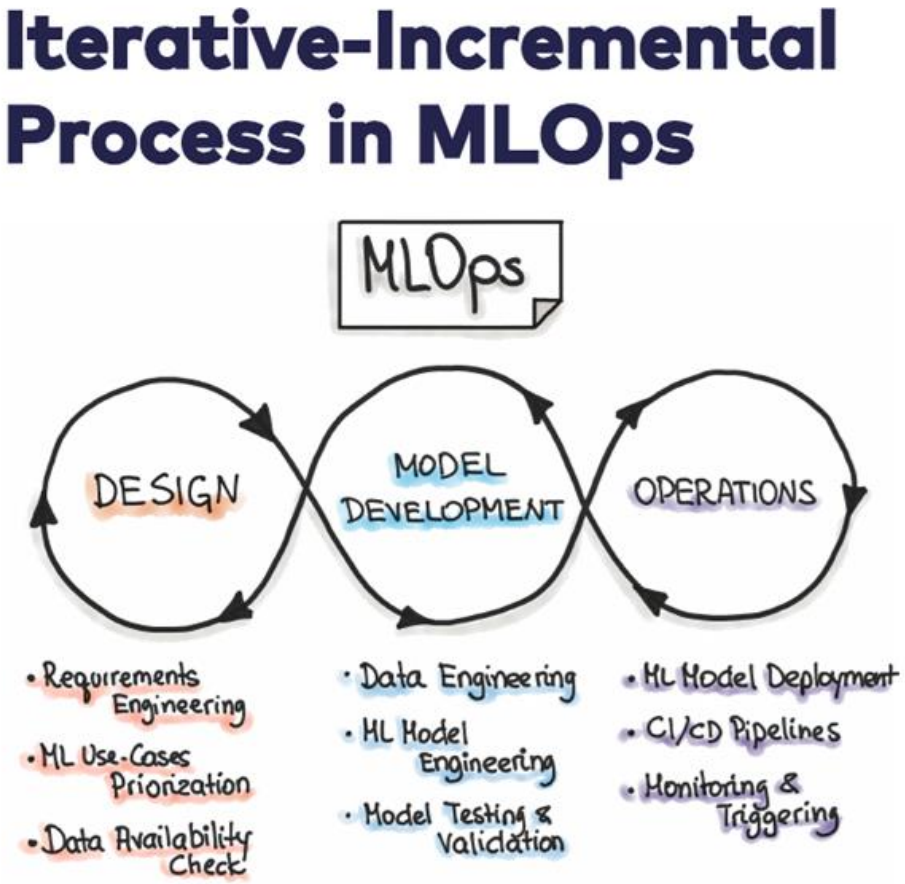
Business Drivers	Data Science	Model Versioning	Model Deployment	Data Pipelines
Connect with stakeholders	Keep stakeholders updated	Iteratively improve models	Add models to workflows	Build robust pipelines
Low hanging fruits High-impact projects	Build prototypes Log experiments	Feature engineering Hyperparameters	Serve through APIs Monitor performance	Operate on real-time data Archive data
Set performance indicators Set performance targets		Track and archive models Track and archive data	Automate deployment Automate rollback	
Use Design Thinking, 10 Types of Innovation, LEAN, Six Sigma, etc	Iterate, iterate, iterate	Make it reproducible		
	Python, R, Julia		*Spark, Docker, Kubernetes*	*Kafka, MQTT, Avro, Thrift*

Deployments



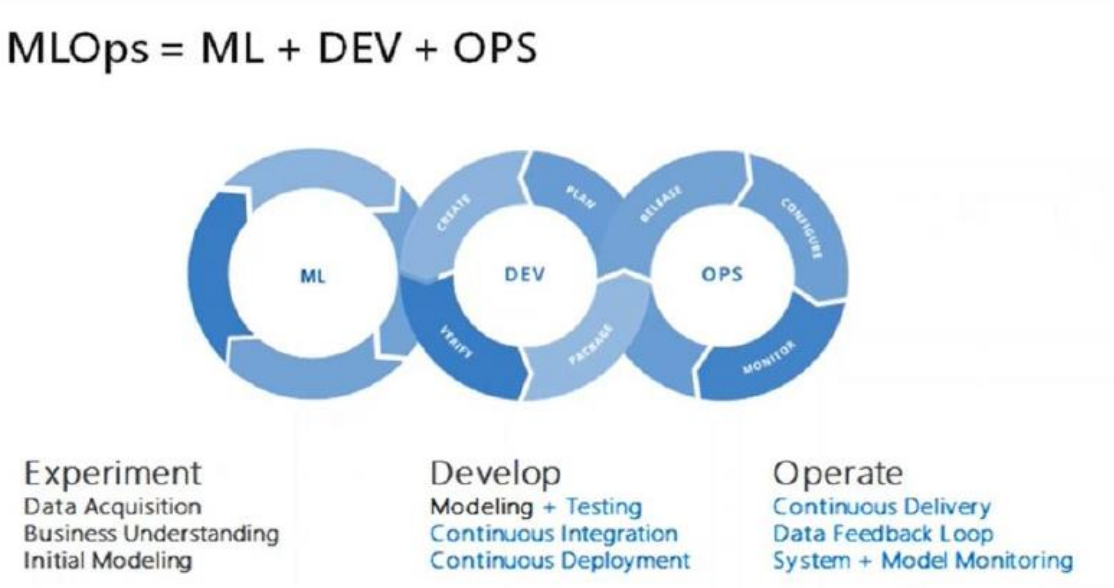
ML Ops


Machine Learning
ML Ops
DevOps
Data
Engineering


Iterative-Incremental Process in MLOps
MLOps
DESIGN
MODEL DEVELOPMENT
OPERATIONS
• Requirements Engineering
• ML Use-Cases Priorization
• Data Availability Check
• Data Engineering
• ML Model Engineering
• Model Testing & Validation
• ML Model Deployment
• CI/CD Pipelines
• Monitoring & Triggering

ML Ops

- MLOps DevOps untuk Machine Learning. MLOps ini mencari masalah yang terkait penerapan ML pada produksi
- Dengan MLOPs ini maka akan dicapai proses pengembangan machine learning end-to-end, dari proses desain, membangun, dan mengelola pengujian dan proses evolusi dari program yang berbasiskan Machine Learning
- MLOps ini semakin populer di Data Scientist, ML Engineer dan juga pengembang AI.
- Saat ini dibedakan antara model management rekayasa perangkat lunak konvensional dengan perangkat lunak berbasiskan machine learning yang menggunakan MLOps

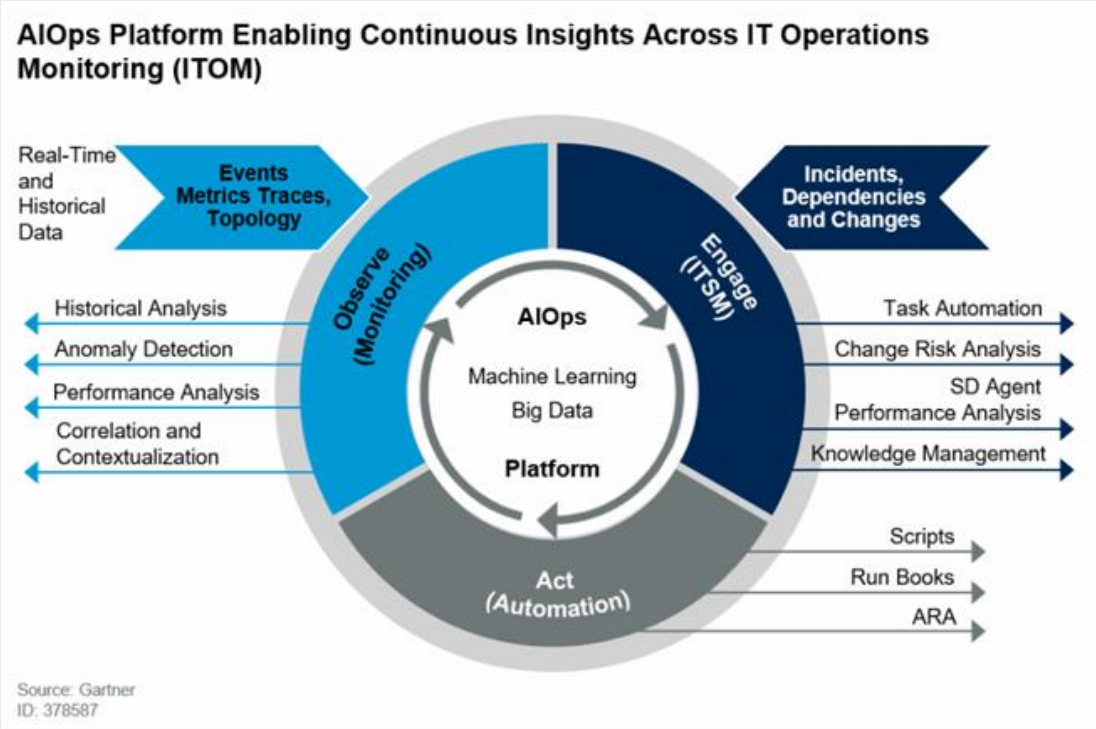


ML Ops

- MLOps bertujuan untuk menyatukan siklus dari machine learning dan perangkat lunak aplikasi
- MLOPs memungkinkan proses pengujian secara otomatis dari artefak machine learning (validasi data, testing model ML, dan testing integrasi model ML)
- MLOps memungkinkan penerapan agile pada proyek Machine Learning
- MLOps mendukung model machine learning dan dataset untuk dibuat dan dikelola sebagai bagian utama pada sistem CI/CD
- MPOPs mereduksi kerumitan teknis dari berbagai model machine learning
- MLOps dalam penerapannya harus bersifat tidak bergantung pada bahasa pemrograman, framework, platform dan infrastruktur

Functionality	Devops	MLOps
Stakeholders	Software developers(SW), Operations, QA	Data Scientist, Data Engineers, Software Developers, Operations, QA.
Automation(CI/CD),	Code by SW developers to Production system continuously.	2 Streams – Data Engineering Code, Model Code are Deployed, Synced up and models are validated and deployed
Governance, Monitoring	Dependency Changes, Failures, Key Metrices. Ethics and Good Model not needed	Devops+ Model Performance+ Data Drift(Quality of data, distribution). Ethics and Explanability of models
Model Retraining	NA	Critical and required.

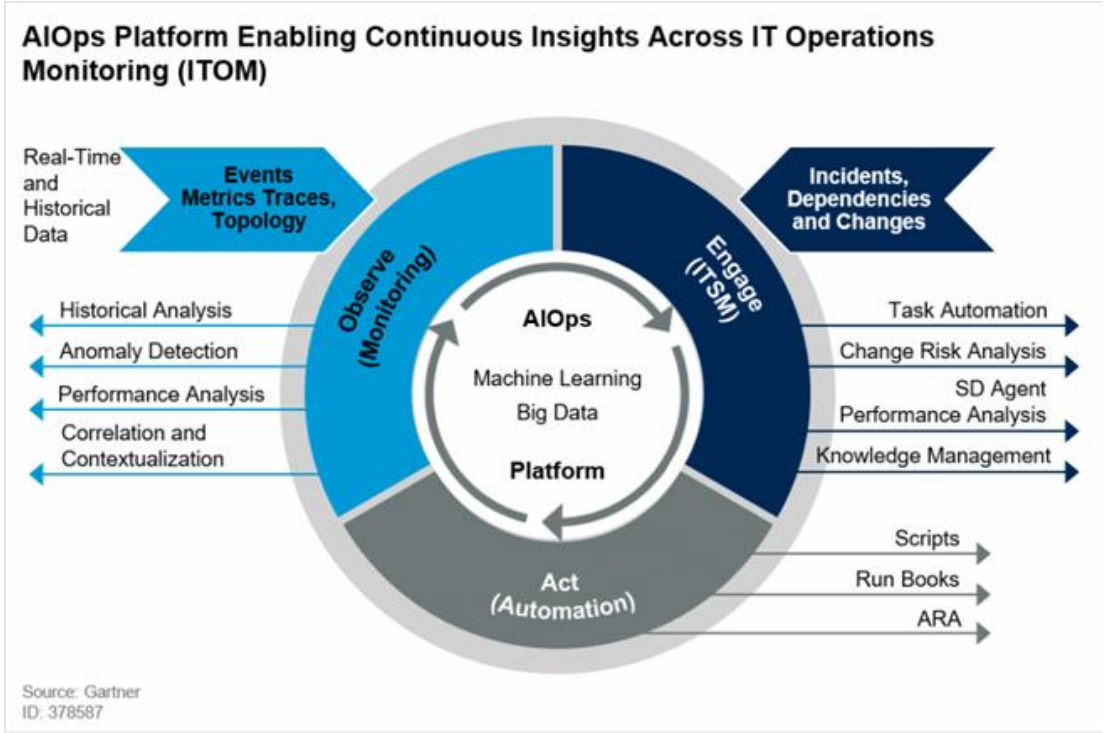
MLOps - AIOps



- MLOps (atau ModelOps) merupakan bidang baru, nama ini dikenal sekitar 2018-2019. Dua istilah MLOps dan ModelOps ini sering digunakan bergantian
- Tetapi ada yang mengatakan Model Ops lebih umum daripada MLOps karena bukan saja terkait model machine learning, tetapi juga model jenis lainnya (rule-based model)
- AIOps kadang juga dicampurkan dengan MLOps, tetapi hal ini sebetulnya berbeda, karena maksudnya adalah penggunaan AI untuk memecahkan masalah operasional, misal penggunaan AI untuk DevOps. Misal memprediksi perawatan untuk kerusakan jaringan, memberikan alert sebelum permasalahan muncul. Termasuk menganalisis root causal

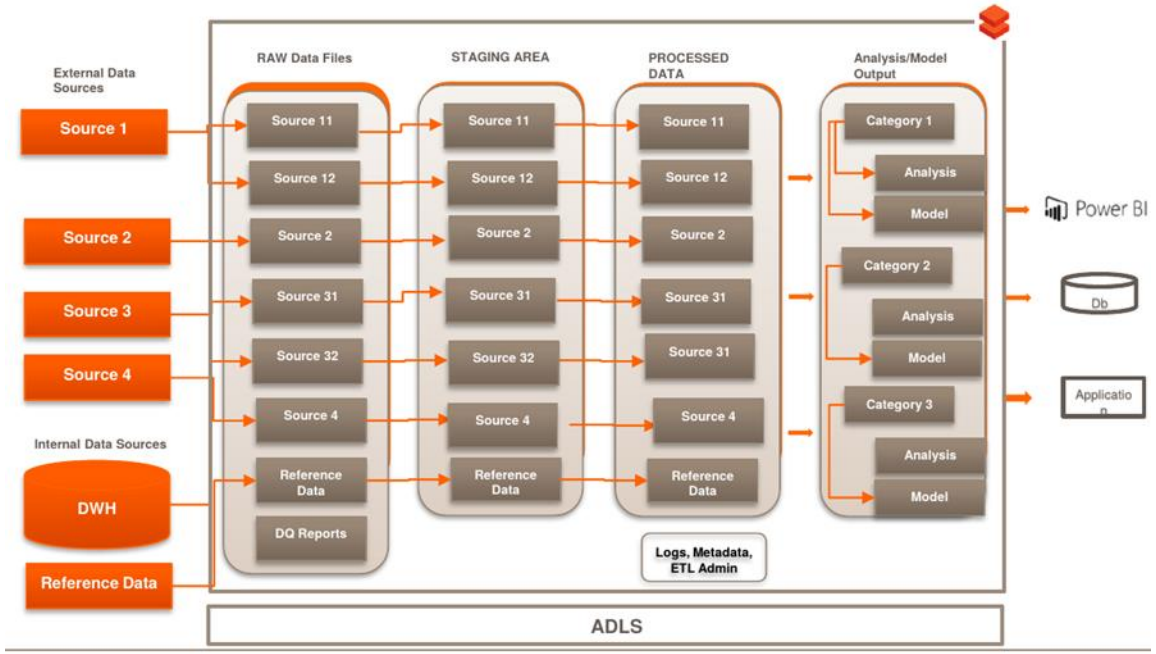
MLOps - AIOps

- AIOps menggunakan platform big data untuk mengolah data operasi performance dan event, system log dan pengukuran, data network (packet), incident dan ticketing, dokumen lainnya.
- AIOps menerapkan ML untuk melakukan
  - Memisahkan event alert yang penting dari noise (pattern matching dan rule dari data-data operasi)
  - Mengidentifikasi penyebab permasalahan dan mengusulkan solusi
  - Melakukan response otomatis, termasuk proaktif
  - Mempelajari secara terus menerus untuk kemampuan penanganan di masa depan.

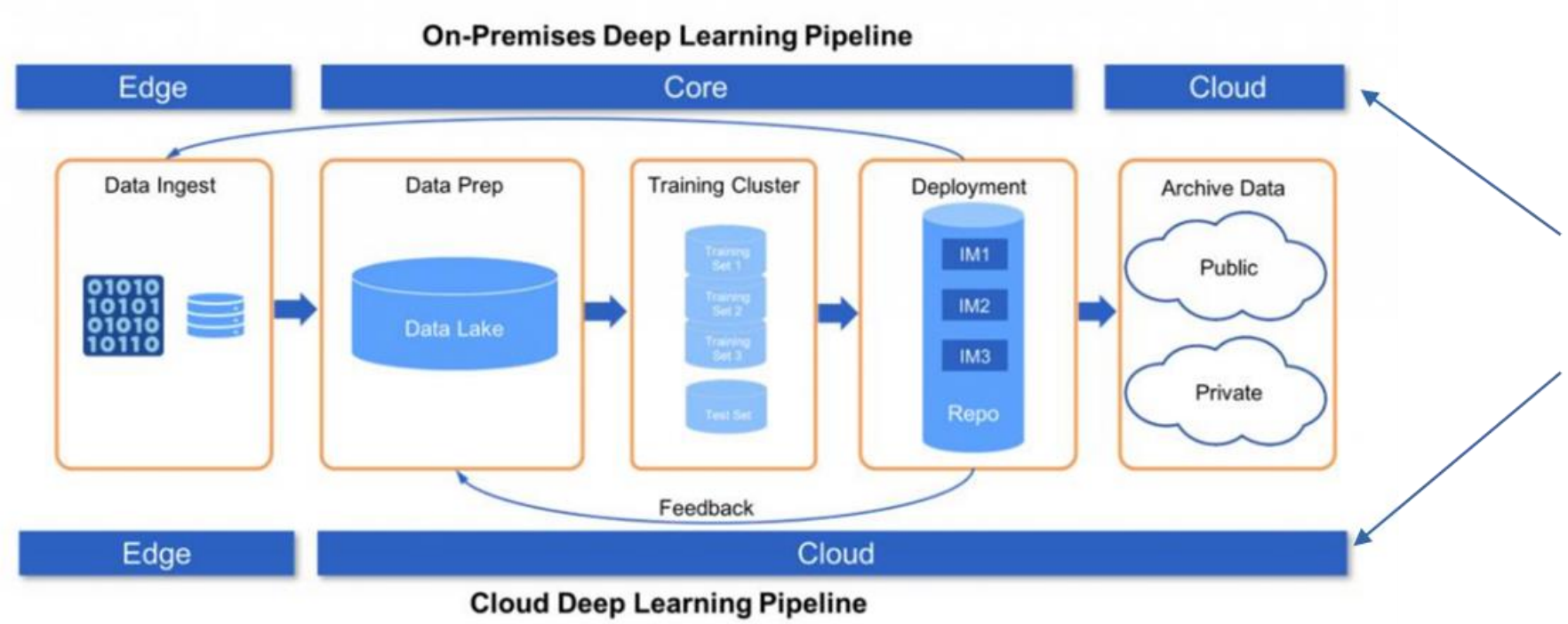


Pipelining Data

- Pada penerapan solusi Data Science besar, disamping aplikasi untuk melakukan algoritma data science dibutuhkan komponen lainnya
- Aliran data (pipelining) harus dikelola secara otomatis
- Dibutuhkan perangkat bantu seperti ETL, Metadata Management, Data Warehouse, Database dan lain sebagainya
- Sehingga proses pemanfaatan data tahap demi tahap dikelola oleh software pipelining tersendiri



Deep Learning Pipeline



Pemilihan dilakukan berdasarkan kriteria organisasi dan tujuan bisnis.

Beberapa aturan menyebabkan sulit melakukan di Public Cloud tetapi harus Private Cloud ataupun semuanya on-premise.

Penerapan Model Berbasis Pola

Suatu model dapat di deploy berdasarkan beberapa pola:

- Secara statistik/static deployment
- Secara dinamis pada perangkat pengguna (Dynamic deployment on user's devices)
- Secara dinamis pada server (Dynamic deployment on a server)

Penerapan Model Berbasis Pola: Static Deployment (1/1)

Static Deployment mirip dengan deployment perangkat lunak seperti biasa.

- File binary yang dapat diinstall dari seluruh perangkat lunak.
- Model dikemas sebagai resources yang tersedia saat runtime.
- Dll files (windows), *.so (linux), Java dan .Net.

Keuntungan static deployment:

- Eksekusi lebih cepat: perangkat lunak memiliki akses langsung ke model.
- Efisien waktu dan privacy: data pengguna tidak harus di upload di server.
- Model dapat dipanggil saat pengguna offline.
- Pengoperasian model menjadi tanggung jawab pengguna, bukan vendor perangkat lunak.

Penerapan Model Berbasis Pola: Dynamic Deployment (1/21)

Dynamic deployment on user's devices

- Dynamic deployment **pada perangkat user** mirip dengan static deployment, perbedaannya adalah model bukan merupakan bagian dari binary code aplikasi. Pembaruan model dapat dilakukan tanpa harus memperbaharui seluruh aplikasi yang berjalan pada perangkat pengguna.
- Dynamic deployment **pada perangkat user** dapat dilakukan dengan:
  - Deploying model parameters
  - Deploying a serialized object
  - Deploying ke web browser

Penerapan Model Berbasis Pola: Dynamic Deployment (2/21)

Deploying model parameters

- Pada scenario ini, file model hanya berisi parameter yang dipelajari.
- Perangkat pengguna telah menginstall runtime environment untuk model tersebut.
- Dapat menggunakan beberapa versi ringan dari machine learning package sehingga dapat berjalan pada mobile devices seperti
  - TensorFlow,
  - Apple’s Core ML
  - Scikit-learn, Keras, XGBoost

Penerapan Model Berbasis Pola: Dynamic Deployment (3/21)

Deploying model parameters

Contoh machine learning package untuk mobile devices:

Penerapan Model Berbasis Pola: Dynamic Deployment (4/21)

Deploying of serialized object

- Pada scenario ini, file model adalah object serial yang akan di-deserialize oleh aplikasi.
- Keuntungan: tidak memerlukan runtime environment pada perangkat pengguna.
- Kelemahan: update system akan cukup berat merupakan masalah jika perangkat lunak kita memiliki jutaan pelanggan.

Penerapan Model Berbasis Pola: Dynamic Deployment (5/21)

Deploying to web browser

- Pada skenario ini, akses ke browser banyak digunakan oleh aplikasi desktop maupun seluler.
- Deploying to web browser berarti model dapat dilatih dan berjalan pada browser. Contoh: framework TensorFlow.js
- Skenario lain: Model TensorFlow ditraining menggunakan python kemudian dideploy dan dijalankan di browser dengan runtime environment JavaScript.
- GPU (Graphic Processing Unit) dapat digunakan juga oleh TensorFlow.js

Penerapan Model Berbasis Pola: Dynamic Deployment (6/21)

Kelebihan dan kekurangan

Kelebihan dynamic deployment pada user devices:

- Proses model lebih cepat, komputasi sebagian di perangkat pengguna
- Jika diimplementasikan ke browser, pengorganisasian infrastruktur hanya perlu menyajikan halaman web yang menyertakan parameter model
- Model sangat mudah tersedia untuk analisis pihak ketiga

Kekurangan dynamic deployment pada user devices:

- Biaya bandwidth meningkat (jika berbasis browser)
- Updating model (jika serial)
- Performansi model susah dimonitor

Penerapan Model Berbasis Pola: Dynamic Deployment (7/21)

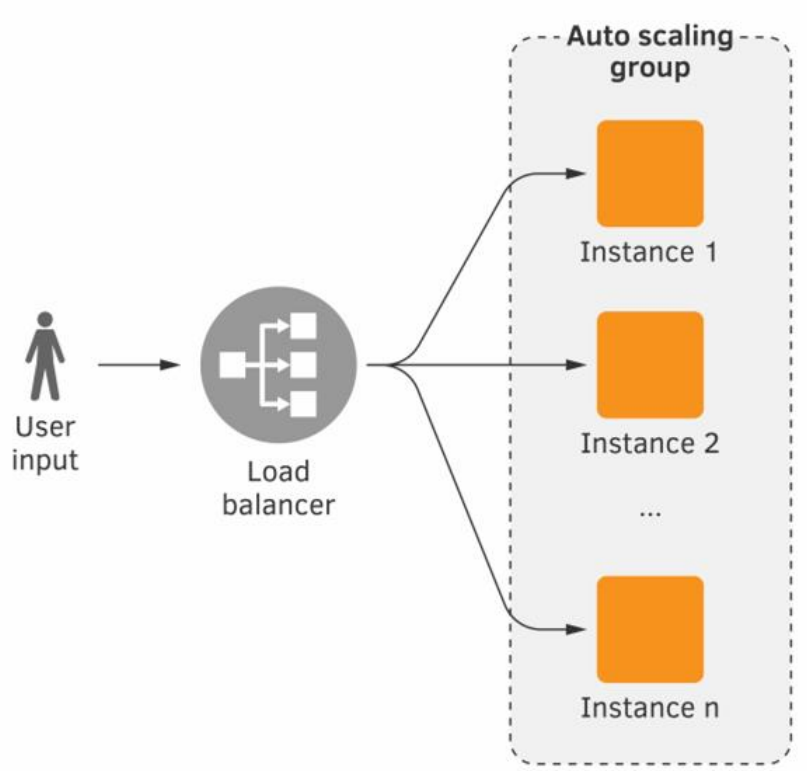
Dynamic deployment on a server

- Dynamic deployment **pada server** merupakan salah satu **solusi** untuk mengatasi kekurangan pada dynamic deployment **pada perangkat user**.
- Model ditempatkan pada server, tersedia untuk antarmuka REST API atau gRPC Google.
- Dynamic deployment **pada server** dapat dilakukan dengan:
  - Deployment pada virtual machine
  - Deployment dalam container
  - Serverless deployment
  - Model streaming

Penerapan Model Berbasis Pola: Dynamic Deployment (8/21)

Deployment pada virtual machine

- Pada domain web di cloud, prediksi disajikan sebagai respons HTTP.
- Ilustrasi:
  Pengguna: Layanan website (pada Virtual Machine/VM).
  Koneksi ke machine learning:
  Mengubah output ke dalam bentuk JavaScript Object Notation (JSON) atau XML.
  VM berjalan secara paralel untuk mengatasi beban komputasi yang tinggi.



Penerapan Model Berbasis Pola: Dynamic Deployment (9/21)

Deployment pada virtual machine

- Tensorflow:TensorFlow Serving (layanan gRPC bawaan).
- Keuntungan implementasi pada VM:
  - Arsitektur sistem perangkat lunak secara konseptual sederhana.
- Kekurangan implementasi pada VM:
  - Pemeliharaan server (fisik maupun virtual).
  - Latensi jaringan
  - Biaya relative tinggi dibanding dengan menggunakan container atau serverless.

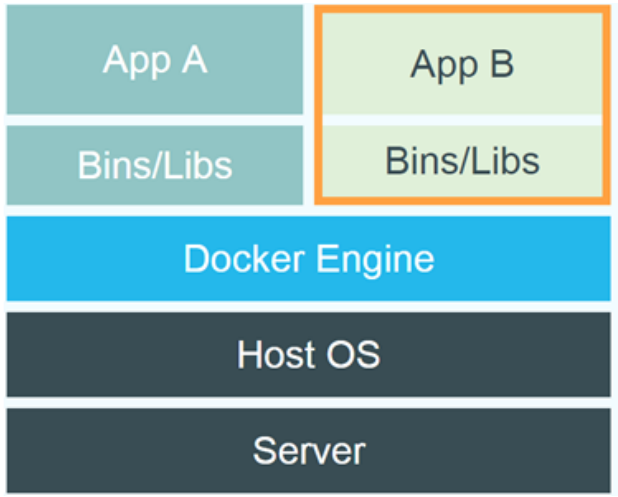
Penerapan Model Berbasis Pola: Dynamic Deployment (10/21)

Deployment pada container

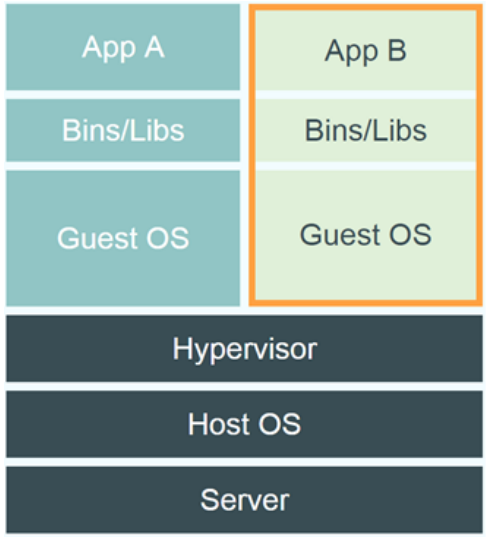
- Container mirip dengan VM
  - Memiliki runtime yang terisolasi dengan sistem file, CPU, Memory, dan ruang prosesnya sendiri.
- Container merupakan suatu alternatif modern dibanding dengan deployment pada VM.
  - Semua container berjalan pada VM atau fisik yang sama dan berbagi sistem operasi.
  - VM Menjalankan sistem operasi sendiri.
- Keuntungan deployment pada container:
  - Lebih hemat resources dan fleksibel dibanding menggunakan VM
- Kelemahan deployment pada container:
  - Deployment pada container sering dianggap kompleks dan membutuhkan expert/tenaga ahli

Docker vs. Virtual Machine

Container



- Standarisasi paket untuk software dan dependensi
- Mengisolasi aplikasi setiap os
- Menggunakan kernel OS
- Bekerja di Windows server dan Linux



- Resource Polling yang lebih bagus
- Mudah di skalabilitas
- VM ada di cloud

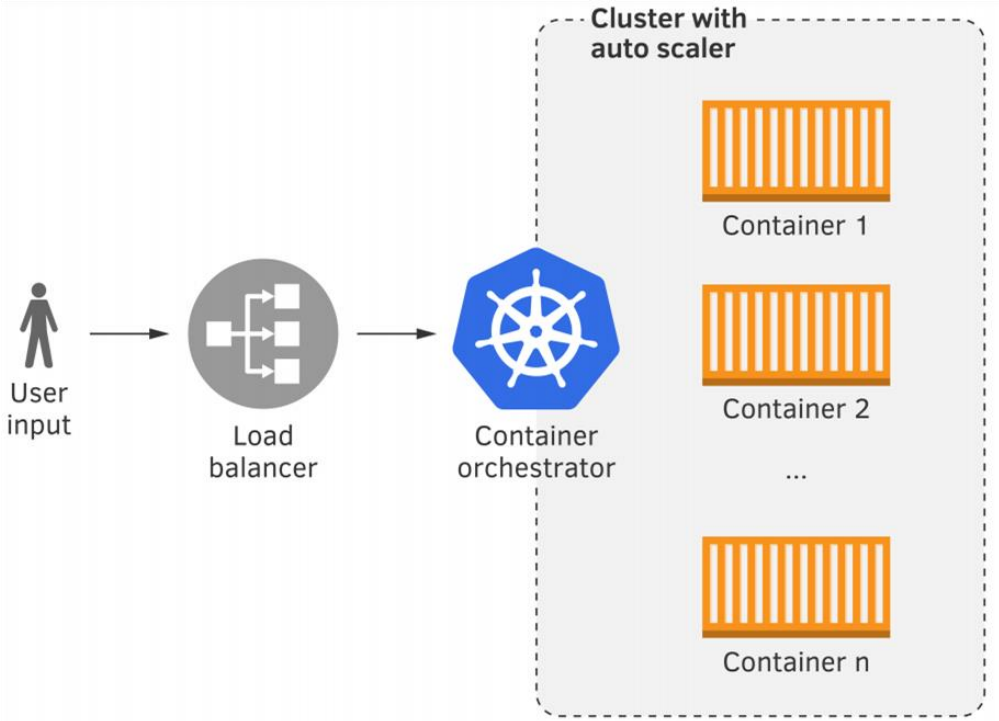
VM

- Setiap VM membutuhkan:
  - CPU
  - Storage
  - RAM
  - Keseluruhan Sistem Operasi
- Semakin banyak VM yang berjalan semakin banyak resource yang dibutuhkan

Penerapan Model Berbasis Pola: Dynamic Deployment (11/21)

Deployment pada container

- Sistem machine learning dan web services di install pada container (*Docker Container*).
- Container orchestrator digunakan untuk menjalankan container pada sekelompok server fisik/virtual.
- Contoh container orchestrator: Kubernetes, AWS fargate, Google Kubernetes Engineer.

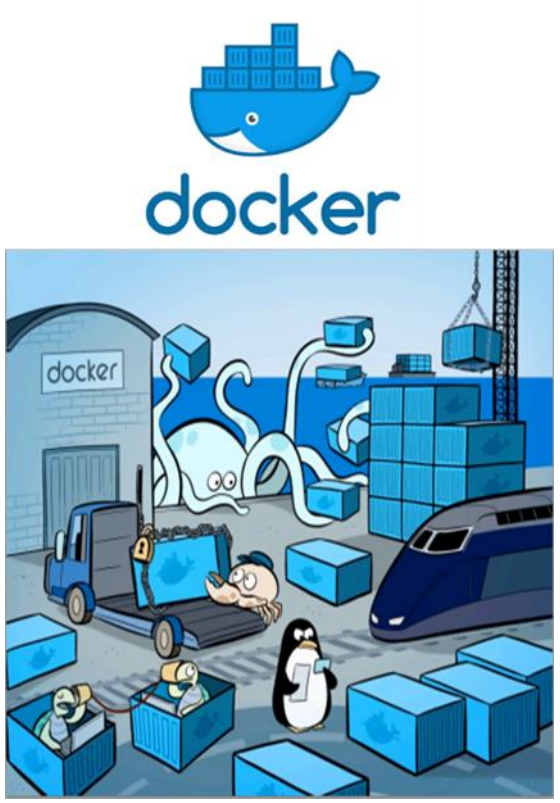


Deploying suatu model sebagai web service pada container yang berjalan pada cluster

Penerapan Model Berbasis Pola: Dynamic Deployment (12/21)

Deployment pada container

- Docker adalah aplikasi open source untuk menyatukan file-file yang dibutuhkan suatu perangkat lunak sehingga menjadi satu kesatuan yang lengkap dan berfungsi.
- Data pengaturan dan file pendukung disebut sebagai image.
- Kumpulan image digabung menjadi suatu wadah yang disebut Container.
- Docker merupakan platform perangkat lunak yang memungkinkan untuk build, test, dan deploy secara cepat dengan cara memaketkan pada wadah yang disebut Container tersebut.
- Dengan cara membuat paket standard tersebut memungkinkan **«Build, Ship and Run Any App, Anywhere»**



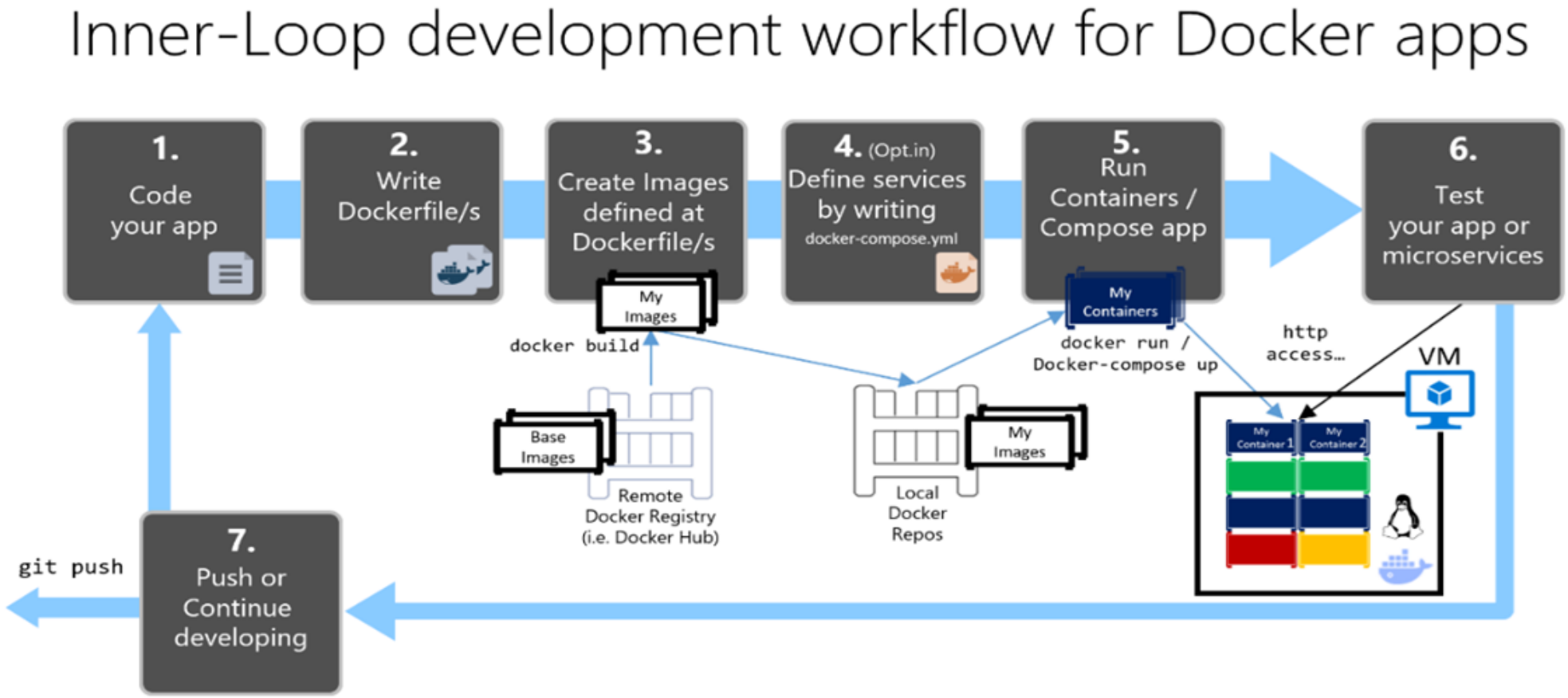
Penerapan Model Berbasis Pola: Dynamic Deployment (13/21)

Deployment pada container

Fitur Docker:

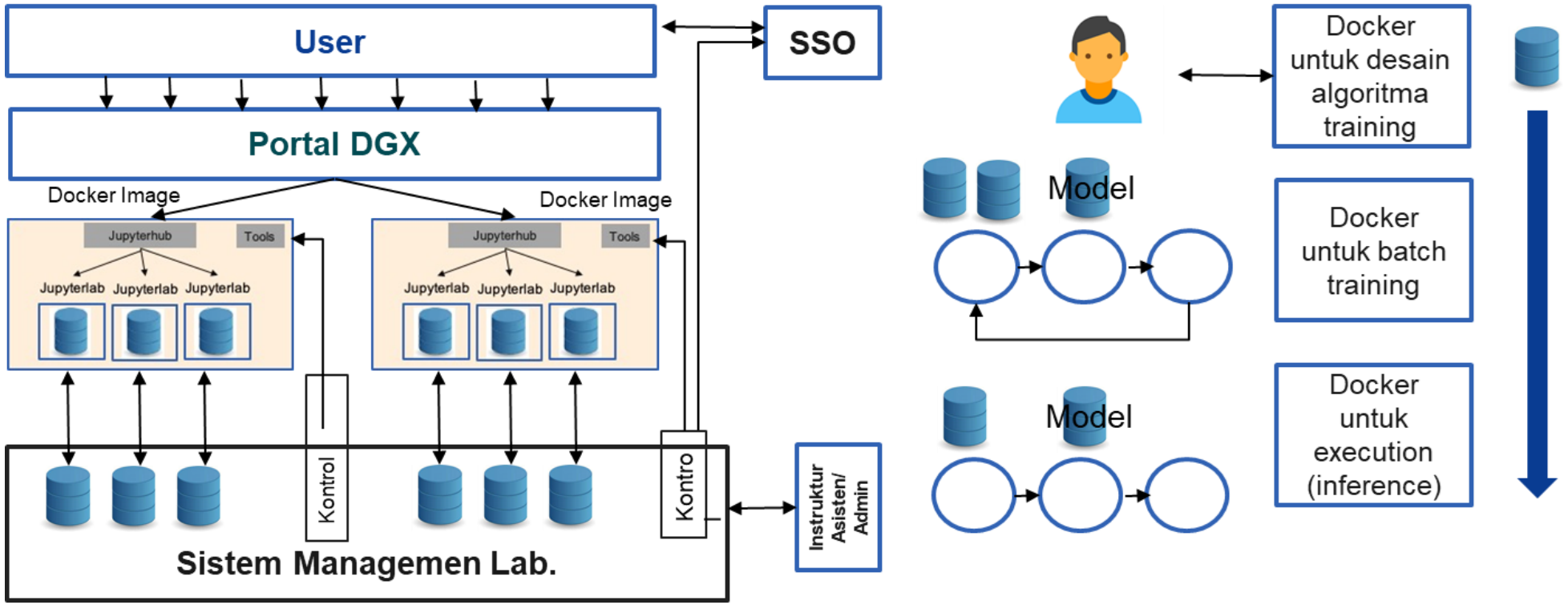
- Docker Engine, digunakan untuk membangun Docker Images dan membuat Container Docker.
- Docker Hub, registry yang digunakan untuk berbagai macam Docker Image.
- Docker Compose, digunakan untuk mendefinisikan aplikasi menggunakan banyak Container Docker.
- Docker Mac, Docker Linux, Docker Windows

Perubahan pola deployment



Penerapan Model Berbasis Pola: Dynamic Deployment (14/21)

Deployment pada container



Penerapan Model Berbasis Pola: Dynamic Deployment (15/21)

Serverless deployment

- Serverless deployment merupakan deployment yang memanfaatkan serverless computing dari cloud services providers
  - Amazon (lambda functions pada AWS),
  - Google (Google cloud platform) dan
  - Microsoft (functions pada Microsoft Azure).
- Serverless deployment terdiri dari:
  - Arsip ZIP yang berisi codes untuk menjalankan machine learning (model, feature extractor, dan scoring code).
  - File arsip ZIP berisi: nama tertentu yang berisi fungsi tertentu, atau class-method dengan spesifik signature.
  - File arsip ZIP diupload pada cloud platform dan dilakukan registrasi dengan nama yang unik.

Penerapan Model Berbasis Pola: Dynamic Deployment (16/21)

Serverless deployment

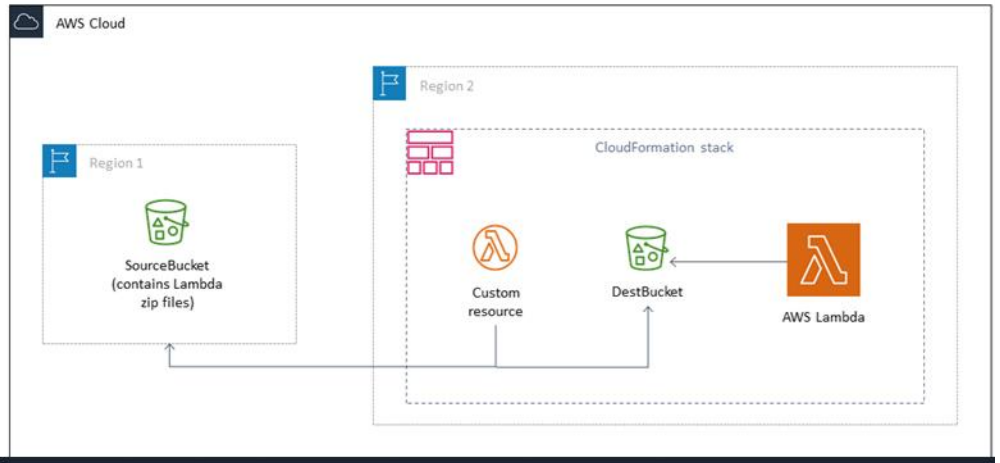
Cloud services:

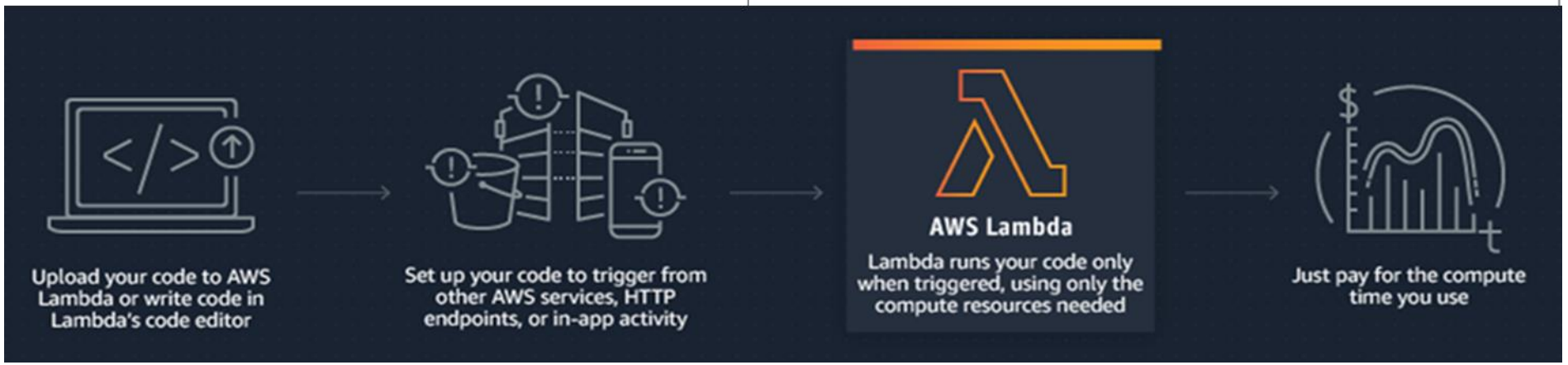
- Menyediakan API untuk mensubmit masukan ke Serverless Function.
- Menangani deploying codes dan model agar memadai pada sumber daya yang dimiliki.
- Mengeksekusi codes dan mengarahkan keluaran kembali ke client.
- Limitasi pada waktu eksekusi fungsi, ukuran file ZIP, dan jumlah RAM
  - Harus menyertakan Python Library agar model dapat dieksekusi dengan benar: Numpy, SCIpy, Scikit-Learn.
  - Bergantung pada platform cloud: Java, Go, PowerShell, Node.js, C#, Ruby

Penerapan Model Berbasis Pola: Dynamic Deployment (17/21)

Serverless deployment

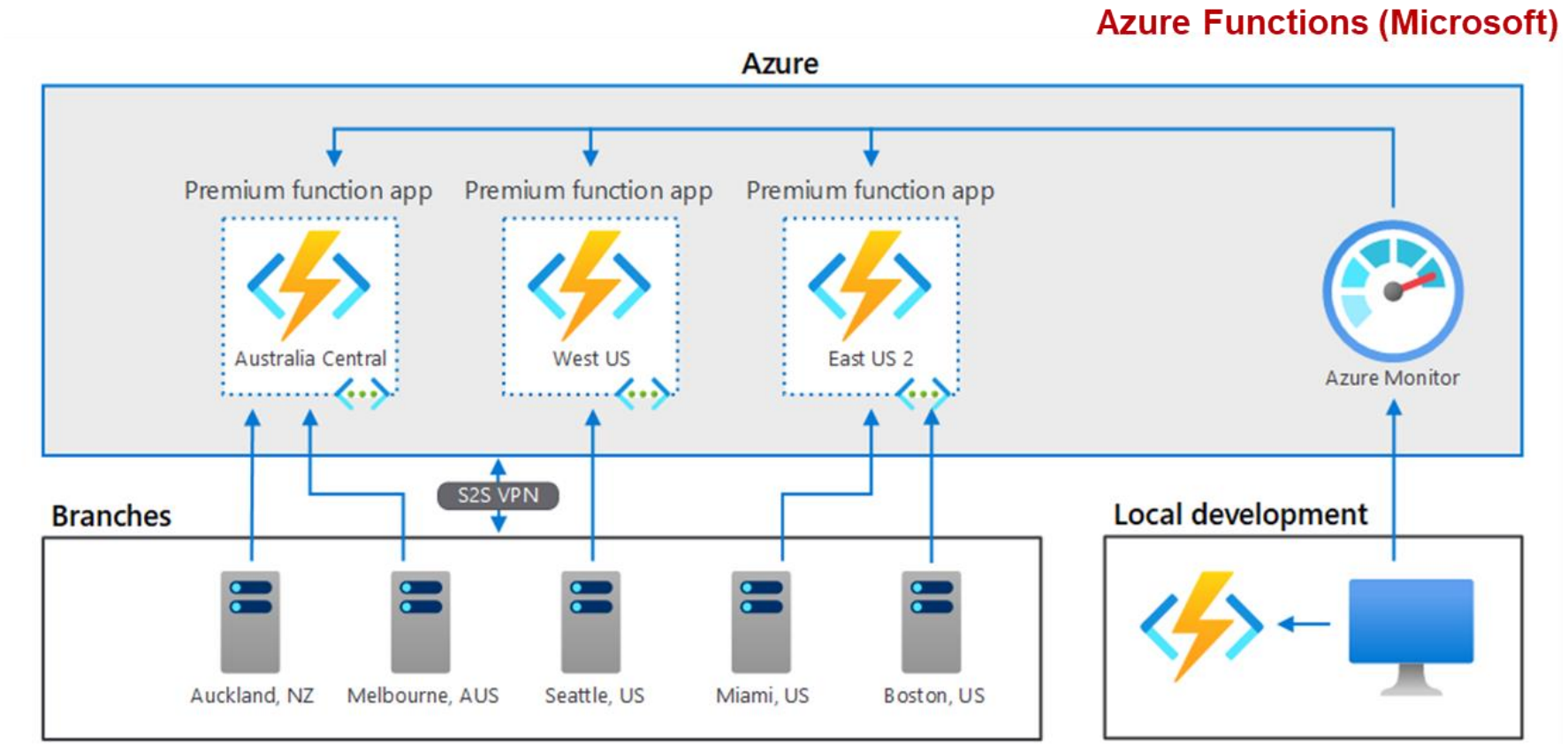
AWS Lambda (Amazon)





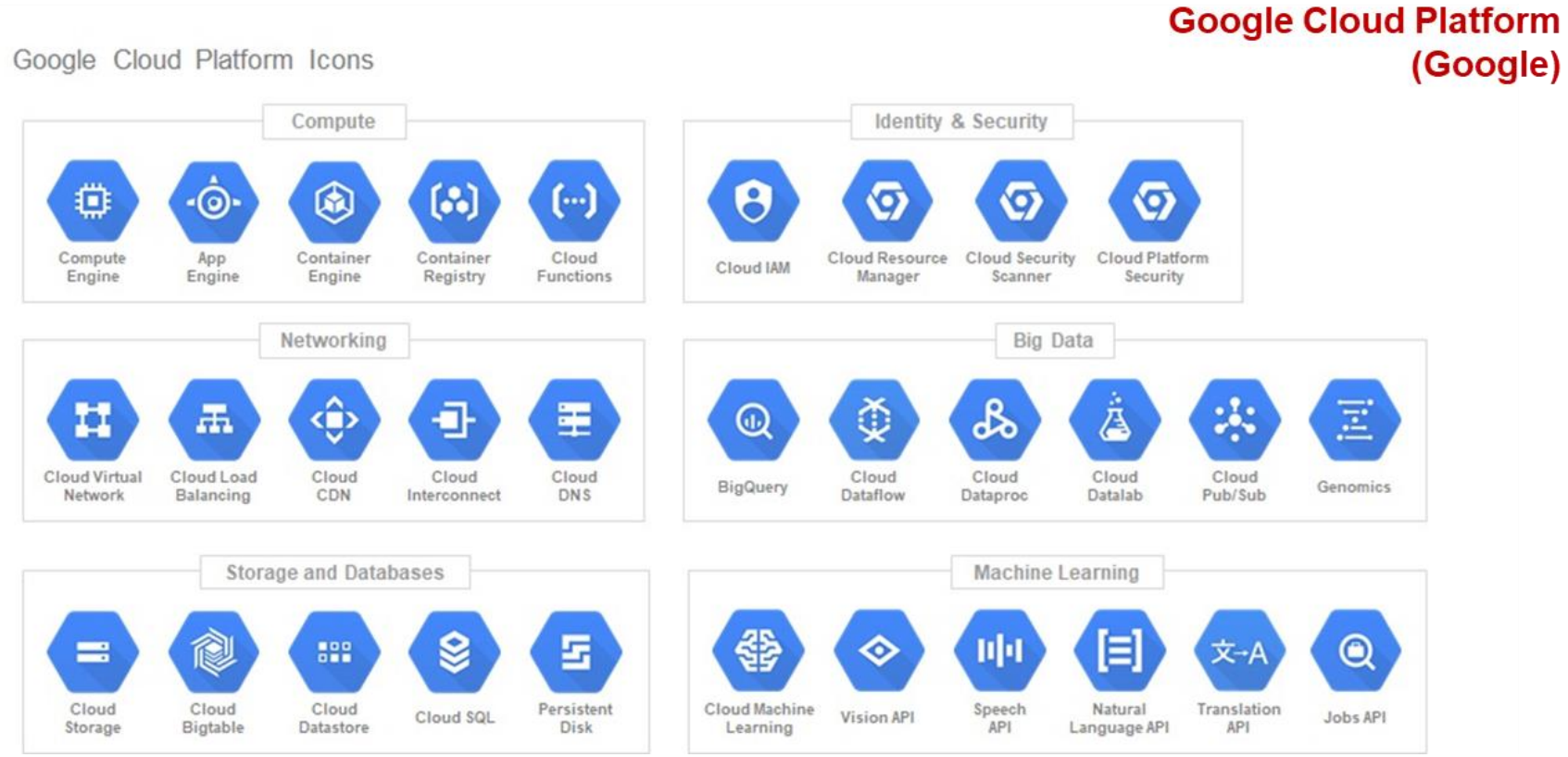
Penerapan Model Berbasis Pola: Dynamic Deployment (18/21)

Serverless deployment



Penerapan Model Berbasis Pola: Dynamic Deployment (19/21)

Serverless deployment



Penerapan Model Berbasis Pola: Dynamic Deployment (20/21)

Serverless deployment

Kelebihan serverless deployment:

- Tidak perlu menyediakan server maupun VM.
- Tidak perlu install dependensi, maintenance atau update system.
- Bersifat scalable.
- Hemat biaya: hanya membayar waktu komputasi.
- Mendukung operasi sinkron dan asinkron
- Rollback yang mudah untuk kembali ke versi sebelumnya.

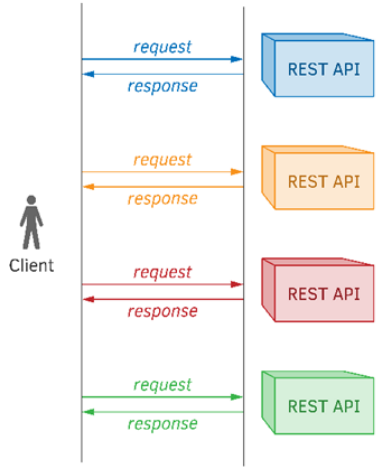
Kekurangan serverless deployment:

- Limitasi pada waktu eksekusi fungsi, ukuran file ZIP, dan jumlah RAM
- Tidak tersedianya akses GPU: terbatas jika ingin mendeploy Deep Model.

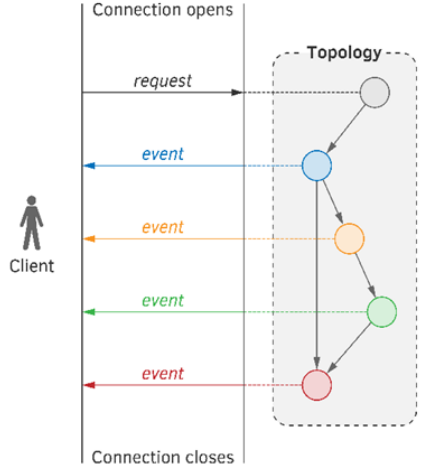
Penerapan Model Berbasis Pola: Dynamic Deployment (21/21)

Model streaming

- Merupakan kebalikan dari REST API.
- Semua model dan codes didaftarkan pada mesin pemrosesan aliran (SPE)
  - Apache Storm, Apache Spark, Apache Flink
  - Aplikasi berbasis stream-processing library (SPL): Apache Samza, Apache Kafka Streams, Akka Streams.



<REST API>



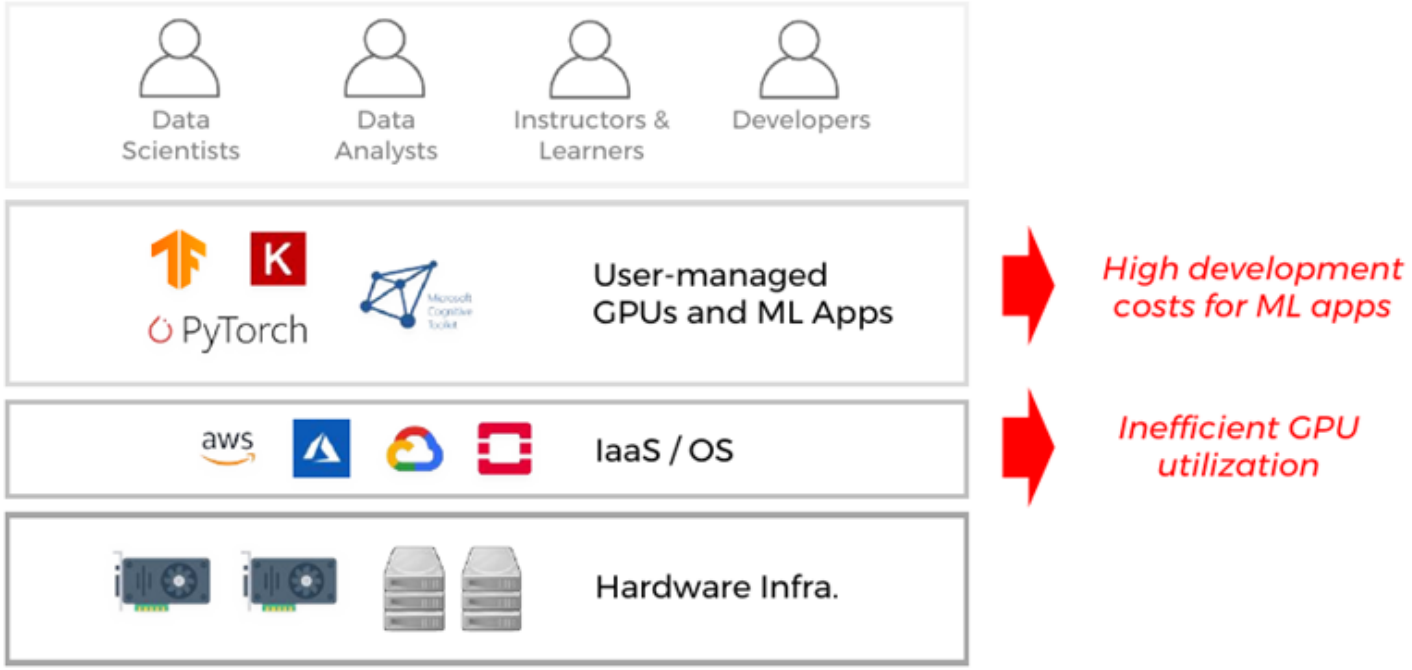
<Streaming>

REST API: Memproses elemen data, client menggunakan REST API mengirimkan permintaan satu-per-satu dan menerima respon secara sinkronus.

STREAMING: Memproses elemen data, client menggunakan STREAMING dengan membuka koneksi, mengirimkan permintaan, dan mendapatkan update event ketika hal tersebut terjadi

Perangkat Keras untuk AI

- OpenVINO
- Google Coral
- Huawei Ascend
- NVIDIA Jetson Nano
- NVIDIA Graphics Card
- NVIDIA DGX series



OpenVINO - INTEL

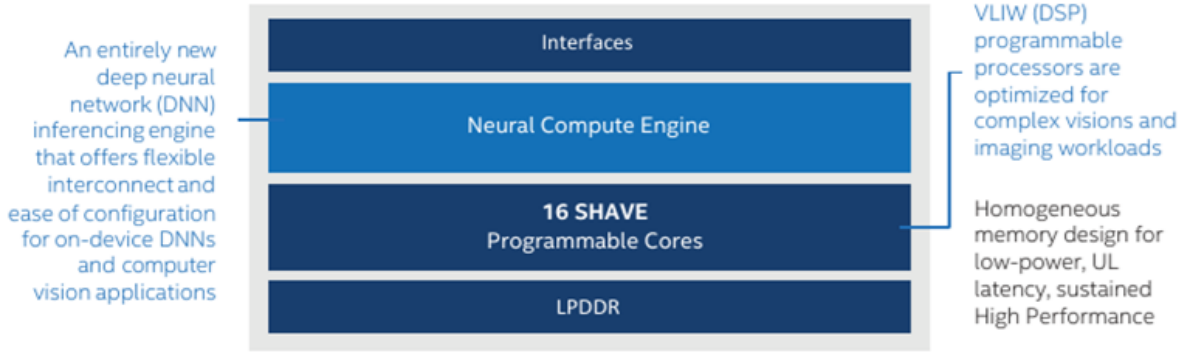
Bringing computer vision and AI to you
IoT and edge device prototypes are nov
easier than ever with enhance
capabilities of the Intel® Neura
Compute Stick 2 (Intel® NCS2).

It's built on the latest Intel®
MovidiusTM MyriadTM X VPU whicl
features the neural compute engine—
dedicated hardware accelerator for deej
neural network inferences. With mor
compute cores than the original version
and access to the Intel® Distribution of
OpenVINOTM toolkit, the Intel® NCS2
delivers 8X* performance boost over the
previous generation.1

The Intel Distribution of OpenVINOTM
toolkit is the default software
development kit1 to optimize
performance, integrate deep learning
inference, and run deep neural networks
(DNN) on Intel® MovidiusTM Vision
Processing Units (VPU).

Vision Processing Unit Architecture

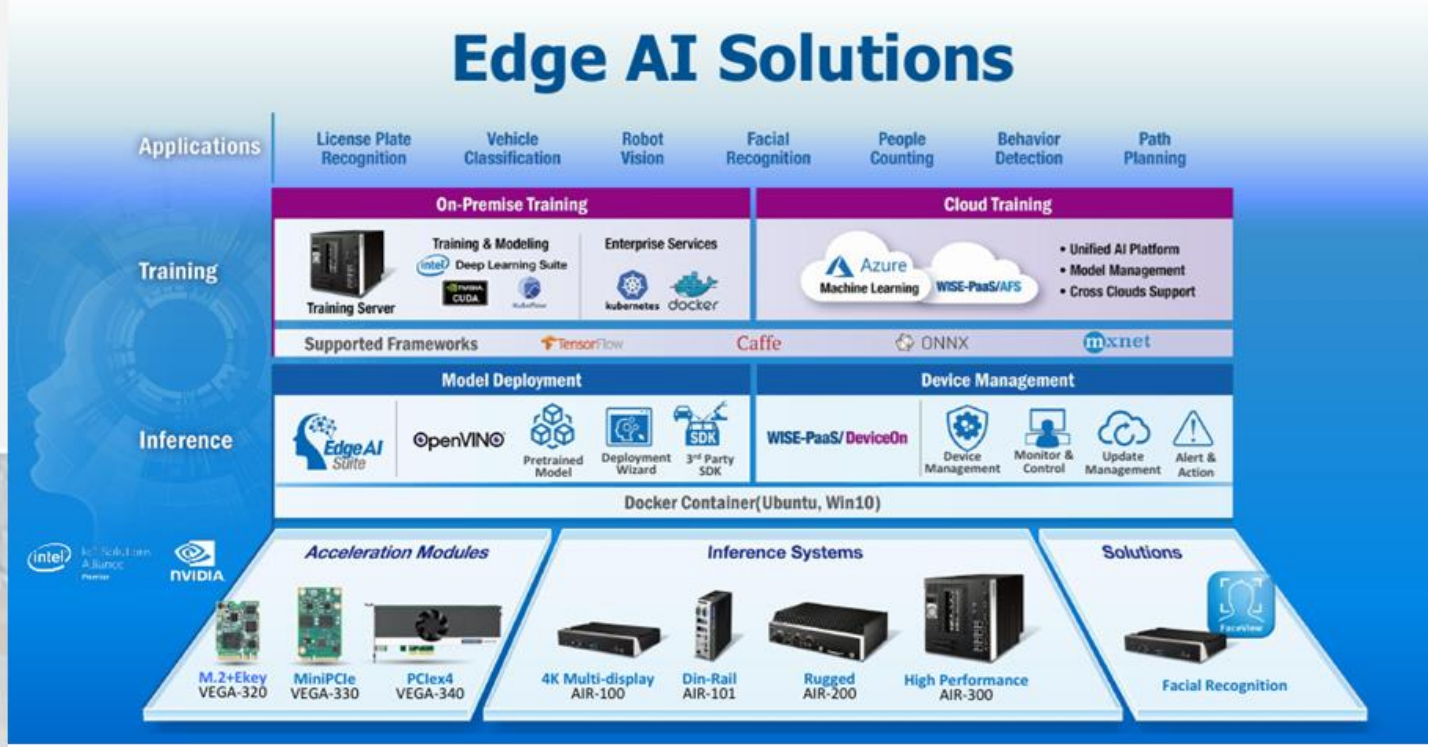
Intel® Movidius™ Myriad™ X VPU



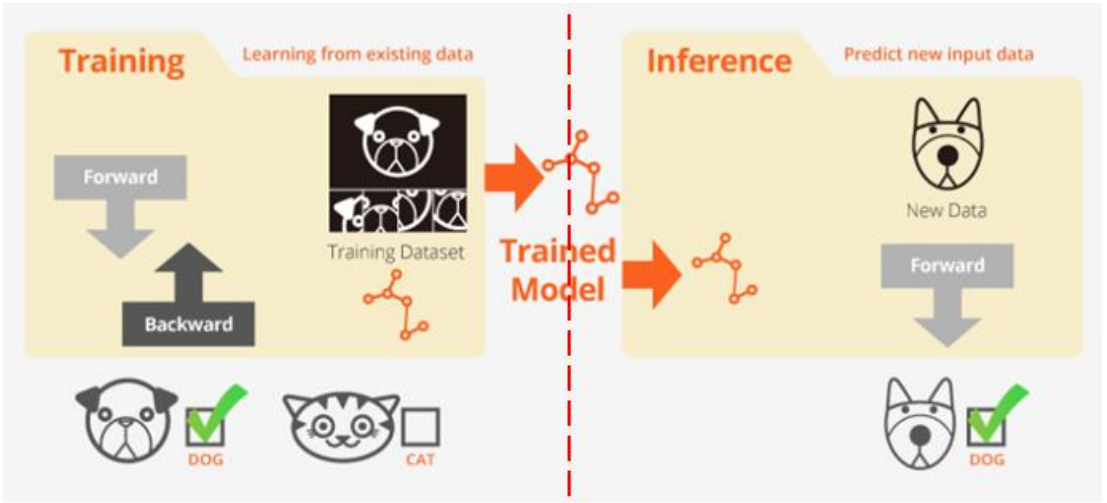
Technical Specifications

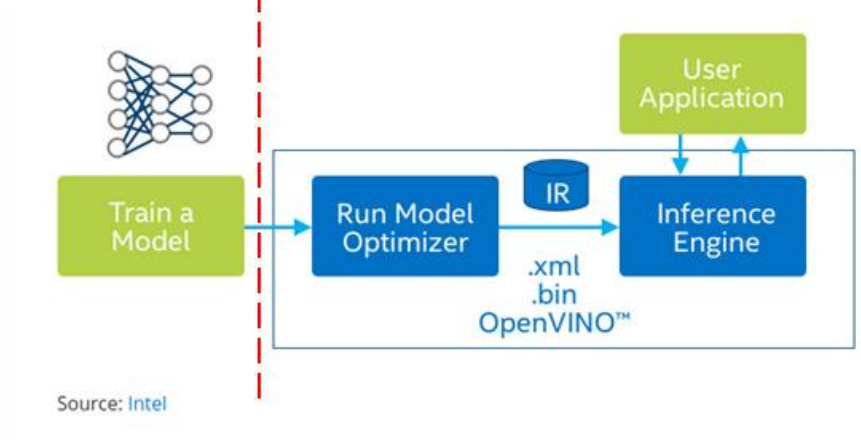
Specifications	Intel® Neural Compute Stick 2
Vision Processing Unit (VPU)	The Intel® Movidius™ Myriad™ X VPU
Software development kit	Intel® Distribution of OpenVINO™ toolkit
Operating Systems support	Ubuntu* 16.04.3 LTS (64 bit), Windows* 10 (64 bit), CentOS* 7.4 (64 bit), Raspbian*, and other via the open-source distribution of OpenVINO™
Supported framework	TensorFlow*, Caffe*, MXNet*, ONNX*, and PyTorch* / PaddlePaddle* via ONNX* conversion
Connectivity	USB 3.1 Type-A, USB 2.0 Type-A
Dimensions	72.5mm X 27mm X 14mm
Operating temperature	0° - 40° C
Material Master Number	964486
MSRP	$69 USD as of July 14, 2019
Supported platforms	x86_64, ARM

Pengembangan dengan OpenVINO

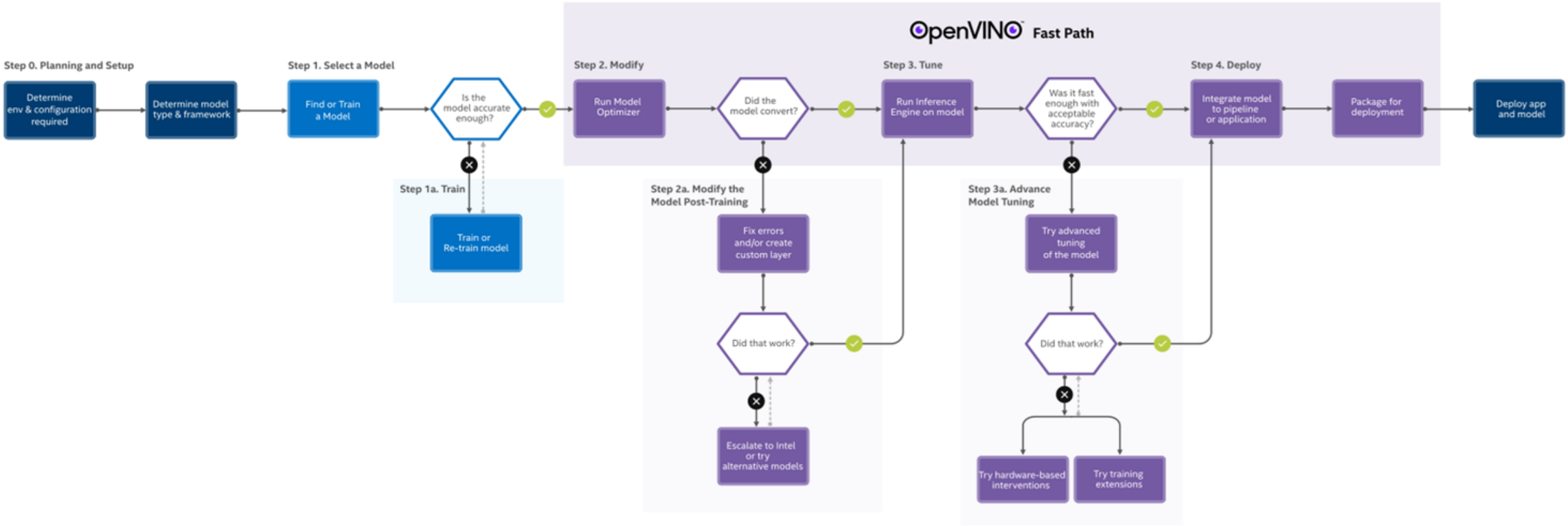


Proses Inference di OpenVINO





Siklus Deployment dengan OpenVINO

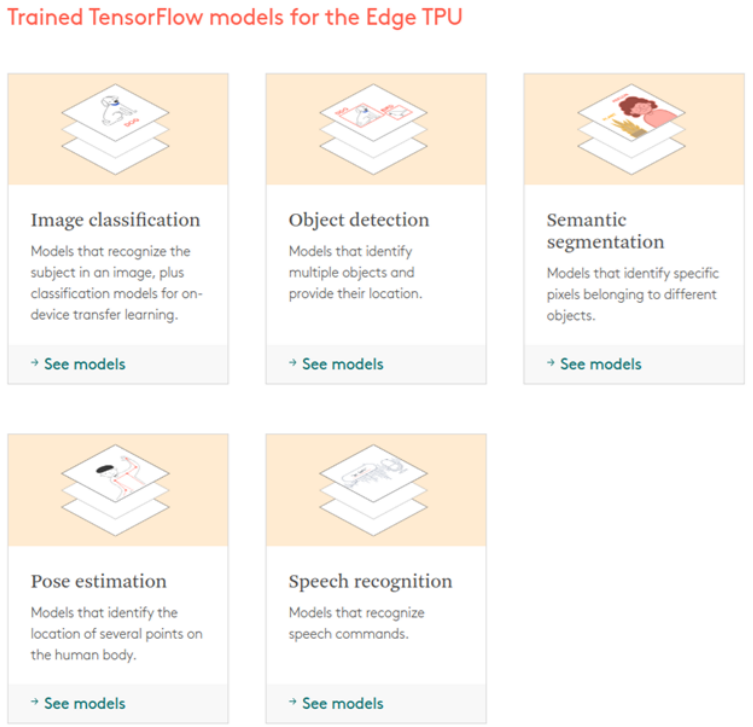


Google Coral [http://coral.ai]

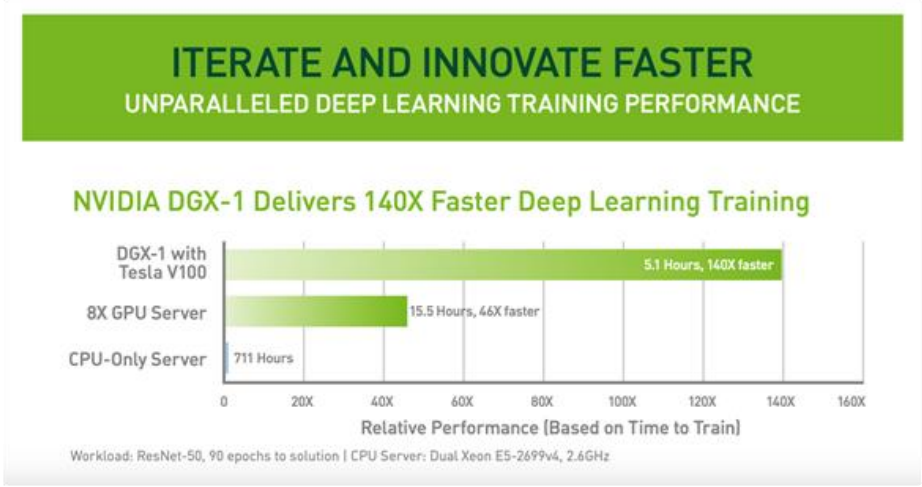
- Coral is a complete toolkit to build products with local AI. Coral is a hardware and software platform for building **intelligent devices with fast neural network inferencing**.
- At the heart of our devices is the Edge TPU coprocessor. This is a small ASIC built by Google that's specially-designed to execute state-of-the-art **neural networks at high speed, with a low power cost**.
- The Edge TPU is capable of performing 4 trillion operations (tera-operations) per second (TOPS), using 0.5 watts for each TOPS (2 TOPS per watt).
- **Efficient**. Balance power and performance with local, embedded applications.
- **Private.** Keep user data private by performing all inferences locally. You decide when data is stored or transferred.
- **Fast**. Run lightning-fast AI at industry-leading inference speeds for embedded devices.
- **Offline**. Deploy in the field where connectivity is limited.

Pre Trained Model

- **Model compatibility**. The Edge TPU supports a variety of model architectures built with TensorFlow, including models built with Keras. Our workflow to create models for Coral is based on the TensorFlow framework. No additional APIs are required to build or run your model. You only need a small runtime package, which delegates the execution of your model to the Edge TPU. To build a compatible model, you need to convert a trained model into the TensorFlow Lite format and quantize all parameter data (you can use either quantization-aware training or full integer post-training quantization). Then pass the model to our Edge TPU Compiler and it's ready to execute using the TensorFlow Lite API.
- **Simultaneous inferencing**. For applications that run multiple models, you can execute your models concurrently on a single Edge TPU by co-compiling the models so they share the Edge TPU scratchpad memory. Or, if you have multiple Edge TPUs in your system, you can increase performance by assigning each model to a specific Edge TPU and run them in parallel.
- **Model pipelining**. For applications that require very fast throughput or large models, pipelining your model allows you to execute different segments of the same model on different Edge TPUs. This can improve throughput for high-speed applications and can reduce total latency for large models that otherwise cannot fit into the cache of a single Edge TPU.
- **On-device training**. Although the Edge TPU is primarily intended for inferencing, you can also use it to accelerate transfer-learning with a pre-trained model. To simplify this process, we've created a Python API that executes the backbone of your model on the Edge TPU during training, and then calculates and saves new weight parameters for the final layer.



Penerapan Model Berbasis Pola: AI Infrastructure, Platform (1/5)



(NVIDIA)

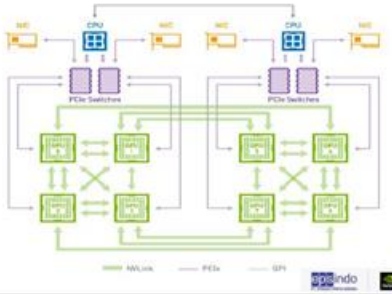
SYSTEM SPECIFICATIONS

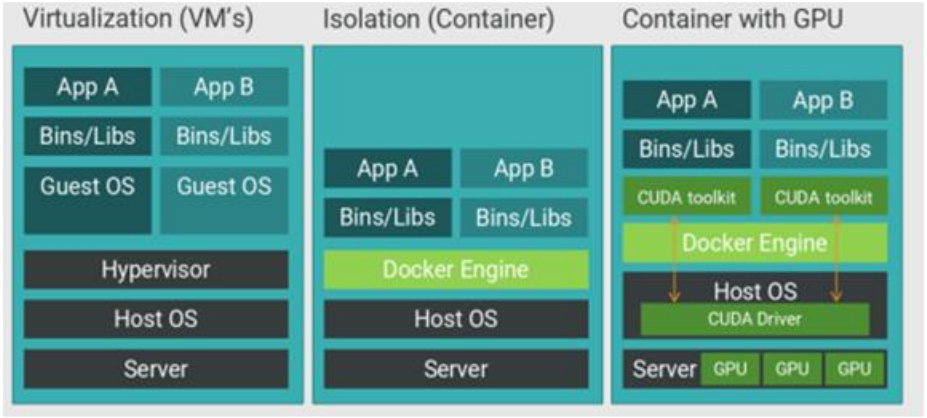
	NVIDIA DGX A100 640GB	NVIDIA DGX A100 320GB
GPUs	8x NVIDIA A100 80 GB GPUs	8x NVIDIA A100 40 GB GPUs
GPU Memory	640 GB total	320 GB total
Performance	5 petaFLOPS AI 10 petaOPS INT8	
NVIDIA NVSwitches	6	
System Power Usage	6.5 kW max	
CPU	Dual AMD Rome 7742, 128 cores total, 2.25 GHz (base), 3.4 GHz (max boost)	
System Memory	2 TB	1 TB
Networking	8x Single-Port Mellanox ConnectX-6 VPI 200Gb/s HDR InfiniBand 2x Dual-Port Mellanox ConnectX-6 VPI 10/25/50/100/200 Gb/s Ethernet	8x Single-Port Mellanox ConnectX-6 VPI 200Gb/s HDR InfiniBand 1x Dual-Port Mellanox ConnectX-6 VPI 10/25/50/100/200 Gb/s Ethernet
Storage	OS: 2x 1.92 TB M.2 NVME drives Internal Storage: 30 TB (8x 3.84 TB) U.2 NVMe drives	OS: 2x 1.92TB M.2 NVME drives Internal Storage: 15 TB (4x 3.84 TB) U.2 NVMe drives

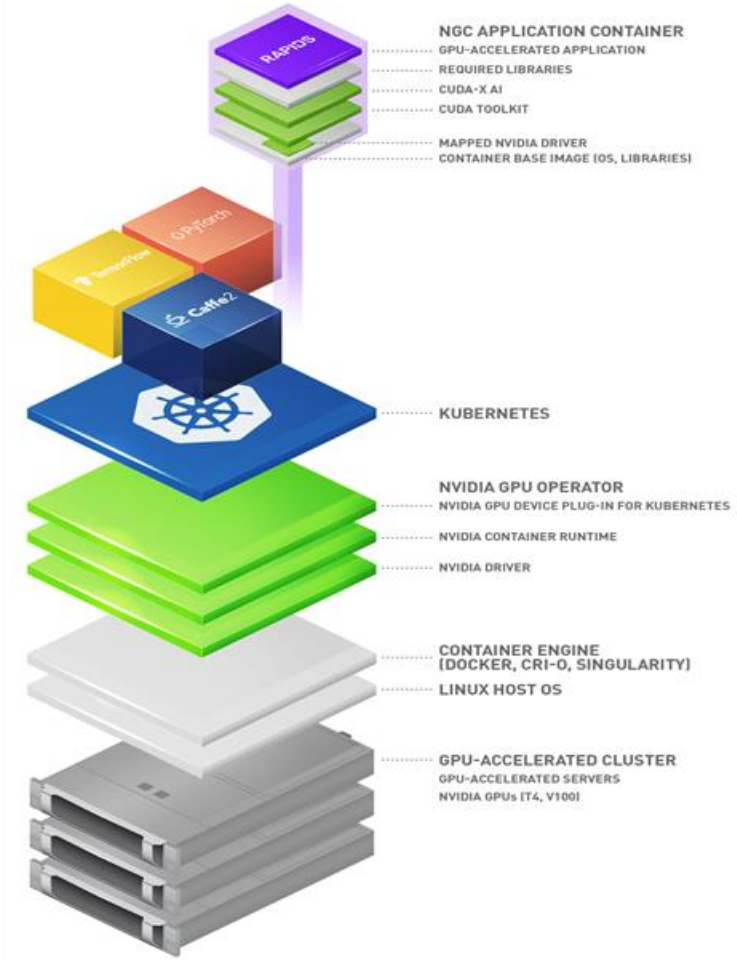
NVIDIA DGX-1

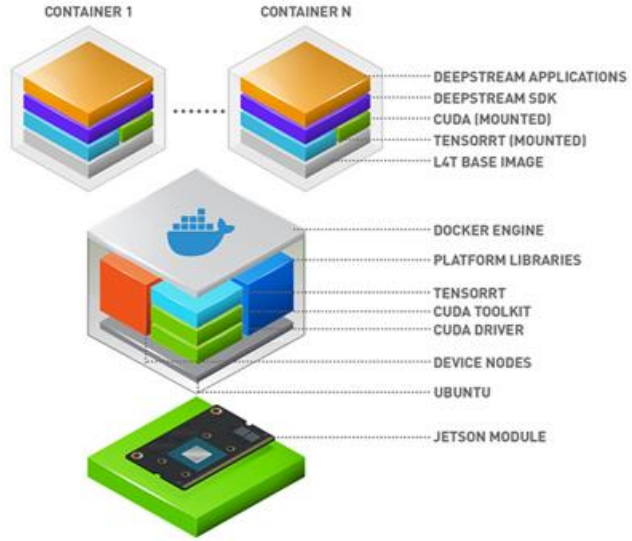
NVIDIA DGX-1
AI supercomputer-appliance-in-a-box

8x Tesla V100 connected via NVLINK
(125 TFLOPS FP32, 1 PFLOPS Tensor Core performance)
Dual Xeon CPU, 512 GB Memory
7 TB SSD Deep Learning Cache
Dual 10GbE, Quad IB 100Gb
3RU - 3200W
Optimized Deep Learning Software across the entire stack
- Containerized frameworks
- Always up-to-date via the cloud









NVIDIA GPU CLOUD REGISTRY

Common Software stack across NVIDIA GPUs

Deep Learning

All major frameworks with multi-GPU optimizations Uses NCCL for NVLINK data exchange Multi-threaded I/O to feed the GPUs

Caffe, Caffe2,CNTK, mxnet, PyTorch, Tensorflow, Theano, Torch

HPC

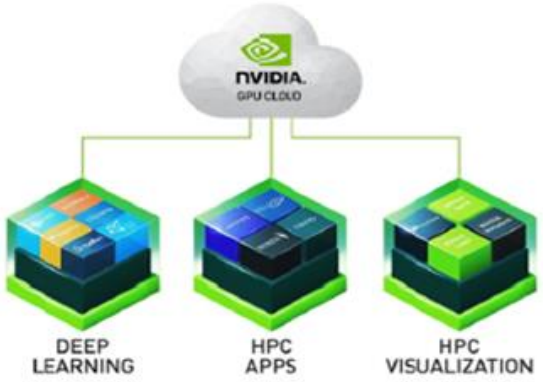
NAMD, Gromacs, LAMMPS, GAMESS, Relion, Chroma, MILC

HPC Visualization

Paraview with Optix, Index and Holodeck with OpenGL visualization base on NVIDIA Docker 2.0, IndeX, VMD

Single NGC Account

For use on GPUs everywhere - https://ngc.nvidia.com



NVIDIA GPU Cloud containerizes GPU-optimized frameworks, applications, runtimes, libraries, and operating system, available at no charge

NVIDIA Cloud Registry – Software Stack

Deep Learning

All major frameworks with multi-GPU optimizations Uses NCCL for NVLINK data exchange Multi-threaded I/O to feed the GPUs

Caffe, Caffe2,CNTK, mxnet, PyTorch, Tensorflow, Theano, Torch

HPC

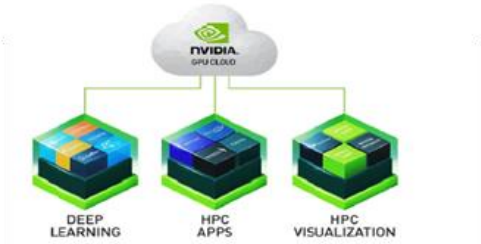
NAMD, Gromacs, LAMMPS, GAMESS, Relion, Chroma, MILC

HPC Visualization

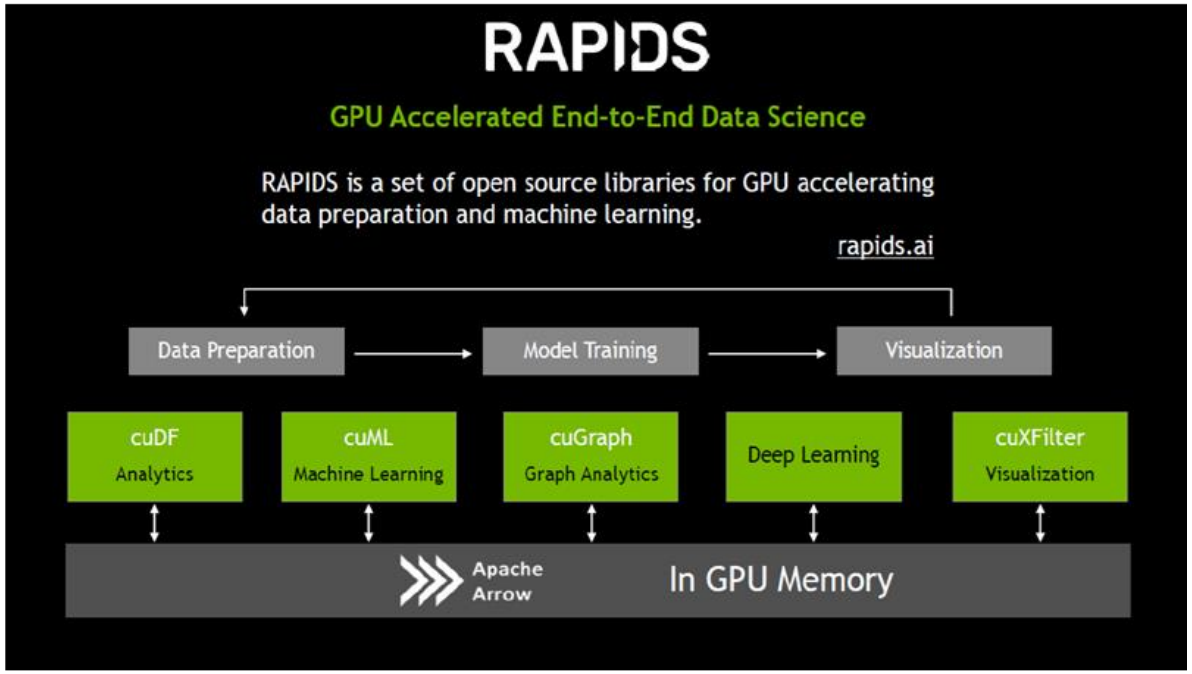
Paraview with Optix, Index and Holodeck with OpenGL visualization base on NVIDIA Docker 2.0, IndeX, VMD

Single NGC Account

For use on GPUs everywhere - https://ngc.nvidia.com



NVIDIA GPU Cloud containerizes GPU-optimized frameworks, applications, runtimes, libraries, and operating system, available at no charge



DGX A100

- DGX A100 is NVIDIA's third generation dedicated AI system. It provides a massive 5 PetaFLOPS of computing power in one system.
- Its Tensor Core architecture enables AVolta V100 GPUs to use mixed-precision multiply-accumulate operations, to significantly accelerate training for large neural networks.
- There are two models of the A100, one with 320GB system RAM and the other with 640GB RAM.

GPU	8x NVIDIA A100 GPUs
GPU Memory	Depending on model: 320GB 640GB
Power	6,500MW
CPU	2x AMD Rome 7742, 128 cores, 2.25 GHz (boosts up to 3.4 GHz)
System Memory	Depending on model: 1TB 2TB
Operating System Storage	2x 1.92TB M.2 NVME drive
Internal Storage	U.2 NVMe drives. Storage capacity depending on model: 15TB 30TB
Operating System	Canonical Ubuntu, Red Hat Enterprise Linux, or CentOS

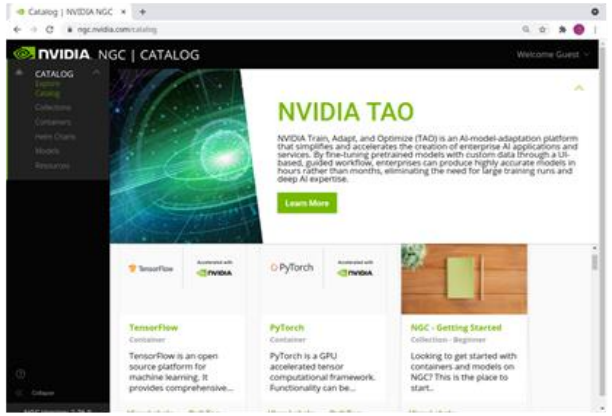
Hardware Architecture

- DGX A100 is a powerful system on its own, but a main focus of its design is to enable massive scalability. It can be used to build large AI infrastructure clusters, using the DGX SuperPOD deployment pattern (see below). DGX A100 supports elastic scalability, enabling users to:
- Scale up by connecting up to thousands of DGX A100 systems together
- Scale down by splitting each GPU into seven separate GPU instances, each with its own cache, memory, and compute resources, using NVIDIA multi-Instance GPU (MIG) technology
- Massively parallel GPU workloads depend on very high I/O performance. To this end, DGX A100 provides:
- Next generation NVLink—10x faster than 4th generation PCIe
- NVSwitch—8 Mellanox ConnectX-6 HDR InfiniBand adapters, each of them running at 200 GB/s
- Magnum IO software SDK—makes it possible to distribute workloads across thousands of GPUs
- This makes it possible to use DGX A100 for the most demanding deep learning use cases, such as conversational AI, image and video classification. These workloads can be distributed across hundreds or thousands of nodes, as needed.

SYSTEM SPECIFICATIONS	
GPUs	8x NVIDIA A100 Tensor Core GPUs
GPU Memory	320 GB total
Performance	5 petaFLOPS AI 10 petaOPS INT8
NVIDIA NVSwitches	6
System Power Usage	6.5kW max
CPU	Dual AMD Rome 7742, 128 cores total, 2.25 GHz (base), 3.4 GHz (max boost)
System Memory	1TB
Networking	8x Single-Port Mellanox ConnectX-6 VPI 200Gb/s HDR InfiniBand 1x Dual-Port Mellanox ConnectX-6 VPI 10/25/50/100/200Gb/s Ethernet
Storage	OS: 2x 1.92TB M.2 NVME drives Internal Storage: 15TB (4x 3.84TB) U.2 NVME drives
Software	Ubuntu Linux OS
System Weight	271 lbs (123 kgs)
Packaged System Weight	315 lbs (143kgs)
System Dimensions	Height: 10.4 in (264.0 mm) Width: 19.0 in (482.3 mm) MAX Length: 35.3 in (897.1 mm) MAX
Operating Temperature Range	5°C to 30°C (41°F to 86°F)

Software Architecture

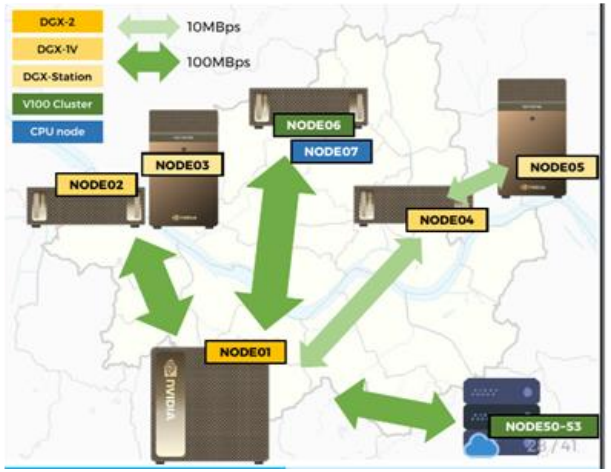
- Like previous generations, DGX-100 uses a containerized architecture:
  - A minimal operating system and driver install
  - All applications and SDKs provisioned as containers, accessible through NGC Privacy Registry
- The NGC Private Registry provides:
  - Containers enabling deep learning, machine learning, high performance computing (HPC), and deep learning orchestration with Kubernetes
  - Applications with pretrained models, scripts, Kubernetes Helm charts
  - Ability to store custom containers, model code, scripts and Helm charts and share them within the organization



DGX SuperPOD – Multinode

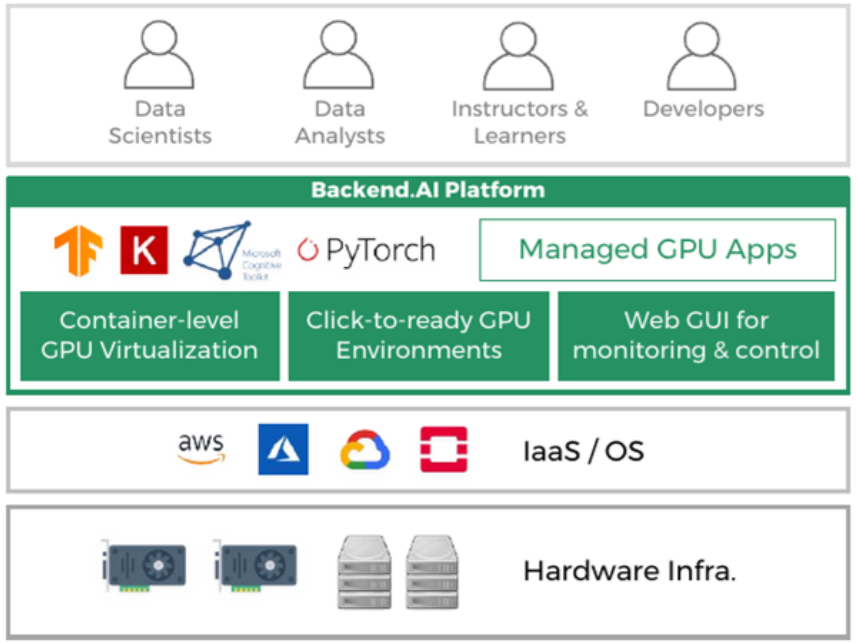
- NVIDIA DGX SuperPOD is a full-stack computing platform including compute, storage, networking, and tools to support data science pipelines. It comes with an implementation service from NVIDIA, to assist with deployment and ongoing maintenance.
- SuperPOD can support up to 140 DGX A100 systems, combined into one AI infrastructure cluster. The cluster's capabilities are as follows:

Computing Power	100-700 PetaFLOPS
Number of DGX A100 nodes	20-140
Number of GPUs	160-1120
Storage	1-10 Petabytes
NVIDIA Mellanox Bandwidth	200 Gbps



DGX A100 SuperPod

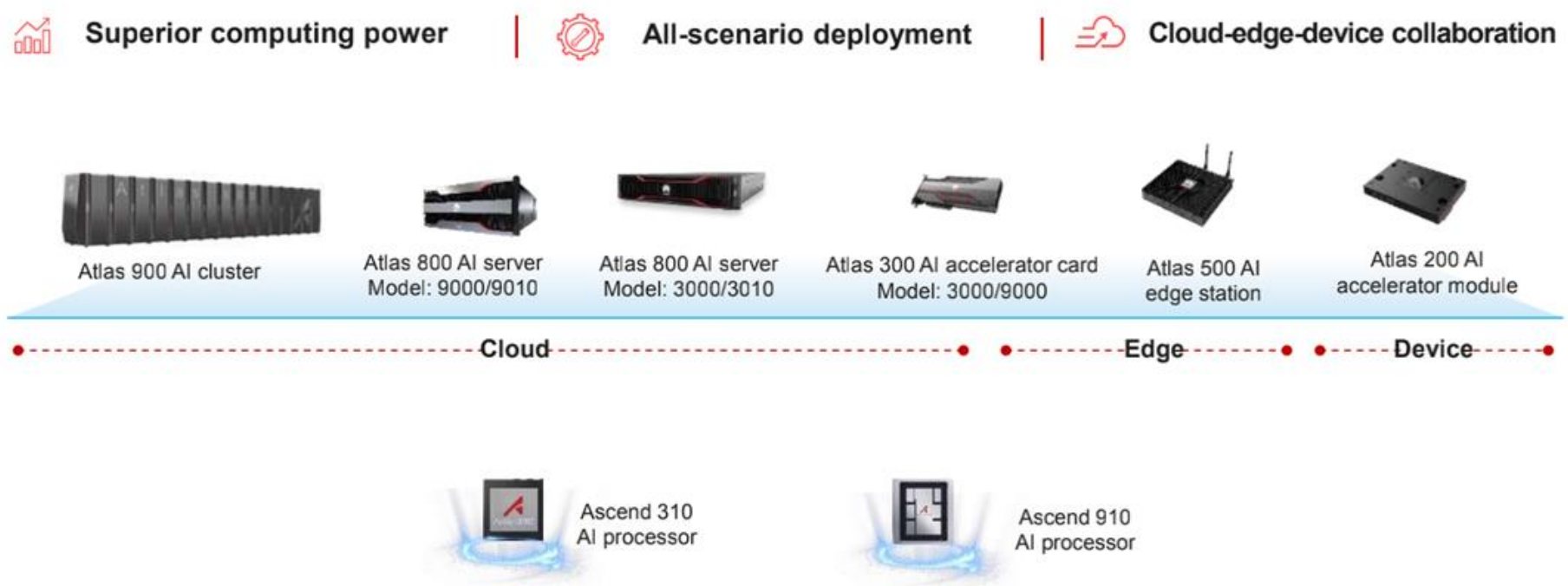
- Software Architecture
- The NVIDIA SuperPod solution includes the following software tools that support the AI cluster, in addition to AI tooling provided on each DGX-100 system:
- DGX operating system based on Ubuntu Linux, optimized and tuned specifically for DGX hardware. The DGX OS includes certified GPU drivers, network software, NFS caching, NVIDIA data center GPU management (DCGM), GPU-enabled container runtime, CUDA SDK, and support for NVIDIA GPUDirect.
- Cluster management and orchestration tools—DGX POD management software for workload scheduling, and third-party certified cluster management and orchestration tools such as Run:AI tested to work on DGX POD racks.
- Kubernetes container orchestration—DGX POD management software runs on Kubernetes for fault tolerance and high availability. It provides automated network configuration (DHCP) and automated provisioning and updates of DGX OS and software over the network (PXE).



Penerapan Model Berbasis Pola: AI Infrastructure, Platform (2/5)

Atlas AI Computing Platform Portfolio

(Huawei)

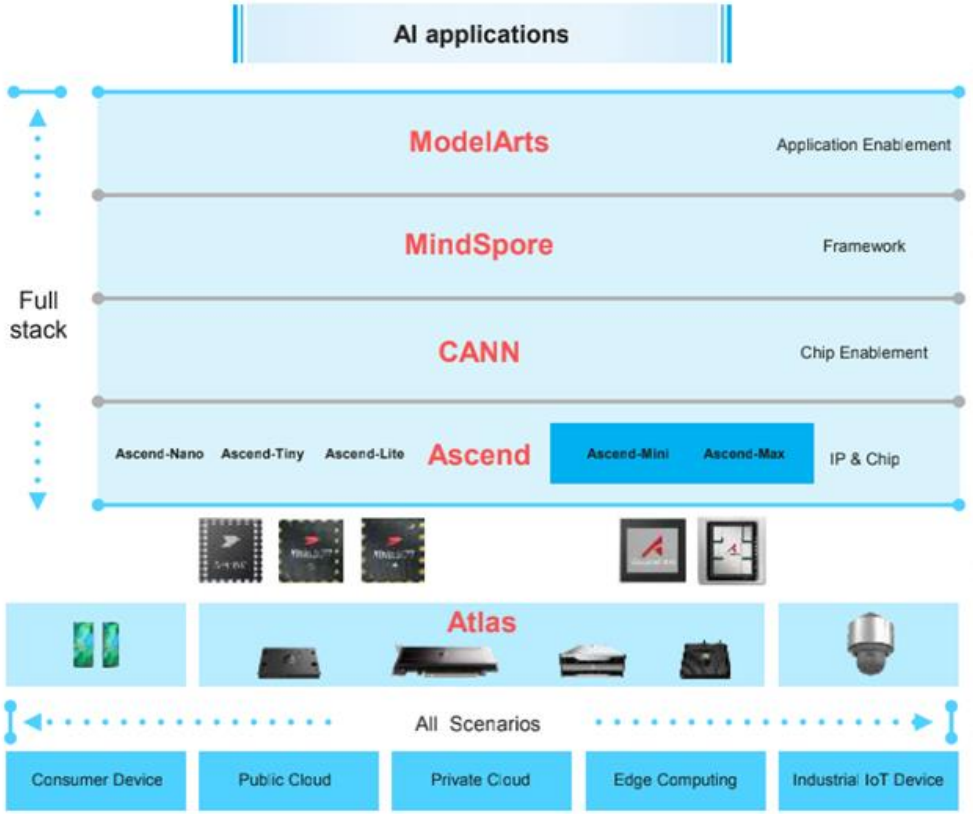


HUAWEI

Penerapan Model Berbasis Pola: AI Infrastructure, Platform (3/5)

Huawei's Full-Stack, All-Scenario AI Solution

(Huawei)



Application enablement: whole-process services (ModelArts), layered APIs, and pre-integration solution

MindSpore: unified training and inference framework for device/edge/cloud (independent or collaborative)

CANN: chip operator library and highly automated operator development tool

Ascend: a series of AI IPs and chips with unified and scalable architecture

Atlas: various products built on Huawei Ascend AI processors for device-edge-cloud AI infrastructure for all scenarios

19

HUAWEI

Huawei Device

Atlas 800 Inference Server

Model: 3010

Powered by the Intel processors, the Atlas 800 inference server (model: 3010) supports up to 7 Atlas 300I inference cards for 560-channel real-time HD video analytics. It is widely used for AI inference in data centers.

Flexible configuration for various workloads

- Supports any combination of SAS/SATA/NVMe/M.2 SSD drives
- Supports LAN on motherboard (LOM) and FlexIO cards, providing rich network interface options

Smart video analysis

- Supports up to 7 Atlas 300I inference cards and 560-channel real-time HD video analytics (1080p 25 FPS)

Specifications

Form Factor	2U AI server
Processor	1 or 2 Intel® Xeon® Skylake or Cascade Lake Scalable processors, 205 W TDP
Processor Memory	24 DDR4 DIMM slots, up to 2933 MT/s
AI Accelerator Card	Up to 7 Atlas 300I inference cards
AI Computing Power	Up to 616 TOPS INT8
Local Storage	8 x 2.5" SAS/SATA drives 12 x 3.5" SAS/SATA drives 8 x 2.5" SAS/SATA + 12 x 2.5" NVMe 24 x 2.5" SAS/SATA drives 24 x 2.5" NVMe 25 x 2.5" SAS/SATA drives
RAID	RAID 0, 1, 5, 6, 10, 1E, 50, or 60
PCIe	10 PCIe Gen3.0 (including 1 RAID controller card and 1 FlexIO)
Power Supply	2 hot-swappable PSUs, with support for 1+1 redundancy. Supported options include: 550 W AC Platinum PSUs, 900 W AC Platinum/Titanium PSUs, and 1500 W AC Platinum PSUs 1500 W 380 V HVDC PSUs, 1200 W -48 V to -60 V DC PSUs
Fan Modules	4 hot-swappable fan modules, supporting N+1 redundancy
Operating Temperature	5°C to 45°C
Dimensions (H x W x D)	Chassis with 3.5" drives: 748 mm x 86.1 mm x 447 mm Chassis with 2.5" drives: 708 mm x 86.1 mm x 447 mm

Application Scenarios

Deployed in data centers to enable AI inference

Precision marketing · Medical image analytics · Video analytics · OCR

Smart retail · Smart healthcare · Smart city · Smart finance

Building a Fully Connected, Intelligent World

HUAWEI

Atlas 800 Training Server

Model: 9010

The Atlas 800 training server (model: 9010) is an AI training server based on the Intel processors and Huawei Ascend 910 processors. It features the industry's highest computing density and high network bandwidth. The server is widely used in deep learning model development and training scenarios, and is an ideal option for computing-intensive industries, such as smart city, intelligent healthcare, astronomical exploration, and oil exploration.

The ultimate computing density

- Up to 2.24 PFLOPS FP16 in a 4U space
- 1.5x the computing density of industry peers

High-speed network

- 8 100G RoCE v2 ports, slashing cross-server chip interconnect latency by 10–70%

Specifications

Form Factor	4U AI server
Processor	2 Intel V5 Cascade Lake processors
Processor Memory	Up to 24 DDR4 DIMM slots, supporting RDIMMs
AI Processor	8 Ascend 910 processors
HBM	8 * 32 GB
AI Computing Power	2.24 PFLOPS FP16 2 PFLOPS FP16
Local Storage	• 2 x 2.5" SATA + 8 x 2.5" SAS/SATA • 2 x 2.5" SAS/SATA + +6 x 2.5" NVMe
RAID	RAID 0, 1, 10, 5, 50, 6, or 60
Network	8 100GE 1 OCP NIC 3.0 standard card, supporting 2 25GE
PCIe Expansion	Up to 2 PCIe 3.0 x16 and 4 PCIe 3.0 x8 slots
Power Supply	4 hot-swappable 2 kW or 3 kW AC PSUs, supporting 2+2 redundancy
Power Consumption	Maximum: 5.6 kW[1]
Cooling Mode	Air cooling
Fan Modules	8 hot-swappable fan modules, supporting N + 1 redundancy
Operating Temperature	5℃ to 40℃ (Liquid Cooling) 5℃ to 35℃ (Air Cooling)
Dimensions (H x W x D)	790 mm x 175 mm x 447 mm

1. This specification item is in continuous optimization. The value is dynamically updated based on the optimization result.

Application Scenarios

Deployed in data centers to enable AI training

Model training · HPC

Smart city · Smart healthcare

Astronomical exploration · Oil exploration

Building a Fully Connected, Intelligent World

HUAWEI

Proses pada Ascend

Figure 3-1 Engine process

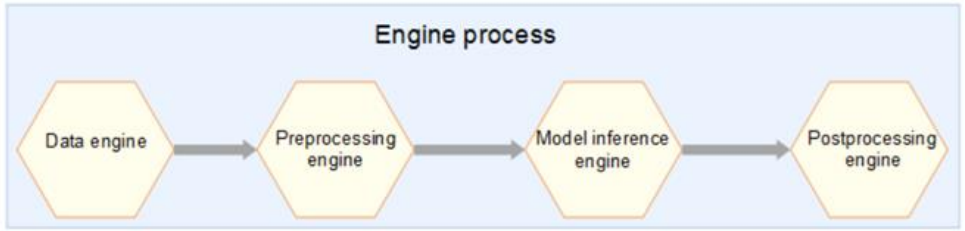
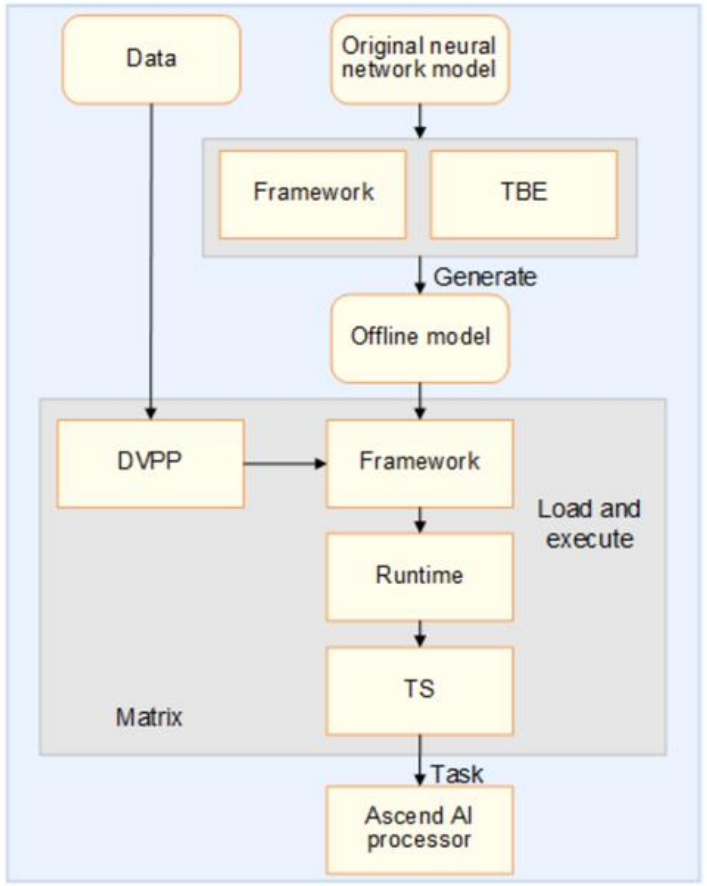
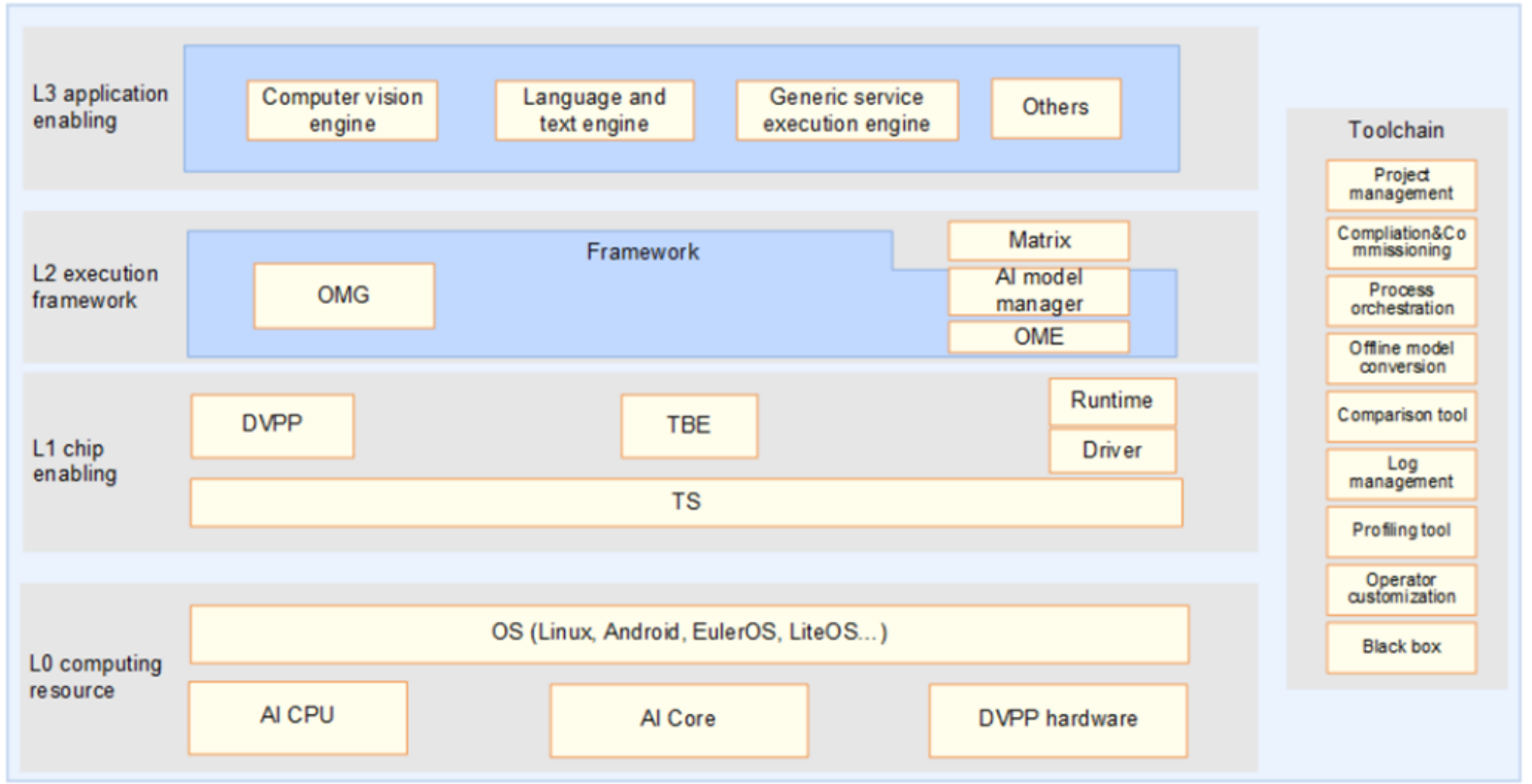


Figure 2-1 Neural network software architecture

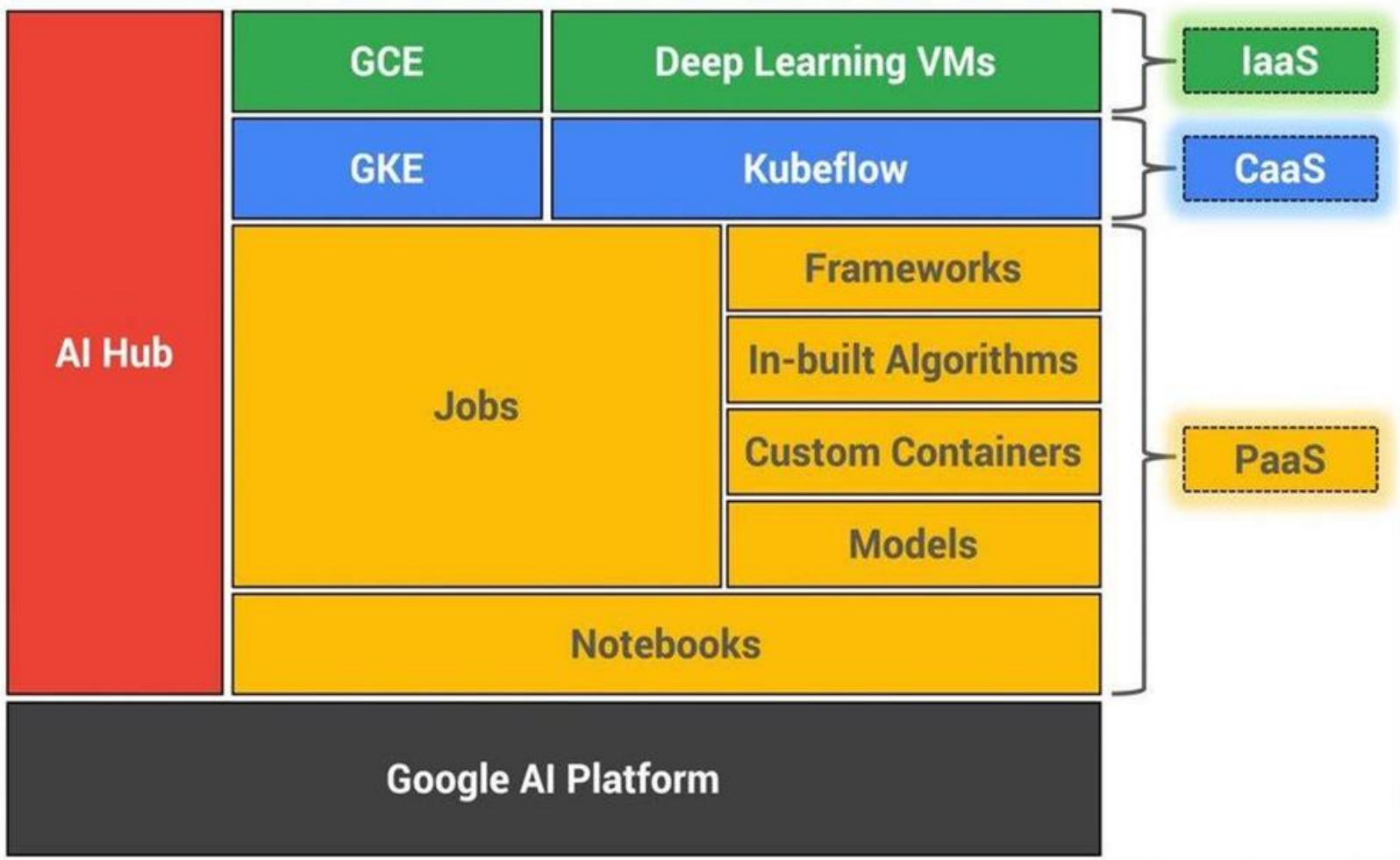


Software Stack

Figure 1-1 Logical architecture of the Ascend AI software stack

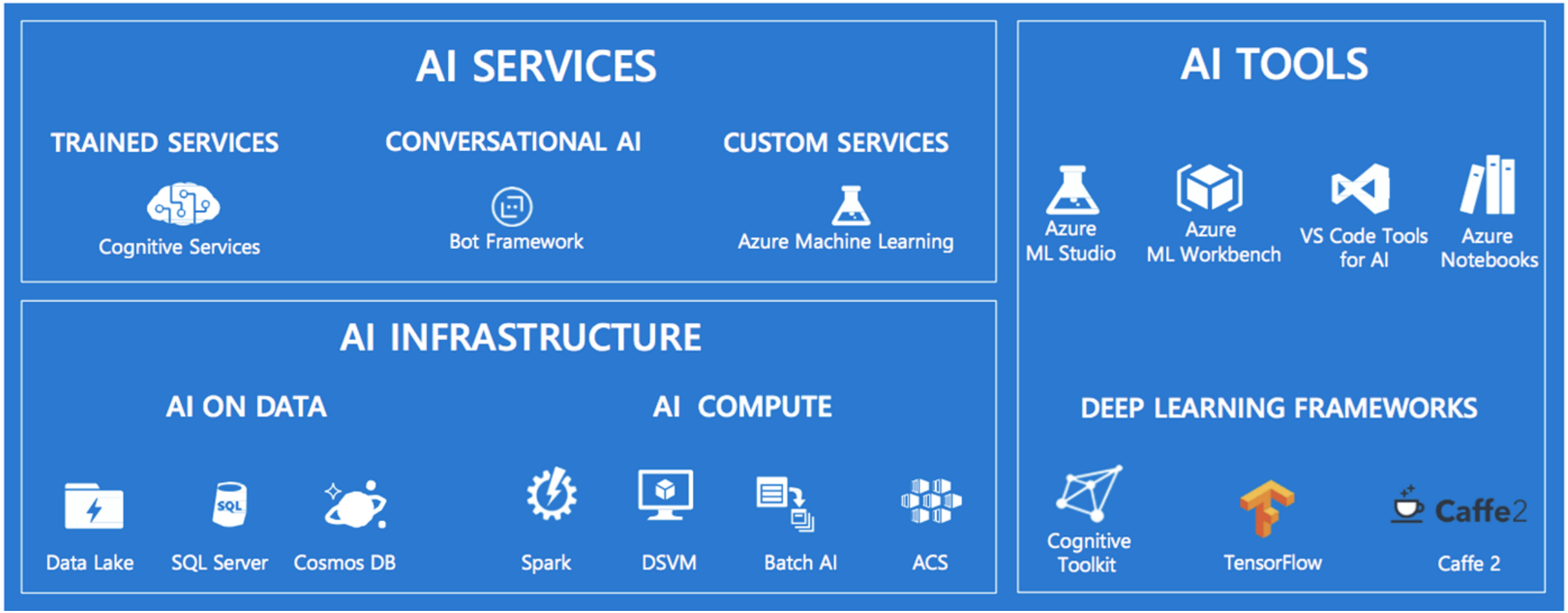


Penerapan Model Berbasis Pola: AI Infrastructure, Platform (4/5)

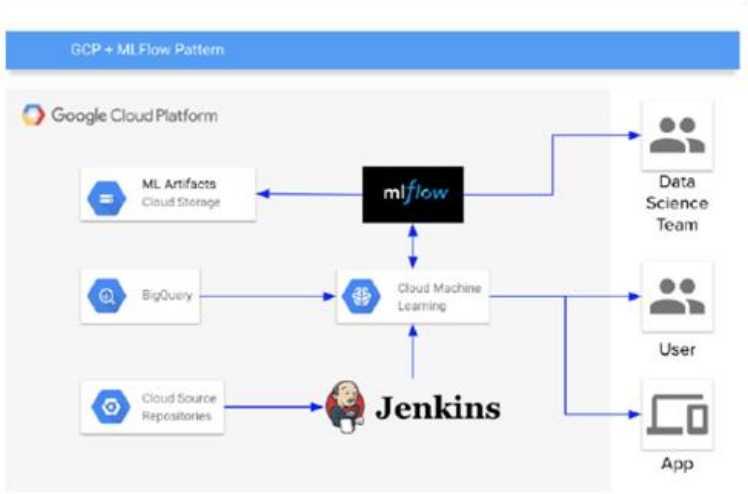


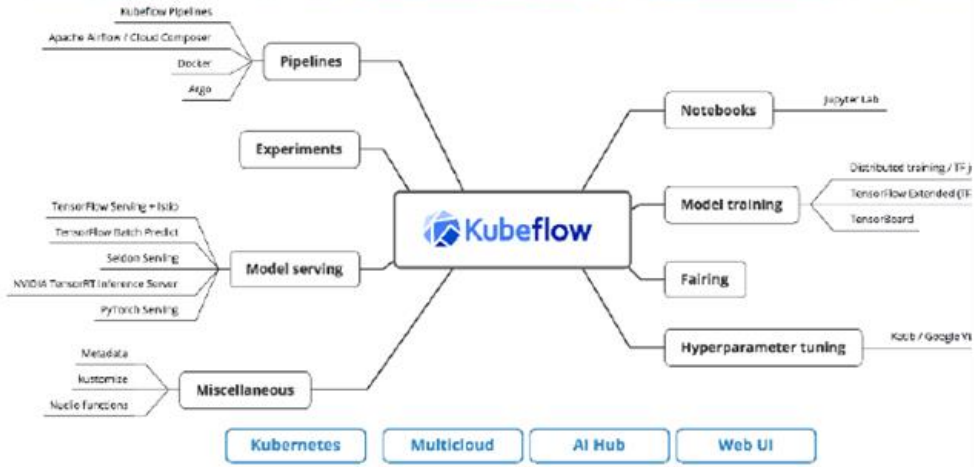
(Google)

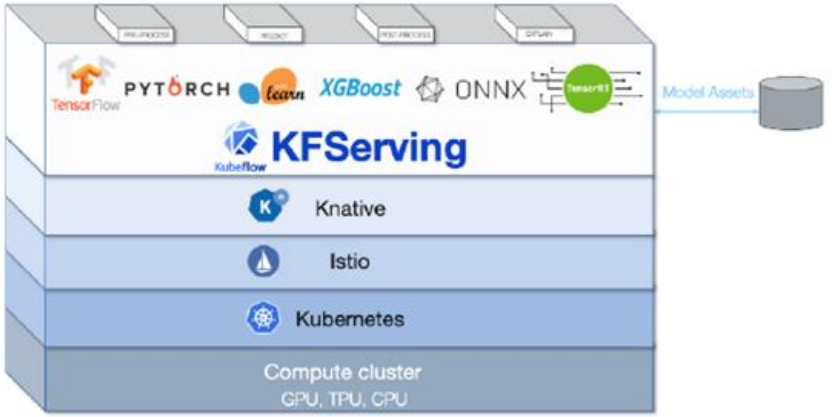
Penerapan Model Berbasis Pola: AI Infrastructure, Platform (5/5)



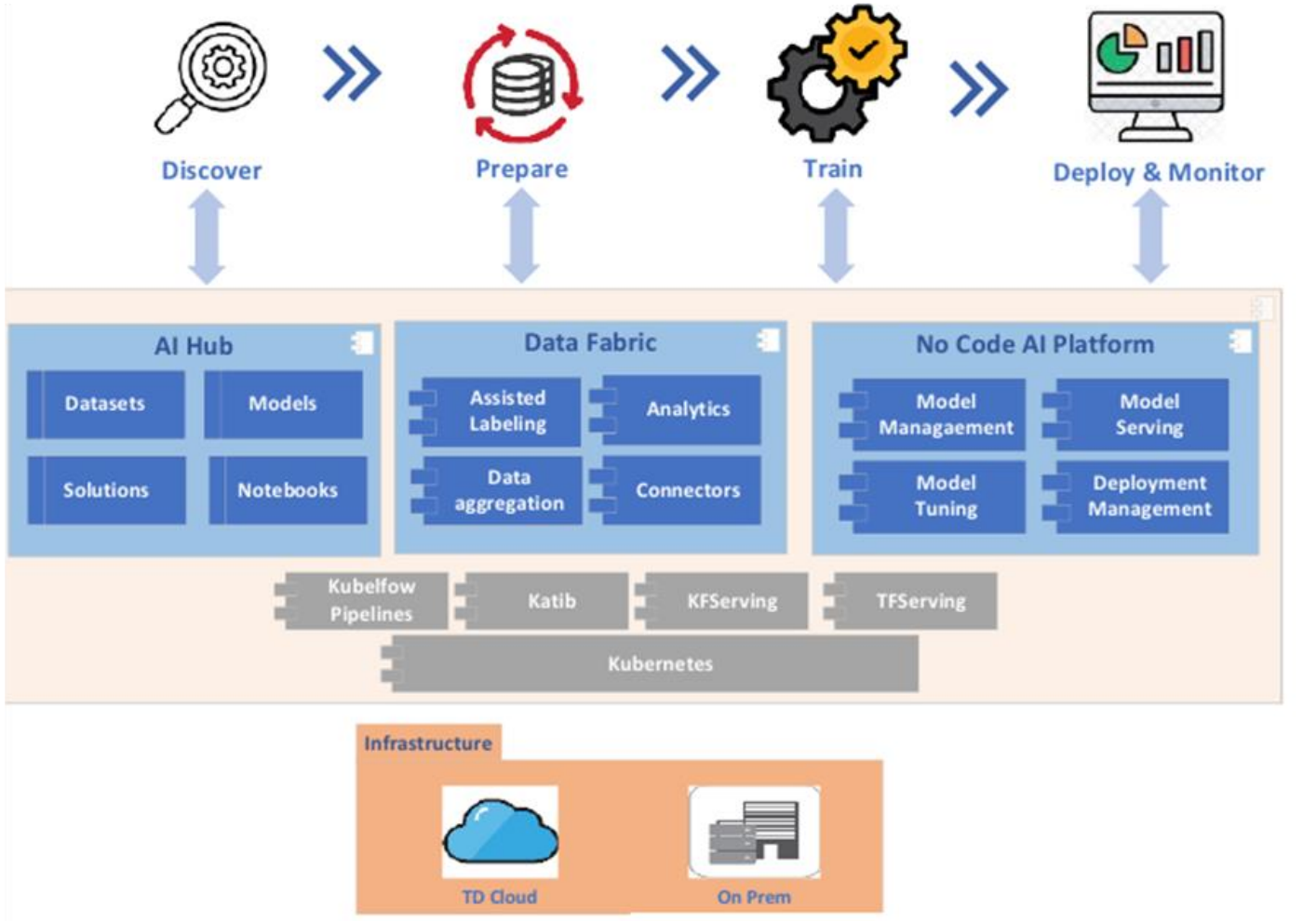
Tools untuk MLOps



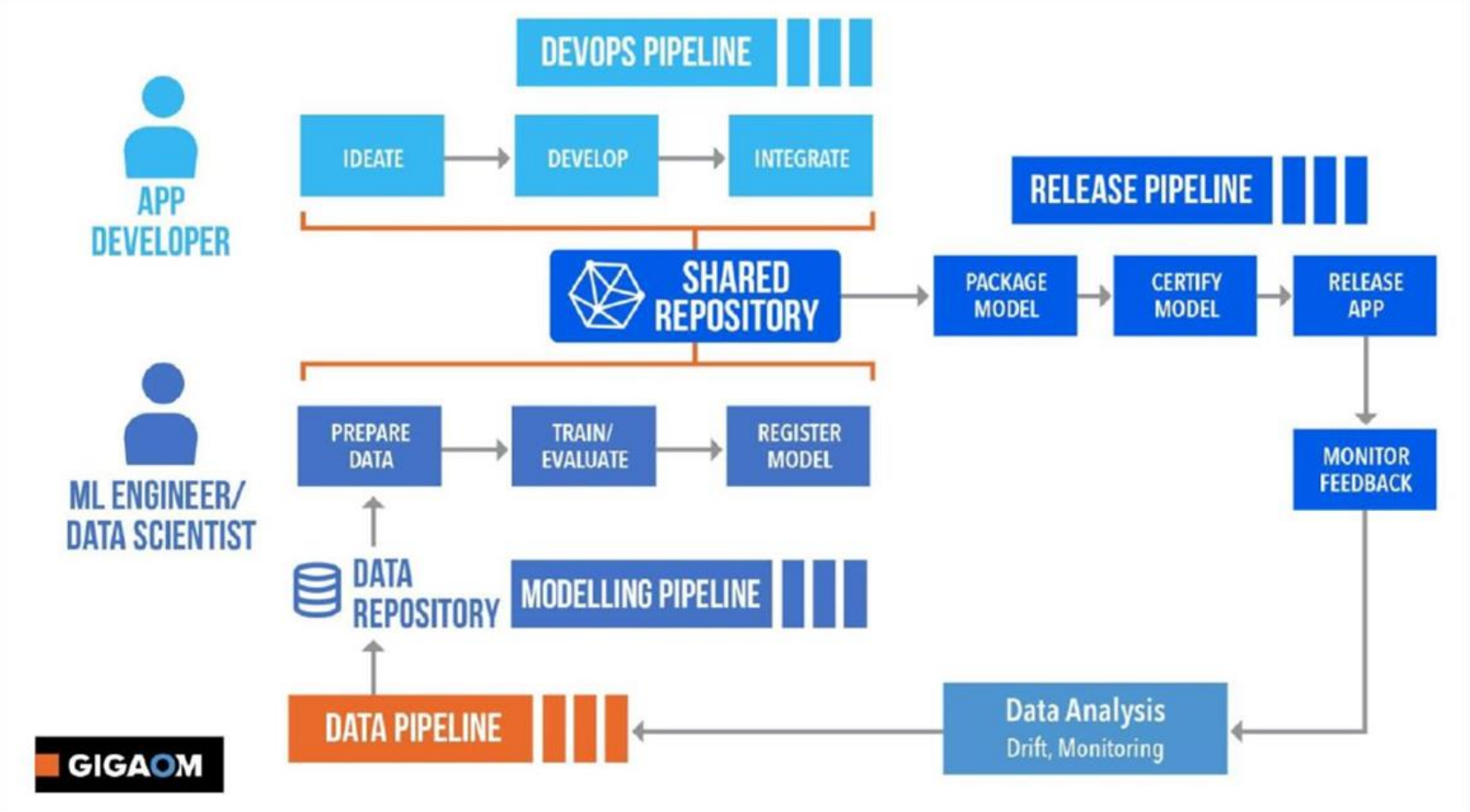




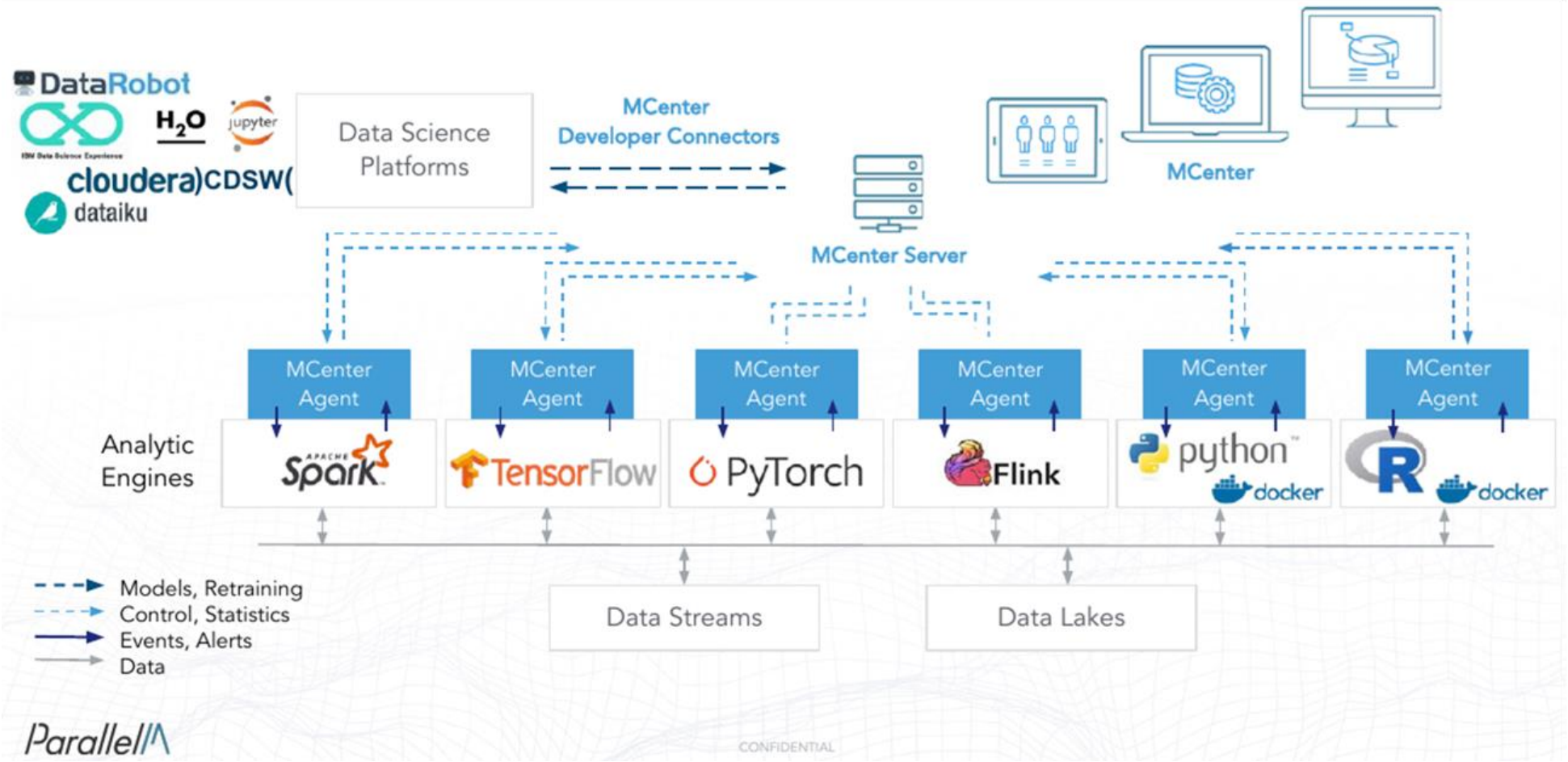
Environment utk Produksi



Siklus Terintegrasi

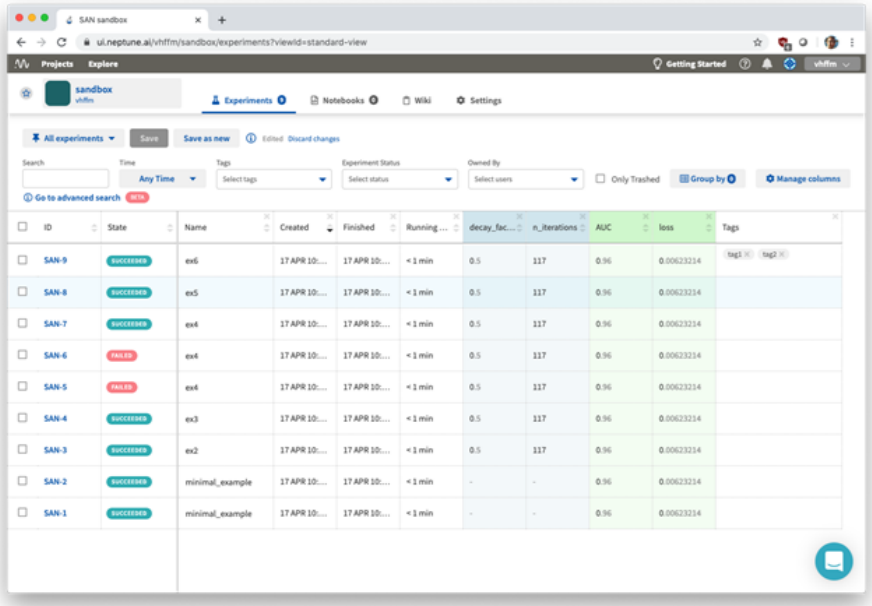


Contoh Environment Produksi



Neptune

- Neptune is a tracker for experiments with focus on Python. It is a hosted service.
- Experiments are tracked by using library hooks to register (model) parameters, evaluation results, and upload artifacts (such as models, hashes of training data, or even code). The library can track hardware usage and experiment progress.
- The results can be analyzed and compared on a website. There are also collaborative options. Neptune has integrations with Jupyter notebooks, various ML libraries, visualizers (HiFlow, TensorBoard), other trackers (MLFlow), and external offerings (Amazon Sagemaker).
- They provide an API to query results from experiment. This can be used to feed CI/CD pipelines for model deployment.

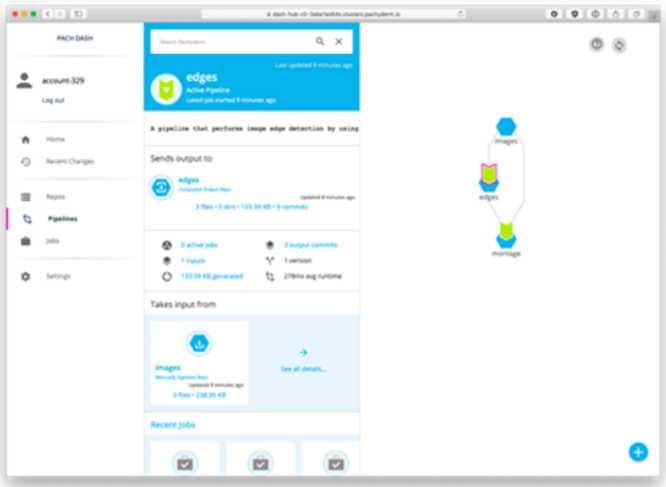



```
$ conda create --name neptune python=3.6
$ conda activate neptune
$ conda install -c conda-forge neptune-client
$ cd /somewhere

---> https://ui.neptune.ai/
---> Create Account, Log In, “Getting Started”
```

Pachyderm

- Pachyderm is a versioning system and execution environment for data and processing pipelines. Hosted and self-hosted options exist.
- At its core is data provenance. Data is committed to a repository and acted upon by processors in pipelines. The results (data and other artefacts like models) are committed back into a repository. By design, all use of data is traceable through pipelines. Pipelines are described in JSON and processors are packaged as Docker images. Pachyderm can integrate (but not deploy) Jupyter and can push/pull data from cloud stores (S3, etc).
- Pachyderm is built on top of Kubernetes, so can easily scale-out and run in various clouds. Self-hosting comes with the usual Kubernetes complexity.

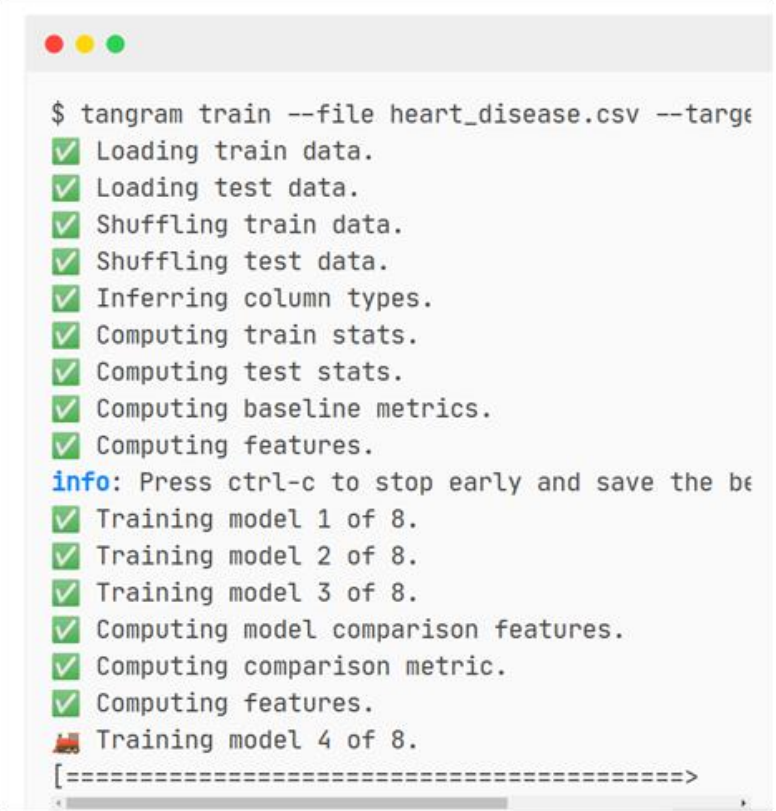


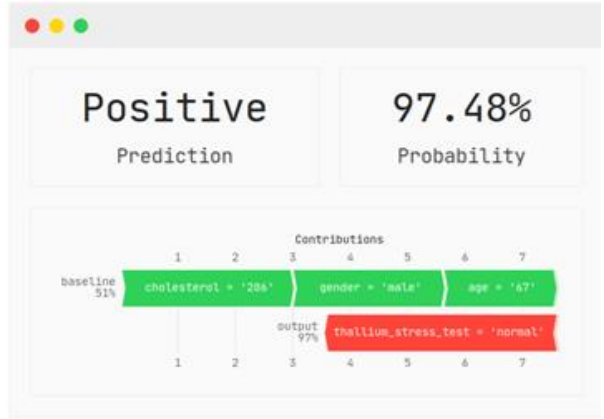

```
$ cd /somewhere
$ download from (https://github.com/pachyderm/pachyderm/releases)
$ tar xfvz release_filename_linux_amd64.tar.gz
$ pachctl version --client-only

--> https://docs.pachyderm.com/latest/pachub/pachub_getting_started/
--- https://docs.pachyderm.com/latest/getting_started/beginner_tutorial/
```

Tangram

- http://tangram.dev
- Train a model on the command line.Train a machine learning model by runningÂ tangram trainÂ with the path to a CSV file and the name of the column you want to predict.
- The CLI automatically transforms your data into features, trains a number of linear and gradient boosted decision tree models to predict the target column, and writes the best model to aÂ .tangramÂ file. If you want more control, you can provide a config file.





Solusi Cloud One-Stop

Rekomendasi

Track Experiments
Archive Some Models
Work on Low Hanging Fruits

Try *Neptune* or *MLFlow*.

Expect Scaling?
Integrate w/ Google Cloud?
Running Kubernetes?

Try *Kubeflow*.

!!! Complex !!!

Want pipelines?
Want containers?
Want data provenance?

Try *Pachyderm*.

Deploy models?
Track data?
Need end-to-end?

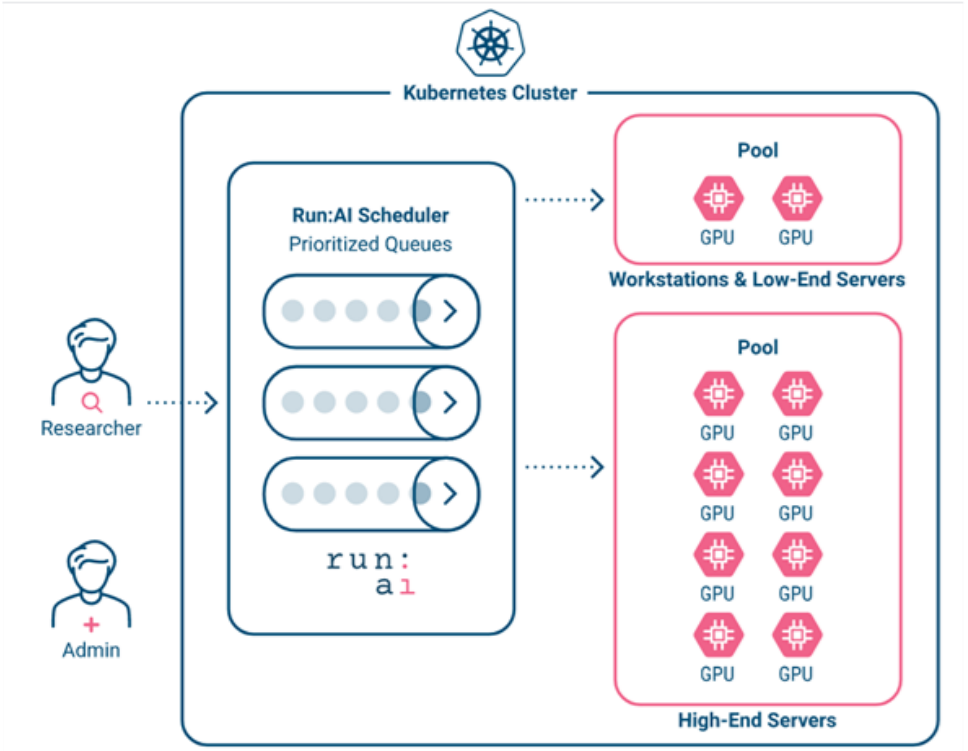
Try *Dataiku*, *SageMaker*, *AzureML*, *DataBricks*, or *Google AI Platform*.

No idea?

Try *Neptune* or *MLFlow*.

Run:AI

- Run:AI pools heterogeneous resources so they can be used within two logical environments, build and train, to natively support data scientists' different compute characteristics and increase utilization. The GPU virtualization pool exists inside a Kubernetes cluster. The two logical environments interact with the Run:AI scheduler for build and training workloads.



Run.AI

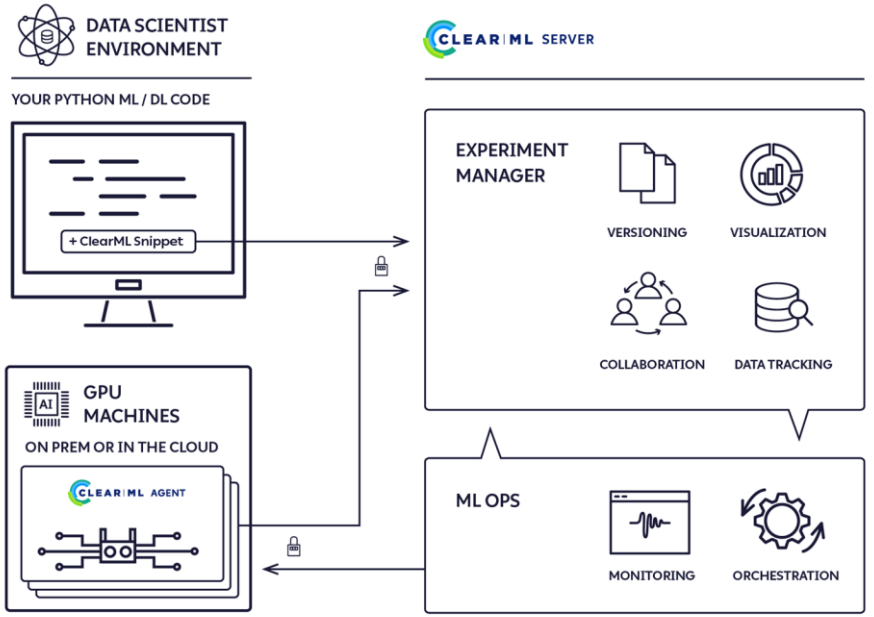
- **Build environment –** dedicated for building models interactively, typically using jupyter notebooks or Pycharm, or simply by SSH-ing into a container. Performance in build environments is typically less critical so build workloads can be run on workstations or low-end servers.
- **Training environment –** dedicated for long training workloads. As performance is important in training, these workloads should run on high-end GPU servers. Containers for training can be supplemented with a checkpointing mechanism that allows automatic preemption and resume without losing the state of the training. Run:AI creates a virtual pool of GPUs which can easily be shared among all users. With Run:AI, users can actually go over their guaranteed quota and use more GPUs than they are assigned.

Run.AI

- By pooling the resources and managing them using the Run:AI scheduler, administrators gain control. They can easily onboard new users, maintain and add new hardware to the pool, and gain visibility, including a holistic view of GPU usage and utilization.
- In addition, data scientists can automatically provision resources without depending on IT admins.

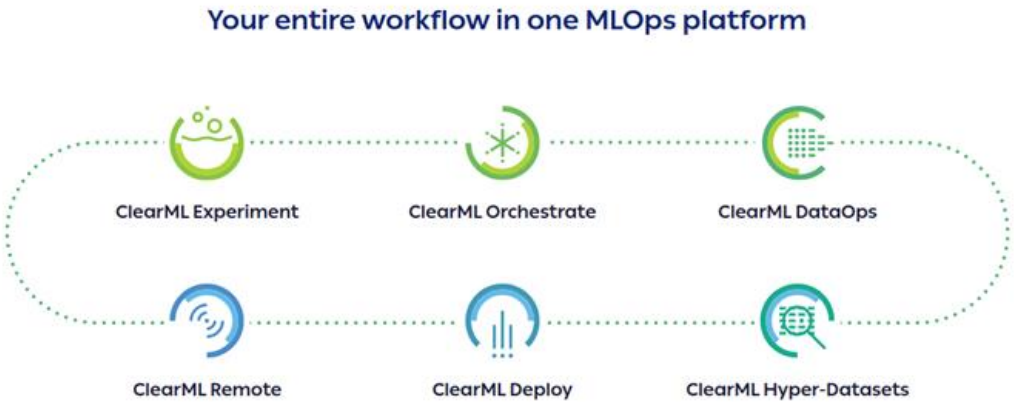
ClearML

- ClearML is an open source platform that automates and simplifies developing and managing machine learning solutions for thousands of data science teams all over the world.
- It is designed as an end-to-end MLOps suite allowing you to focus on developing your ML code & automation, while ClearML ensures your work is reproducible and scalable.

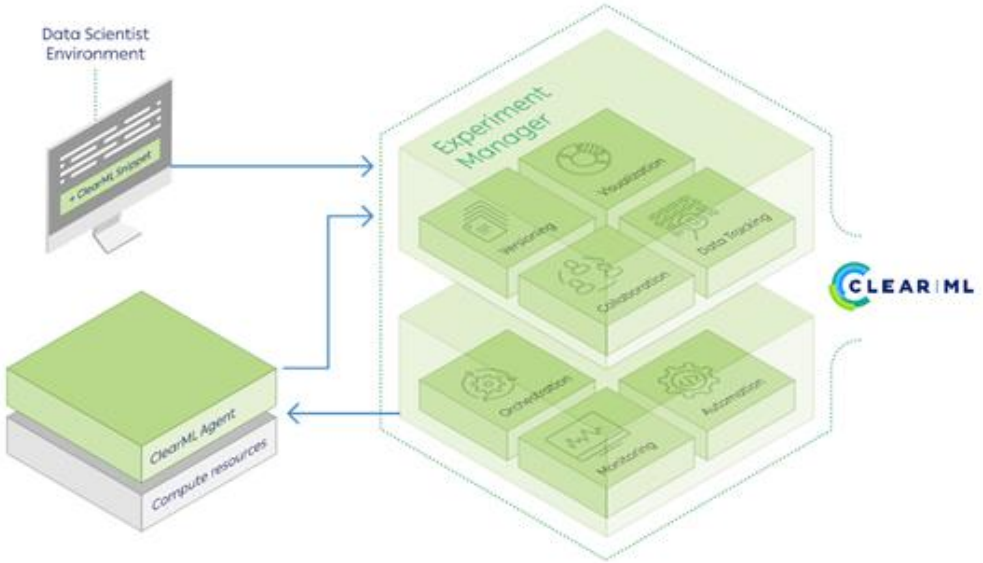


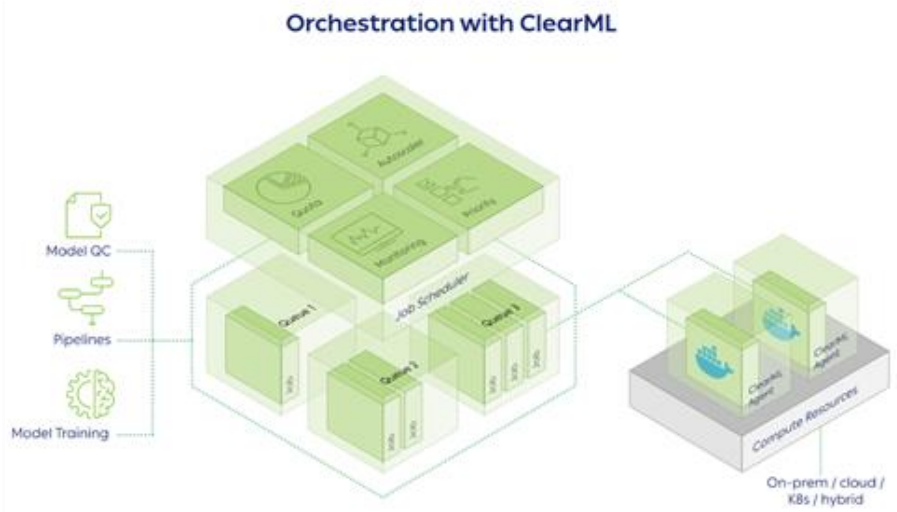
What can we do with ClearML

- Track and upload metrics and models with only 2 lines of code
- Create a bot that sends you a slack message whenever you model improves in accuracy
- Automatically scale AWS instances according to your resources needs
- Reproduce experiments with 3 mouse clicks
- Much More!

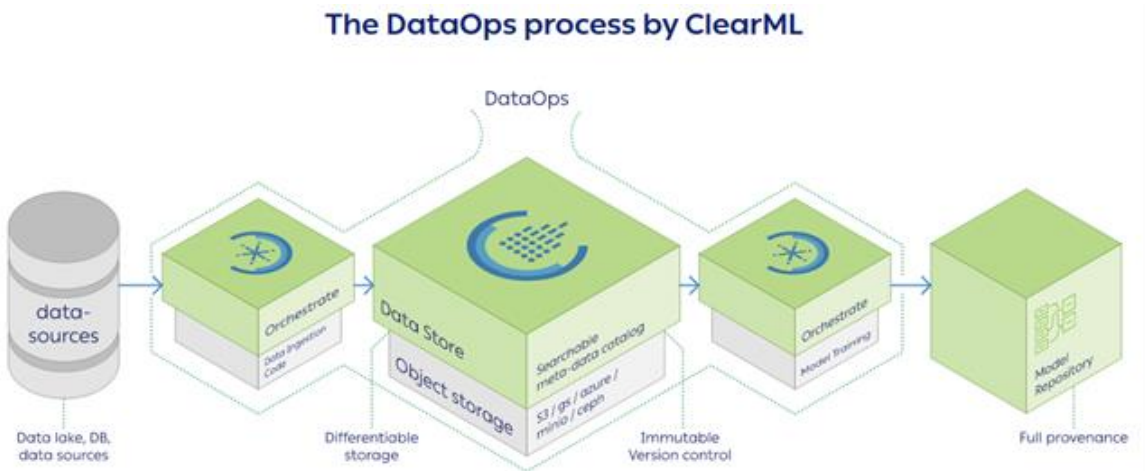


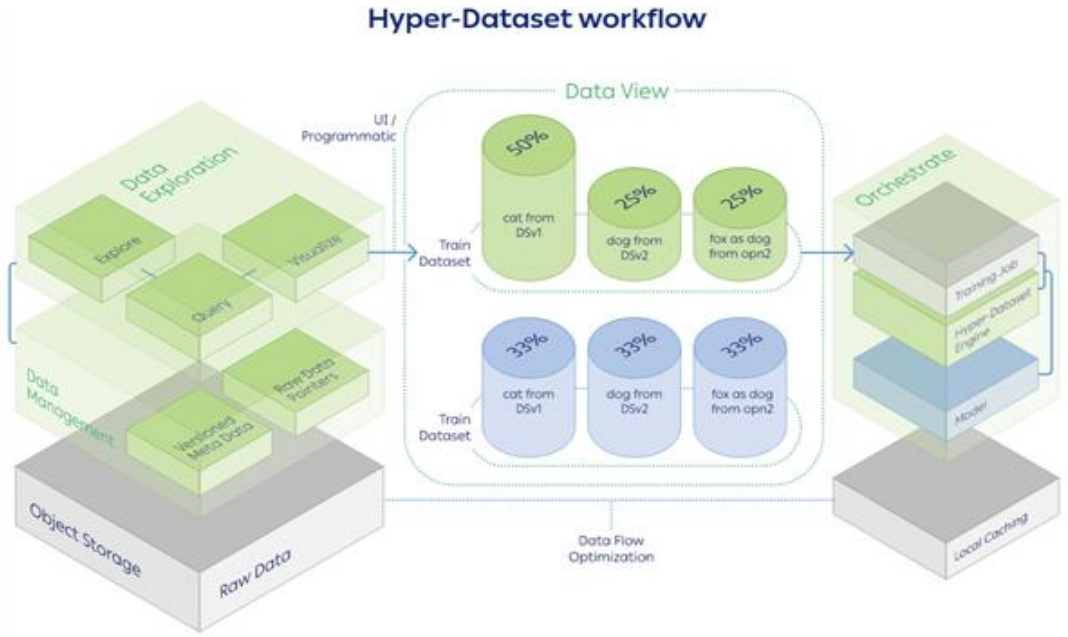
ClearML


Data Scientist Environment
Experiment Manager
Visualization
Versioning
Data Tracking
Collaboration
Orchestration
Automation
Monitoring
CLEARML
ClearML Agent
Compute resources

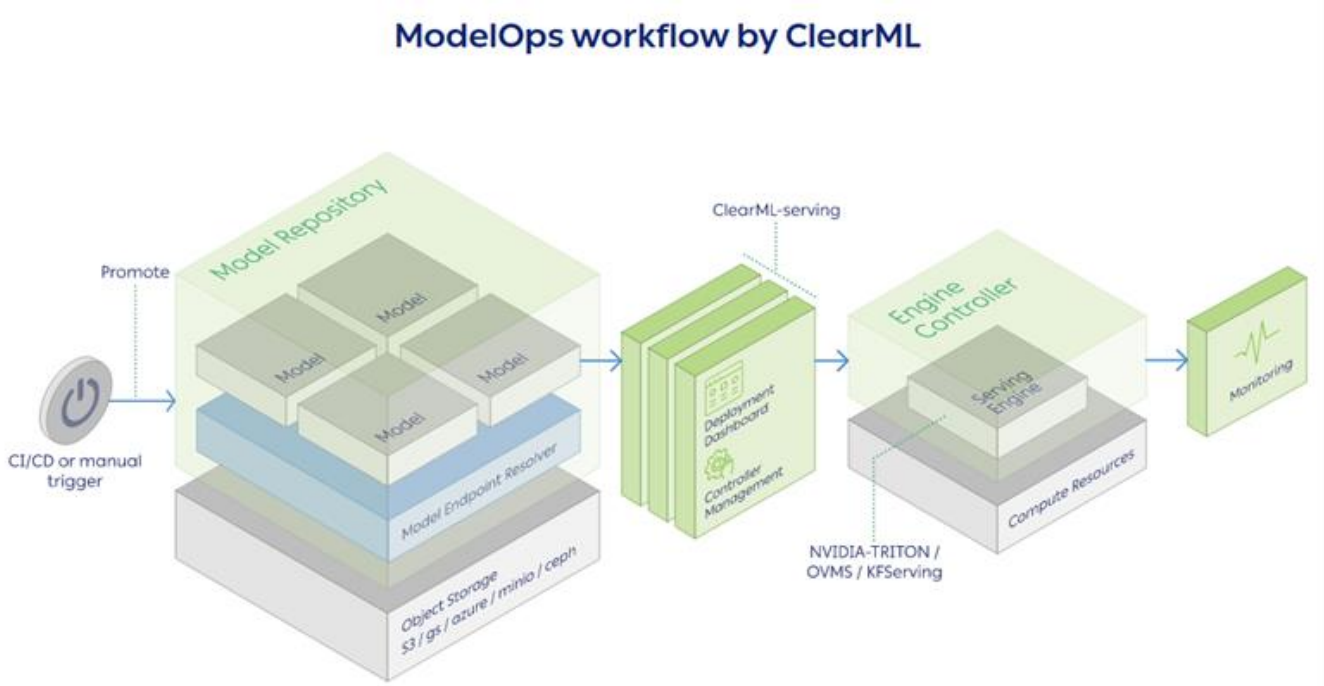

Orchestration with ClearML
Model QC
Pipelines
Model Training
Queue 1
Queue 2
Queue 3
Job Scheduler
Compute Resources
On-prem / cloud / K8s / hybrid

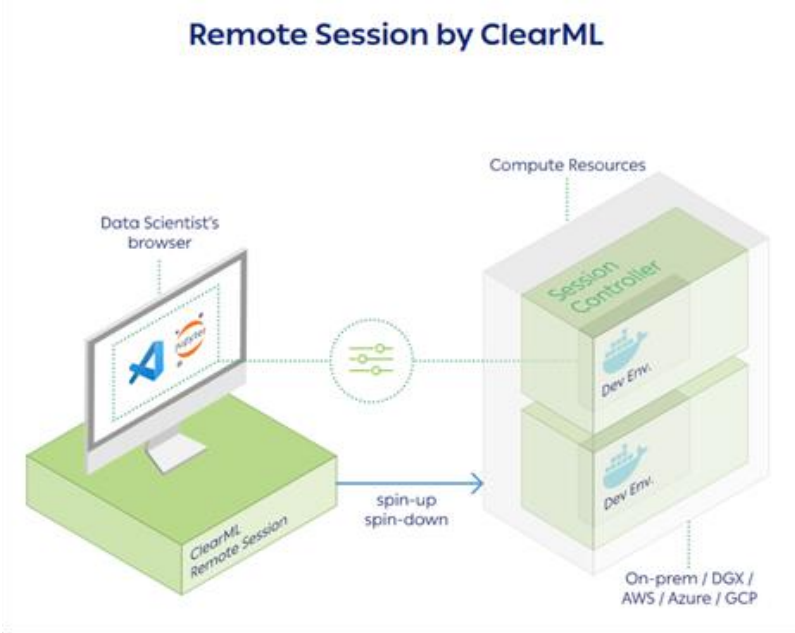
ClearML


The DataOps process by ClearML
DataOps
data-sources
Orchestrate
Data Store
Object storage
Searchable meta-data catalog
S3 / gs / azure / minio / ceph
Orchestrate
Model Repository
Data lake, DB, data sources
Differentiable storage
Immutable Version control
Full provenance


Hyper-Dataset workflow
Data View
UI / Programmatic
Data Exploration
Explore
Visualize
Query
Data Management
Raw Data Pointers
Versioned Meta Data
Object Storage
Raw Data
Train Dataset
50%
cat from DSv1
25%
dog from DSv2
25%
fox as dog from opn2
Train Dataset
33%
cat from DSv1
33%
dog from DSv2
33%
fox as dog from opn2
Data Flow Optimization
Orchestrate
Training Job
Hyper-Dataset Engine
Model
Local Caching

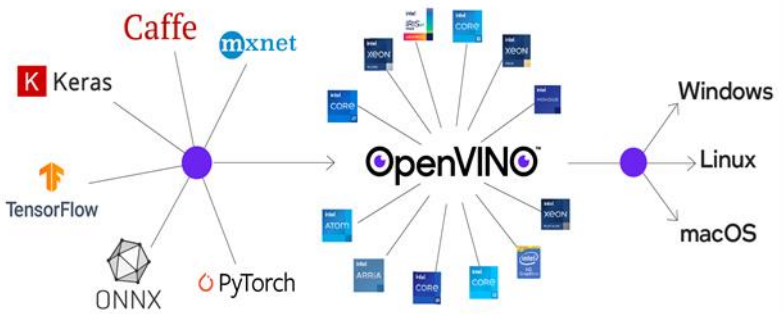
ClearML

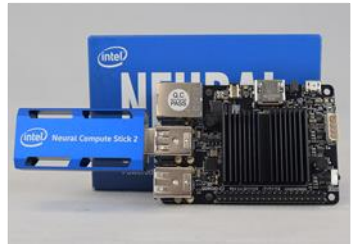

ModelOps workflow by ClearML
Promote
CI/CD or manual trigger
Model Repository
Model
Model
Model
Model
Model Endpoint Resolver
Object Storage
S3 / gs / azure / minio / ceph
ClearML-serving
Deployment Dashboard
Controller Management
Engine Controller
Serving Engine
Compute Resources
NVIDIA-TRITON / OVMS / KFServing
Monitoring


Remote Session by ClearML
Compute Resources
Data Scientist's browser
Session Controller
Dev Env.
Dev Env.
ClearML Remote Session
spin-up
spin-down
On-prem / DGX / AWS / Azure / GCP

Deployment strategy (1/6)

- Deployment strategy
  - Computational resources: melakukan prediksi (inference) lebih baik dibandingkan melakukan training.
  - Inferensi pada perangkat (berbasis Apps):
    - Performansi
    - Data dengan volume yang besar bandwidth resources
    - User tidak memiliki akses internet D
  - Dapat menggunakan OPENVINO, Jetson NANO, etc (berbasis Desktop) Robotics, Surveillance System, dll
  - Dapat menggunakan microplatform FLASK dan HEROKU (berbasis Web)
- Deployment strategy secara tipikal dapat dibagi menjadi:
  - Single Deployment
  - Silent Deployment
  - Canary Deployment





HEROKU

Deployment strategy (2/6)

Single Deployment

- Single deployment merupakan strategi yang paling sederhana.
- Model dibuat serial menjadi file, file lama bisa di replace dengan yang baru, termasuk ekstraktor fitur jika diperlukan.
- Keunggulan: sederhana
- Kelemahan: Jika ada bug, maka semua pengguna akan terpengaruh

- Deployment di cloud environment:
  - VM atau container baru harus disiapkan
  - VM image atau container lama di replace
  - Autoscaler memulai dengan yang baru

- Deployment di server:
  - Upload new model file pada server
  - Replace yang lama dengan yang baru
  - Lakukan restart web service

Deployment strategy (3/6)

Single Deployment

- Merupakan strategi dengan menjalankan versi model yang baru dan lama secara paralel.
- Keunggulan:

  Menyediakan waktu yang cukup untuk memastikan model baru dapat digunakan sesuai kriteria
- Kelemahan:

  Menghabiskan banyak resource karena menjalankan dua model secara bersama-sama

Deployment strategy (4/6)

Canary Deployment

- Merupakan strategi deployment dengan cara mendorong versi dan kode model baru ke sebagian kecil pengguna, dengan tetap menjalankan versi lama untuk sebagian besar pengguna.
- Keunggulan:
  - Memungkinkan untuk memvalidasi model baru dan efek prediksinya.
  - Deployment tidak mempengaruhi banyak user jika ada bug
- Kelemahan:

  Lebih rumit dan kompleks

Deployment strategy (5/6)

- Reproducibility adalah akuntabilitas yang diperlukan dalam bisnis untuk lebih memahami dan mempercayai penerapan machine learning pada kehidupan sehari-hari.
- **Reproducibility pada machine learning** yaitu suatu kemampuan untuk menduplikasi model secara tepat sehingga ketika model dilewati dengan data input yang sama, model yang direproduksi akan mengembalikan output yang sama.
- **Tantangan** pada proses Reproducibility ketika Deployment Model:
  - Feature tidak terdapat pada Live Environment
  - Perbedaan bahasa pemrograman
  - Perbedaan perangkat lunak
  - Kondisi real tidak cocok dengan data dan kondisi pada saat training model

Deployment strategy (6/6)

Solusi pada proses Reproducibility ketika Deployment Model:

- Versi perangkat lunak harus sama persis dan aplikasi harus mencantumkan semua dependensi library pihak ketiga beserta versinya
- Menggunakan container dan perangkat lunak untuk melacak spesifikasi
- Research, develop dan deploy menggunakan bahasa yang sama (contoh: Python)
- Sebelum membangun model, harus memahami terlebih dahulu bagaimana model akan diintegrasikan dengan sistem lain.

Deployment Model: Cara Menyimpan Model

```
H=model.fit(trainX, trainY,validation_data=(testX,
testY), batch_size=b, epochs=e, shuffle=True )

regressor.fit(X,y)
```

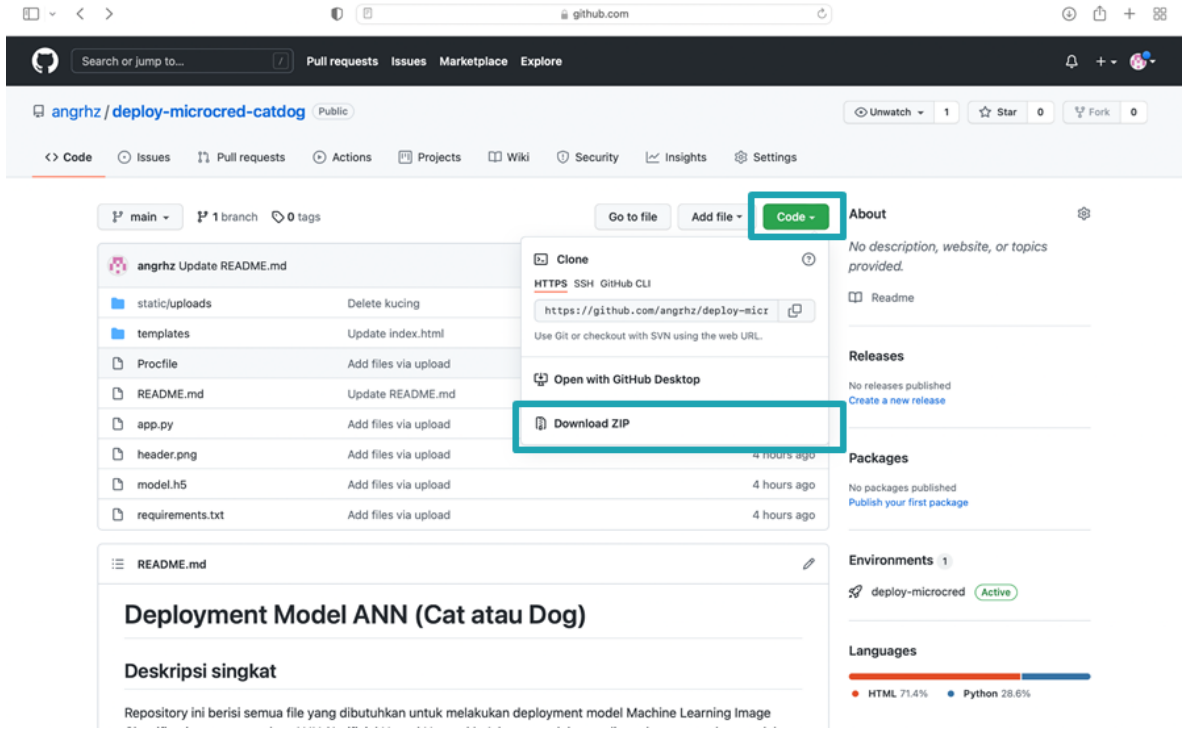
```
from keras.models import load_model
model.save('model.h5')
import pickle
pickle.dump(regressor, open('model.pkl','wb'))
```

Deployment Model: Langkah-Langkah Model Deployment (*Inferencing*)

1. Download dan lakukan installasi perangkat lunak di bawah ini

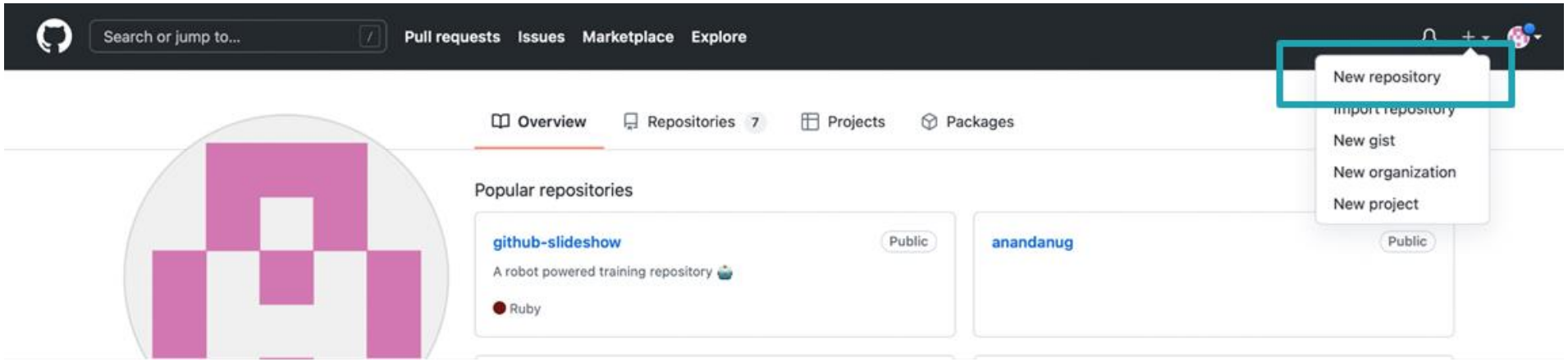
Deployment Model: Langkah-Langkah Model Deployment

2. Download source code dari repository Github



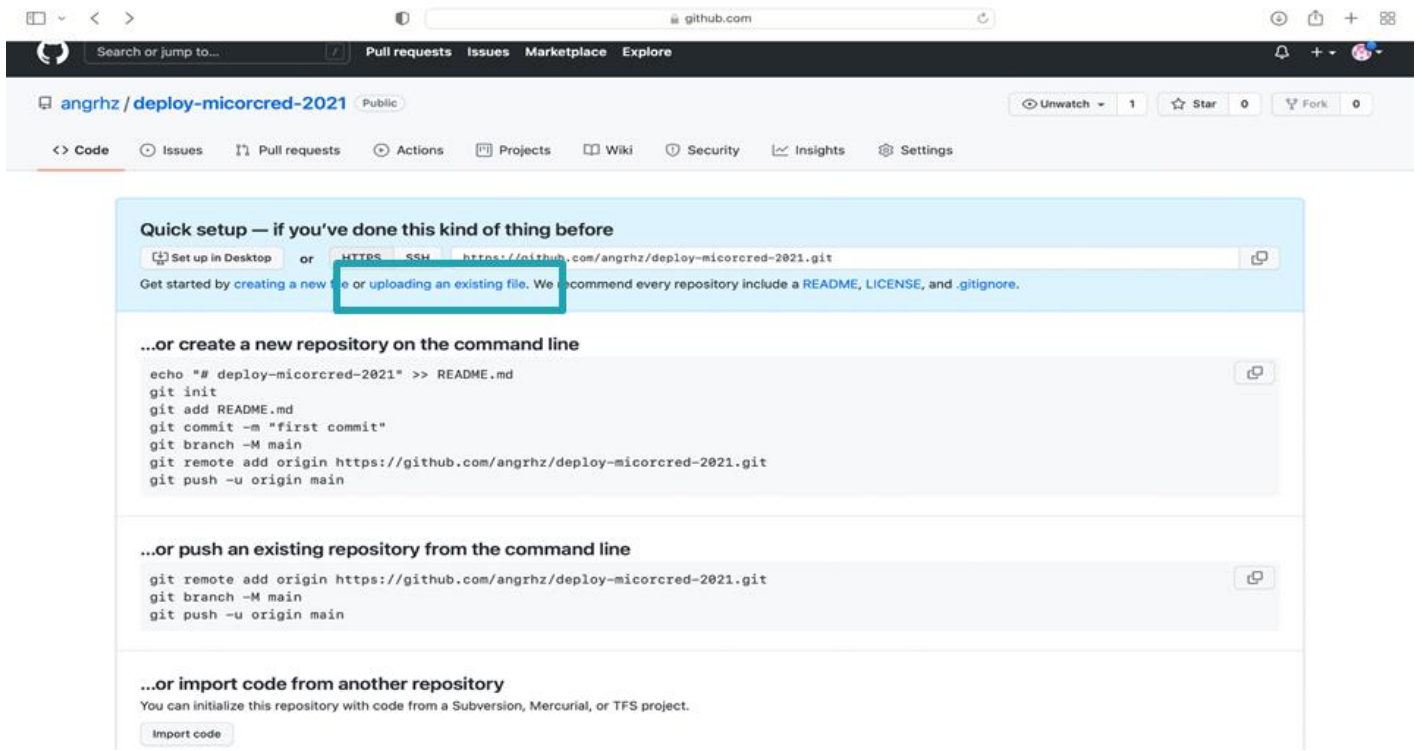
Deployment Model: Langkah-Langkah Model Deployment

3. Membuat repository baru



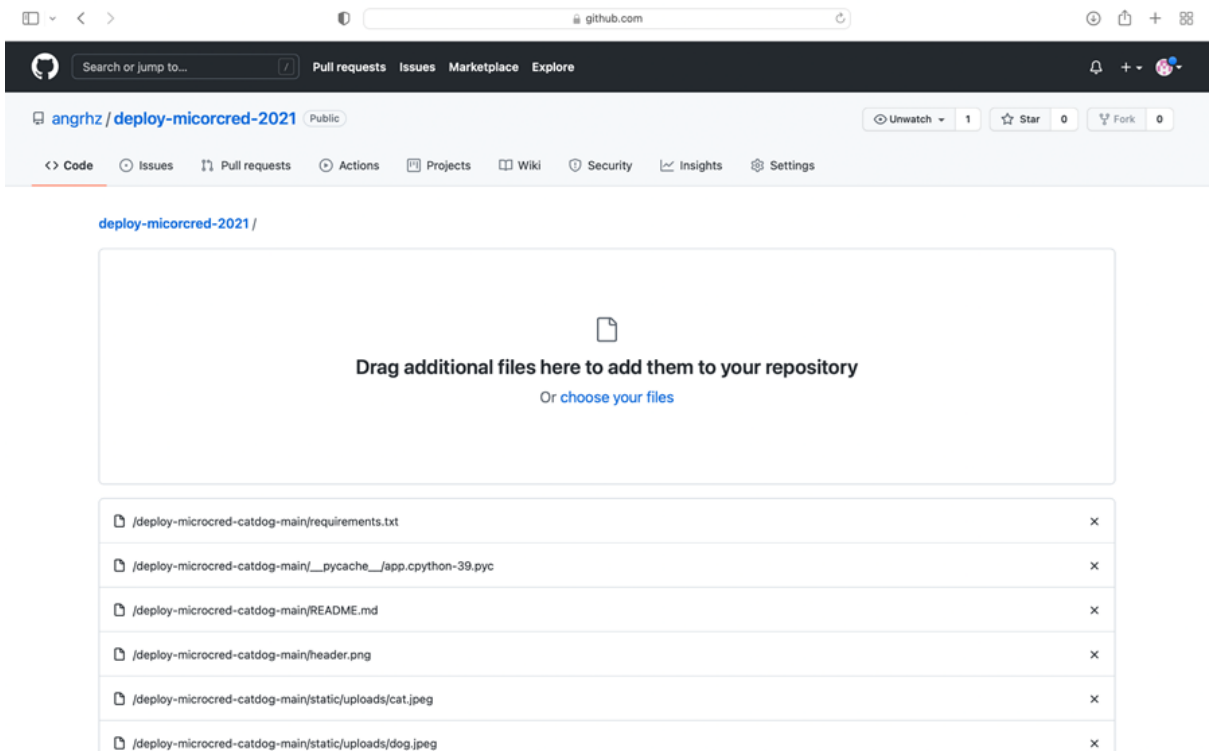
Deployment Model: Langkah-Langkah Model Deployment

6. Pilih uploading an existring file



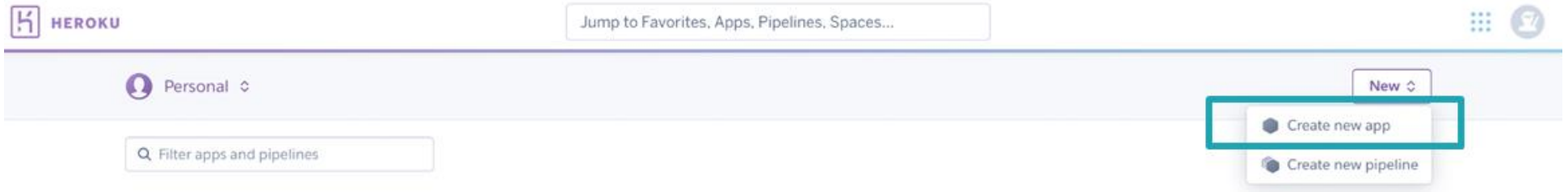
Deployment Model: Langkah-Langkah Model Deployment

7. Upload file repository yang telah di download tadi



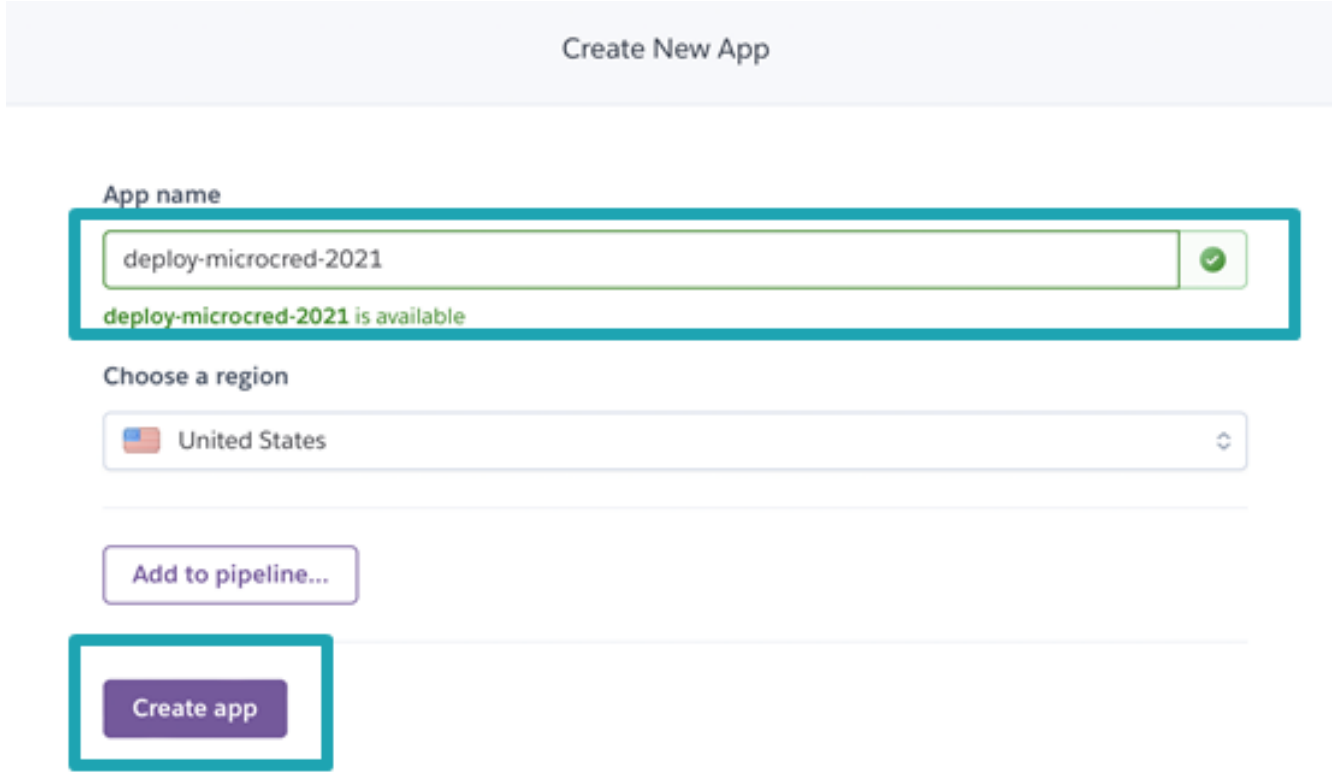
Deployment Model: Langkah-Langkah Model Deployment

9. Klik **New** dan pilih **Create new app**



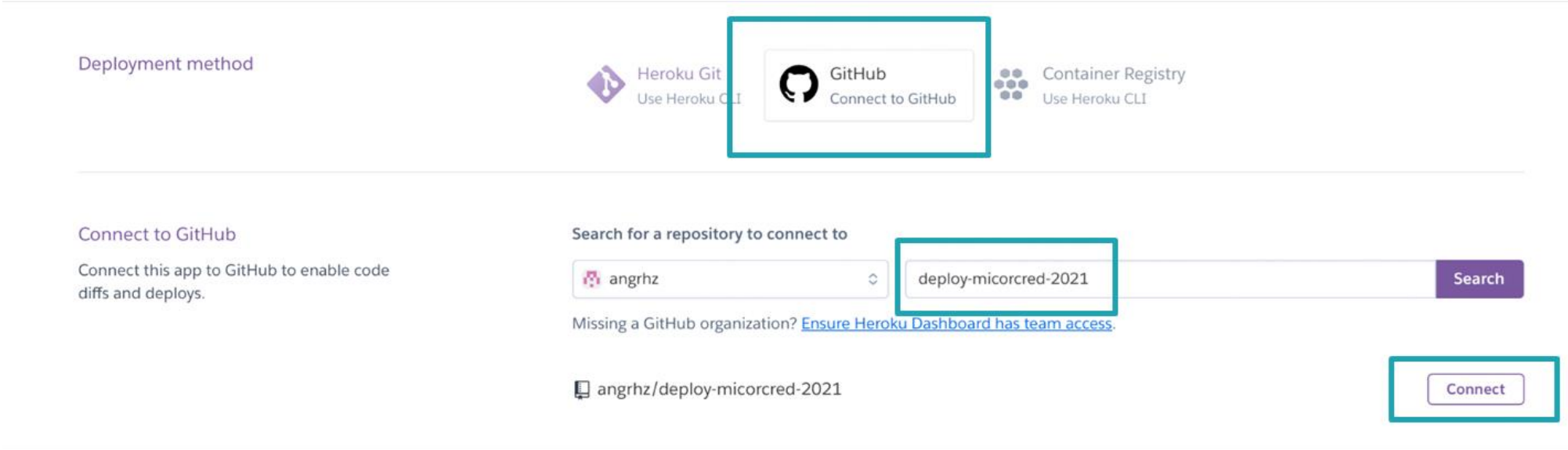
Deployment Model: Langkah-Langkah Model Deployment

10. Masukkan nama aplikasi. Kemudian create aplikasi



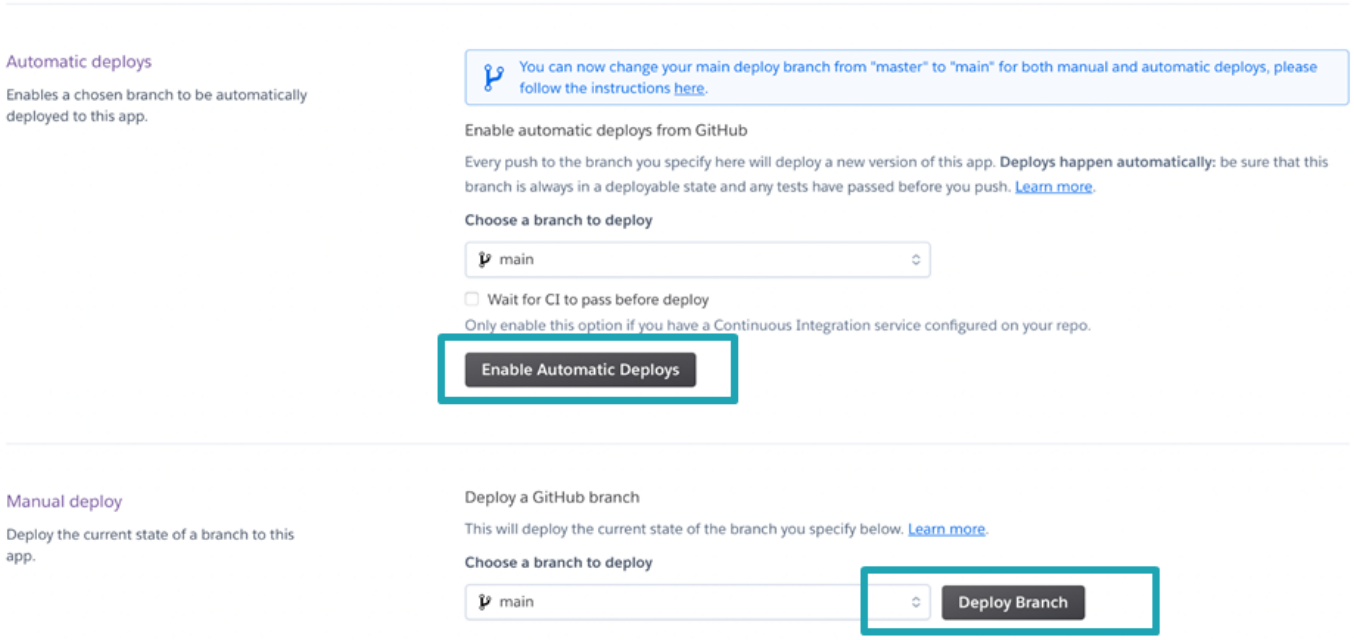
Deployment Model: Langkah-Langkah Model Deployment

11. Pada **Deployment method** pilih GitHub. Kemudian masukkan nama repository dan pilih connect.



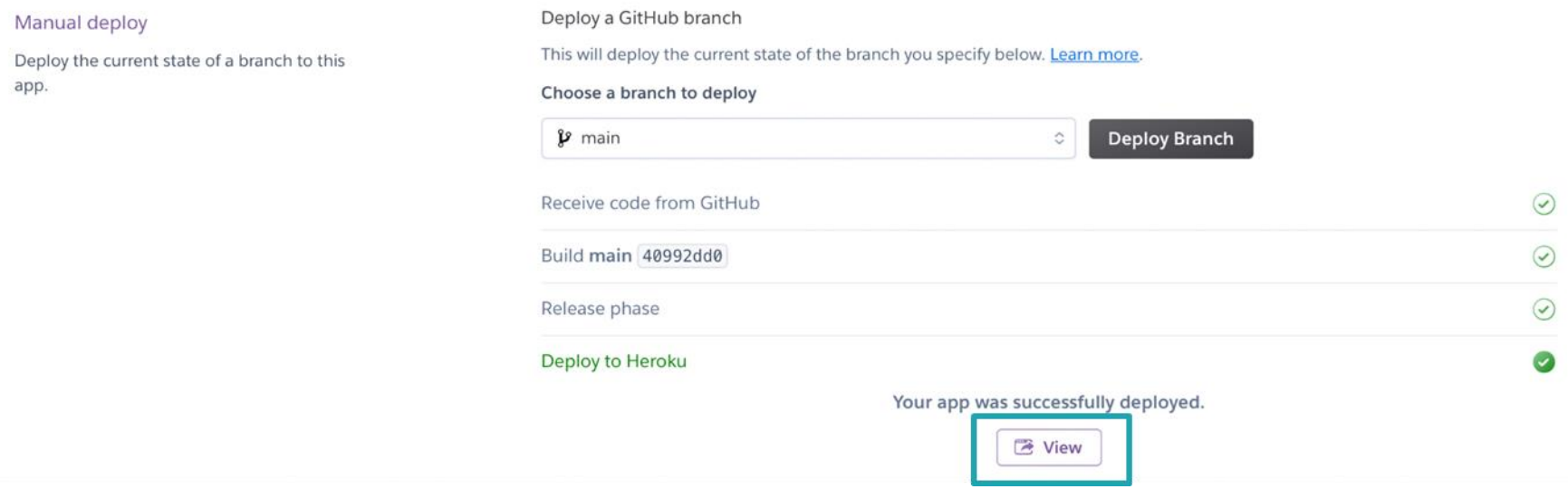
Deployment Model: Langkah-Langkah Model Deployment

12. Pilih **Enable Automatic Deploys** dan lakukan deploy dengan memilih **Deploy Branch**



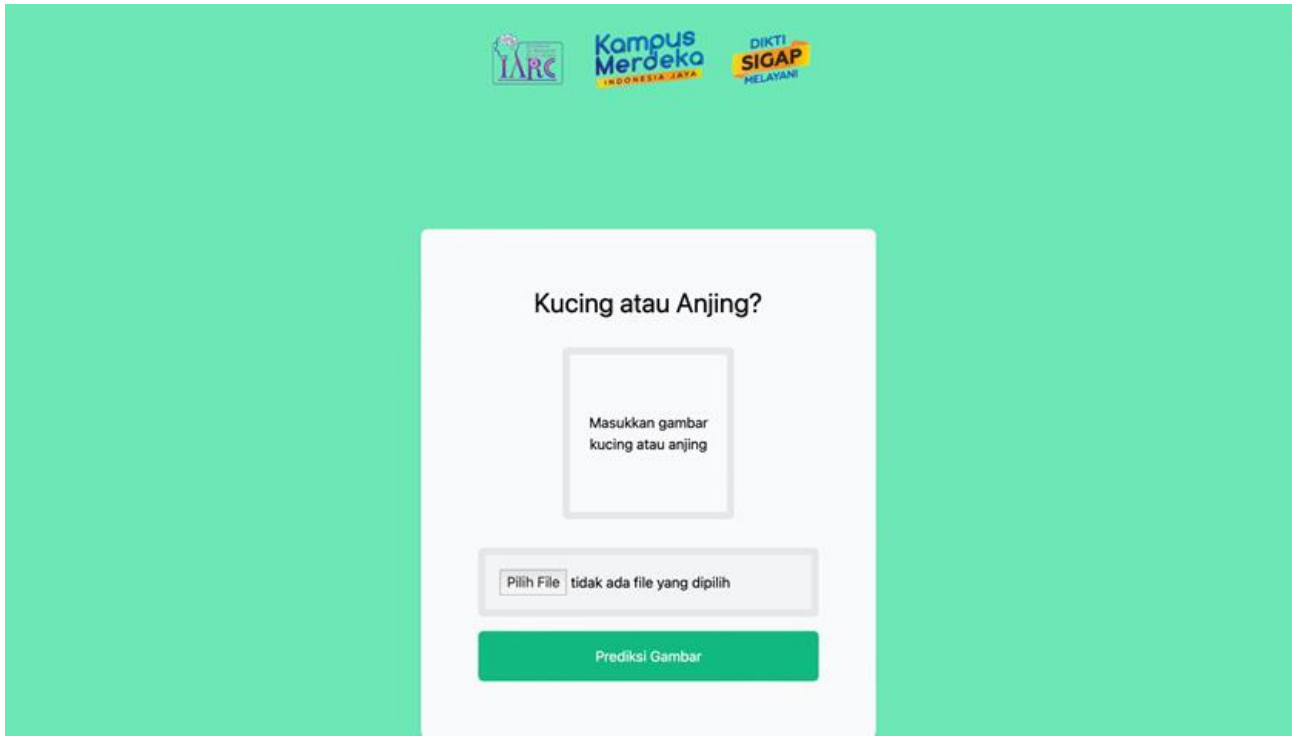
Deployment Model: Langkah-Langkah Model Deployment

13. Tunggu setelah proses selesai. Jika sudah pilih view



Deployment Model: Langkah-Langkah Model Deployment

14. Berikut hasilnya:



Six Strategies for Deploying to Heroku

- There are many ways of deploying your applications to Heroku—so many, in fact, that we would like to offer some advice on which to choose. Each strategy provides different benefits based on your current deployment process, team size, and app. Choosing an optimal strategy can lead to faster deployments, increased automation, and improved developer productivity.
- The question is: How do you know which method is the "best" method for your team? In this post, we'll present six of the most common ways to deploy apps to Heroku and how they fit into your deployment strategy. These strategies are not mutually exclusive, and you can combine several to create the best workflow for your team. Reading this post will help you understand the different options available and how they can be implemented effectively.

Deploying to Production with Git

- Our first method is not only the most common, but also the simplest: pushing code from a Git repository to a Heroku app. You simply add your Heroku app as a remote to an existing Git repository, then use git push to send your code to Heroku. Heroku then automatically builds your application and creates a new release.
- Because this method requires a developer with full access to manually push code to production, it's better suited for pre-production deployments or for projects with small, trusted teams.

  Pros:
  - Simple to add to any Git-based workflow
  - Supports Git submodules

  Cons:
  - Requires access to both the Git repository and Heroku app

GitHub Integration

- If your repository is hosted on GitHub, you can use GitHub integration to deploy changes directly to Heroku. After linking your repository to a Heroku app, changes that are pushed to your repository are automatically deployed to the app. You can configure automatic deployments for a specific branch, or manually trigger deployments from GitHub. If you use continuous integration (CI), you can even prevent deployments to Heroku until your tests pass.
- GitHub integration is also useful for automating pipelines. For example, when a change is merged into the master branch, you might deploy to a staging environment for testing. Once the change has been validated, you can then promote the app to production.

  Pros:
  - Automatically deploys apps and keeps them up-to-date
  - Integrates with pipelines and review apps to create a continuous delivery workflow
  - If you use a CI service (such as Heroku CI) to build/test your changes, Heroku can prevent deployment when the result is fail

  Cons:
  - Requires administrator access to the repository, so it’s only useful for repositories you own
  - Does not support Git submodules

Heroku Review Apps

- When introducing a change, chances are you want to test it before deploying it straight to production. Review Apps let you deploy any GitHub pull request (PR) as an isolated, disposable instance. You can demo, test, and validate the PR without having to create a new app or overwrite your production app. Closing the PR destroys the review app, making it a seamless addition to your existing workflows.

  Pros:

  - Can automatically create and update apps for each PR
  - Supports Docker images
  - Supports Heroku Private Spaces for testing changes in an isolated environment

  Cons:

  - Requires both pipelines and GitHub integration to be enabled

Deploying with Docker

- Docker lets you bundle your apps into self-contained environments, ensuring that they behave exactly the same both in development and in production. This also gives you more control over the languages, frameworks, and libraries used to run your app. To deploy a container to Heroku, you can either push an image to the Heroku container registry, or build the image automatically by declaring it in your app's heroku.yml file.

  Pros:

  - Automatically generate images, or push an existing image to the container registry
  - Consistency between development and production
  - Compatible with Heroku Review Apps

  Cons:

  - If your app doesn’t already run in Docker, you’ll need to build an image
  - Requires you to maintain your own stack
  - Does not support pipeline promotions

Using Hashicorp Terraform

- Infrastructure-as-code tools like Hashicorp Terraform can be helpful to manage complex infrastructure. Terraform can also be used to deploy a Heroku app. Despite it not being officially supported by Heroku, Terraform is being used by many Heroku users. Using Terraform with Heroku, you can define your Heroku apps with a declarative configuration language called HCL. Terraform automates the process of deploying and managing Heroku apps while also making it easy to coordinate Heroku with your existing infrastructure. Plus, Terraform v0.12 now allows you to store Remote State in a PostgreSQL database. This means you can now run Terraform on a Heroku dyno storing Terraform state in a Heroku Postgres database.
- For an example, check out a reference architecture using Terraform and Kafka.

Pros:

- Automates Heroku app deployments
- Allows you to deploy Heroku apps as code
- Simplifies the management of large, complex deployments
- Allows you to configure multiple apps, Private Spaces as well as resources from other cloud providers (e.g. AWS, DNSimple, and Cloudflare) to have a repeatable, testable, multi-provider architecture.

Cons:

- Requires learning Terraform and writing configuration if you don't use it already

The 'Deploy to Heroku' Button

- What if deploying your app was as easy as clicking a button? With the 'Deploy to Heroku' button, it is! It’s great for taking an app for a test run with default settings in a single click, or to help train new developers.
- This button acts as a shortcut allowing you to deploy an app to Heroku from a web browser. This is great for apps that you provide to your users or customers, such as open source projects. You can parameterize each button with different settings such as passing custom environment variables to Heroku, using a specific Git branch or providing OAuth keys. The only requirements are that your source code is hosted in a GitHub repository and that you add a valid app.json file to the project's root directory. We’ve even heard of one company that adds a button to the README for each of their internal services. This forces them to keep the deploy process simple and aids new hires getting up to speed with how services are deployed.

The 'Deploy to Heroku' Button

Deploy
A 'Deploy to Heroku' button.

Pros:

- Easy to add to a project's README file or web page
- Easy to use: simply click the button to deploy the app
- Provides a template with preconfigured default values, environment variables, and parameters

Cons:

- Does not support Git submodules
- Apps deployed via button do not auto-update when new commits are added to the GitHub repo from which it was deployed
- Not a good workflow for apps that you need to keep up to date because buttons can only create new apps and the deployed app is not automatically connected to the GitHub repo from which it came

Metode yang mana?

- The method you choose depends on your specific deployment process, your requirements, and your apps. For small teams who are just getting started, deploying with Git is likely to be your first deployment due to its simplicity. The Heroku Button is equally straightforward, letting you deploy entire apps with a single click. If you use continuous integration or release frequently, integrating with GitHub can simplify this process even more by doing automated deployments when you commit your code. This is a big improvement over deploying on an IaaS system because Heroku manages the entire process automatically.
- As your requirements get more sophisticated, add the other strategies as needed. When your application is running in a production environment and you need quality control, you may want to add pipelines to get the advantages of review apps, automated testing, and staging environments. If you need a custom stack, then you can do so with Docker. As you add more complex infrastructure components, then add Terraform.
- Advanced teams will use a combination of strategies: For example, you may choose to deploy a Docker image by creating a review app from a GitHub pull request, testing the review app, then manually deploying the final version using git push.

EXPLORE WHAT'S FREE ON HEROKU

Many developers are looking for a free cloud web hosting services to run their apps, blogs, or bots without the hassle of managing servers. Others are curious about trying the platform-as-a-service model, but not quite ready to invest. Here's an overview of free cloud services for developers available through the Heroku platform and ecosystem.

Provision resources to run your app

Heroku's free cloud services begins with the apps - apps which can be deployed to dynos - our lightweight Linux containers that are at the heart of the Heroku platform. When you sign up with Heroku, you automatically get a pool of free dyno hours to use for your apps. When your app runs, it consumes dyno hours. When it idles (automatically, after 30 minutes of inactivity), or when you scale it down, your app will stop consuming dyno hours.

You can deploy your free app as many times as you need to, (we love continuous deployment practices), and as long as you have dyno hours, your app will be live and publicly accessible.

Maximize your free platform services

Get 1000 free dyno hours by verifying your Heroku account with a credit card; unverified accounts receive 550 free hours. You will not be charged, unless you decide to use a paid service. Account verification provides other benefits too, including running more than 5 free apps, as well as free custom domain names.

APPS FREE RESOURCES

- Run apps for free using your monthly pool of free dyno hours
- Unverified accounts: receive a pool of 550 free dyno hours
- Verified accounts: receive an additional 450 free dyno hours
- Dyno hours can be shared across any of your free apps
- 1 web dyno/1 worker dyno/1 one-off dyno maximum per app
- 512 MB RAM per dyno
- Free apps sleep automatically after 30 mins of inactivity to conserve your dyno hours
- Free apps wake automatically when a web request is received
- Access to the Heroku Dashboard and Heroku CLI for app management
- Custom domains for every free app (with verified account)
- Up to 5 free apps (unverified) or 100 (verified)

Tools Lab Online

- Anaconda
- Visual Studio Code
- Heroku
- https://code.visualstudio.com/download
- https://www.heroku.com
- https://www.anaconda.com/products/individual
- https://github.com/alfanme/dts-deployment-linreg
- https://github.com/alfanme/dts-deployment-ann

Quiz / Tugas

Silakan kerjakan tugas mandiri yang telah disiapkan untuk pertemuan ini. Tugas mandiri akan dijelaskan lebih detail oleh Instruktur dan asisten Anda.

Terima kasih