



Direktorat Jenderal Pendidikan Tinggi, Riset, dan, Teknologi  
Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi  
Republik Indonesia

DIKTI  
**SIGAP**  
MELAYANI

Kampus  
**Merdeka**  
INDONESIA JAYA



## MICROCREDENTIAL: ASSOCIATE DATA SCIENTIST

01 November – 10 Desember 2021

---

Pertemuan ke-17

# Melakukan Deployment Model (pipelining, teknologi deployment, cara deployment)



[ditjen.dikti](https://www.facebook.com/ditjen.dikti)



@ditjendikt  
i



[ditjen.dikti](https://www.instagram.com/ditjen.dikti/)



Ditjen  
Diktiristek



<https://dikti.kemdikbud.go.id/>



# Profil Pengajar: Nama Lengkap dan Gelar Akademik



Poto  
Pengajar

## Contak Pengajar:

Ponsel:

xxxxxx

Email:

xxxxxxx

Jabatan Akademik: .....

Latar Belakang Pendidikan:

- S1: .....
- S2: .....
- S3: .....

Riwayat/Pengalaman Pekerjaan:

- Dosen
- XXXX
- XXXX
- XXXX
- XXXX

# Tanggal Penting

- Tanggal batas pengumpulan proyek final: 13 Desember 2021
- Tanggal Ujian: Tanggal pada Pertemuan
- **Tugas Proyek:**
  - Aplikasi telah dideploy pada layanan yang digunakan (misal: Heroku)
  - Video deskripsi aplikasi tersebut maximum 3 menit.
  - Deskripsi final project dijadikan satu (dengan revisi yang perlu) dan diunggah pada SpadaDikt. Pada SpadaDikt juga diinformasikan URL
    - . Lokasi aplikasi yang dideploy
    - . Lokasi video deskripsi



No	UK	Judul	Associate Data Scientist	Data Scientist	
1	J.62DMI00.001.1	Menentukan Objektif Bisnis	OK	OK	
2	J.62DMI00.002.1	Menentukan Tujuan Teknis Data science		OK	
3	J.62DMI00.003.1	Membuat Rencana Proyek Data Science			
4	J.62DMI00.004.1	Mengumpulkan Data	OK		
5	J.62DMI00.005.1	Menelaah Data	OK	OK	
6	J.62DMI00.006.1	Memvalidasi Data	OK	OK	
7	J.62DMI00.007.1	Menentukan Objek Data	OK	OK	
8	J.62DMI00.008.1	Membersihkan Data	OK	OK	
9	J.62DMI00.009.1	Mengkonstruksi Data	OK	OK	
10	J.62DMI00.010.1	Menentukan Label Data	OK		<b>Ujian akhir</b>
11	J.62DMI00.011.1	Mengintegrasikan Data			<b>Mock up sertifikasi</b>
12	J.62DMI00.012.1	Membangun Skenario Model		OK	
13	J.62DMI00.013.1	Membangun Model	OK	OK	
14	J.62DMI00.014.1	Mengevaluasi Hasil Pemodelan	OK	OK	
15	J.62DMI00.015.1	Melakukan Proses Review Pemodelan		OK	
16	J.62DMI00.016.1	Membuat Rencana Pengembangan Model			

# 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

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

## Course subtopic

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



# Membuat Rencana Deployment Model

KODE UNIT : J.62DM100.016.1

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 <i>deployment</i>	<p>1.1 <b>Existing system</b> terkait yang sudah dimiliki organisasi diidentifikasi.</p> <p>1.2 Parameter keberhasilan <i>deployment</i> disusun berdasarkan standar yang berlaku.</p> <p>1.3 Diskripsi/perbedaan <b>konfigurasi</b> antara <b>lingkungan pengembangan</b> dengan <b>lingkungan deployment</b> dianalisis sesuai standar yang berlaku.</p> <p>1.4 <b>Strategi deployment</b> ditentukan berdasarkan <b>parameter keberhasilan</b> dan hasil analisis diskripsi/perbedaan.</p> <p>1.5 Tools untuk <i>deployment</i> diidentifikasi.</p>
2. Menyusun instruksi <i>deployment</i>	<p>2.1 <b>Instruksi deployment</b> dibuat sesuai strategi <i>deployment</i>.</p> <p>2.2 <b>Mekanisme pemantauan deployment</b> disusun sesuai strategi <i>deployment</i>.</p> <p>2.3 Instruksi <i>deployment</i> didokumentasikan sesuai standar yang berlaku.</p>

## 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 dikhawasukan untuk pembuatan aplikasi atau sistem berbasis IT.
- 1.4 Lingkungan *deployment* adalah sebuah lokasi sistem berbasis IT yang dikhawasukan untuk penerapan 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 teratur dan terkuantifikasi yang mendefinisikan hal-hal yang perlu dipantau dari sistem.



# Melakukan Deployment Model

KODE UNIT : J.62DM100.017.1

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 <i>deployment</i> sesuai instruksi	1.1 Prasyarat <b>lingkungan deployment</b> disiapkan sesuai strategi <i>deployment</i> . 1.2 <i>Deployment</i> dilaksanakan sesuai <b>instruksi deployment</b> .
2. Membuat laporan hasil <i>deployment</i>	2.1 <b>Keluaran sistem</b> sebagai efek dari <i>deployment</i> didokumentasikan. 2.2 Masalah yang timbul setelah pemodelan karena efek <i>deployment</i> didokumentasikan. 2.3 Hasil proses <i>deployment</i> didokumentasikan sesuai dengan standar yang berlaku.

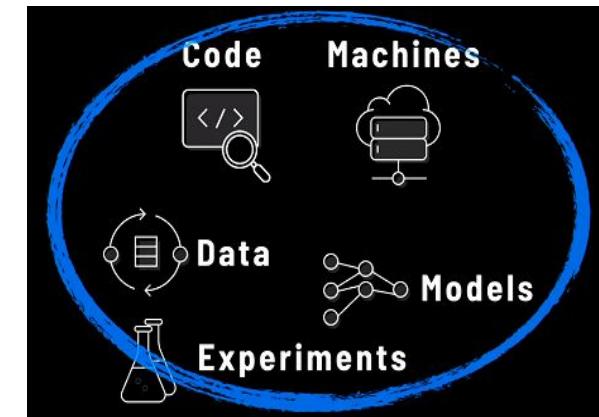
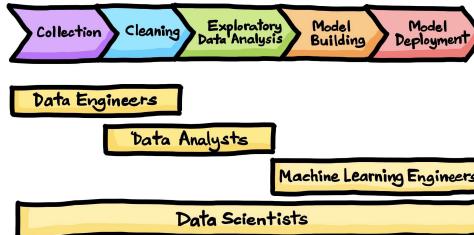
## 1. Konteks variabel

- 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.

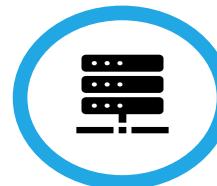


# Pertimbangan Penting

## THE DATA SCIENCE PROCESS



Cases  
& Value



Data  
Ecosystems



Techniques  
& Tools



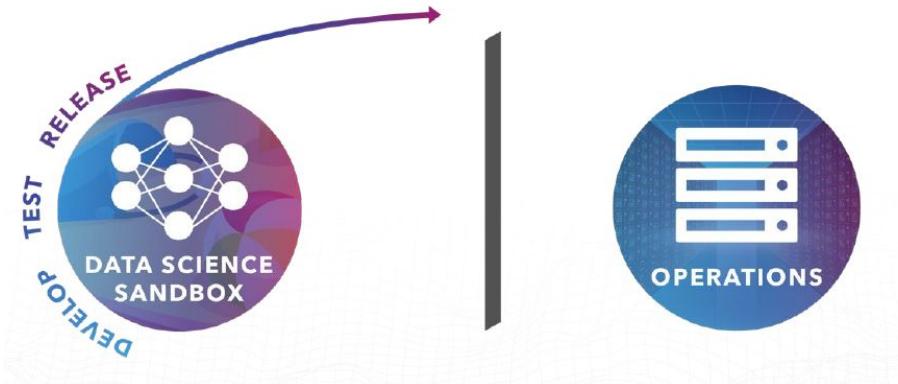
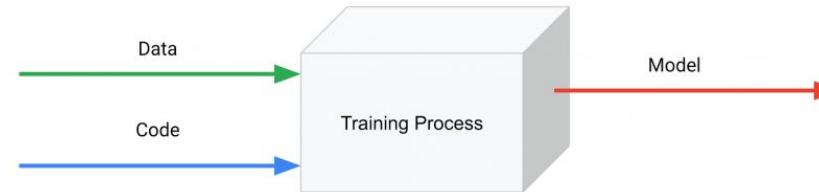
Workflow  
Integration



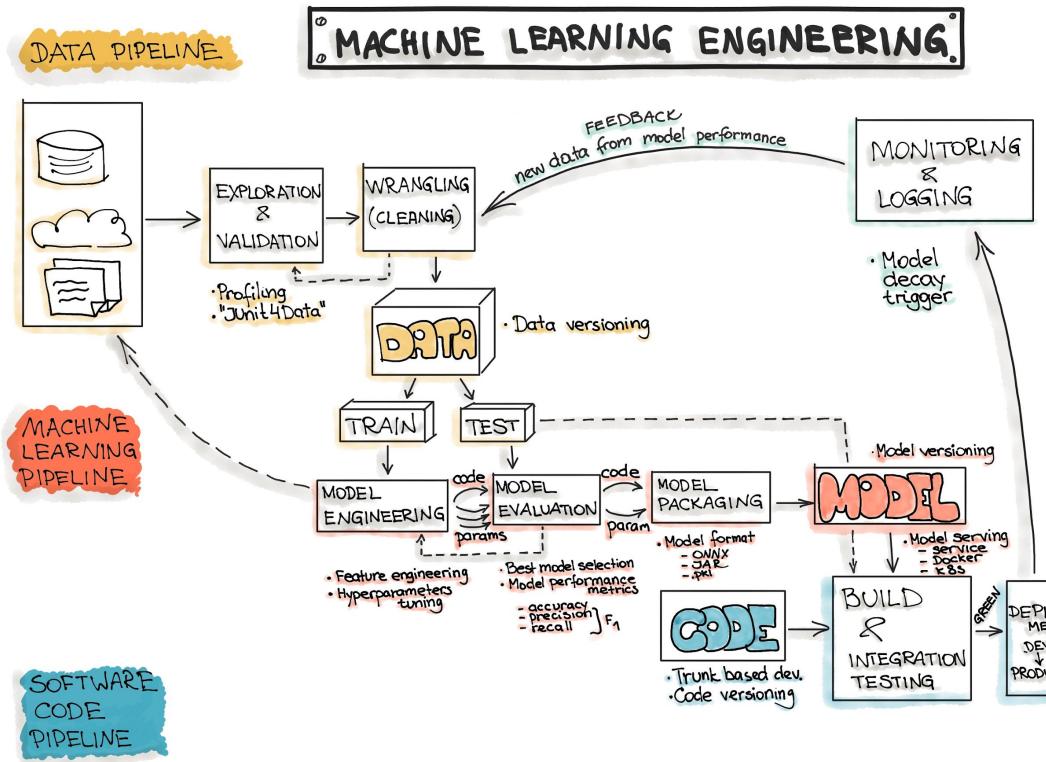
Open  
Culture



# Deployments



## Pendahuluan(1/6)



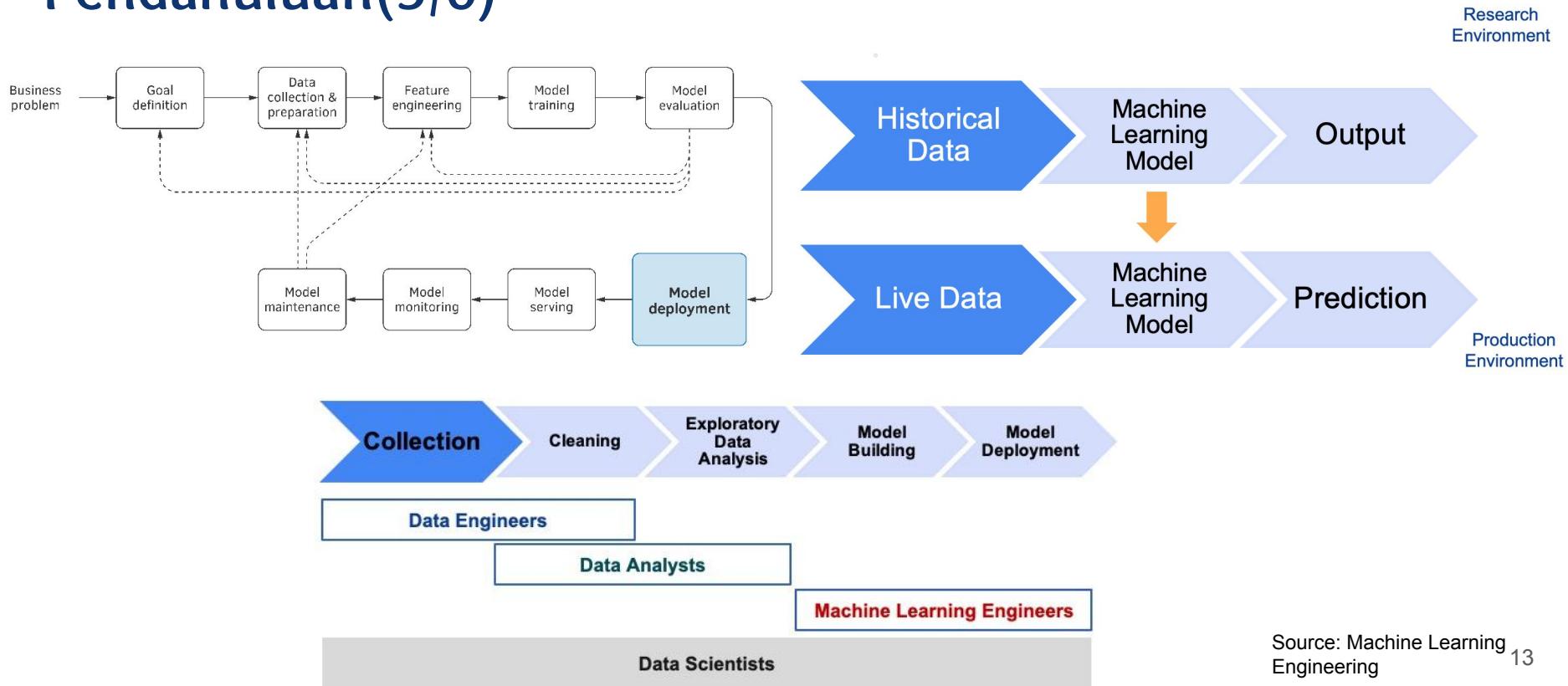
Source: Machine Learning  
Engineering 11

## 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)

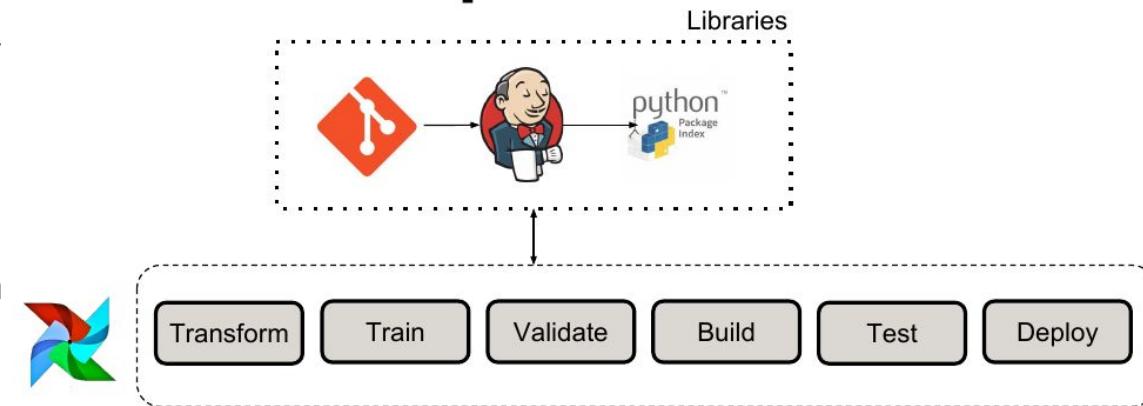
## Tantangan pada implementasi Deployment model:

- 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 penggerjaan proyek dan mengurangi reproduktifitas.



# Tools utk Data Science

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

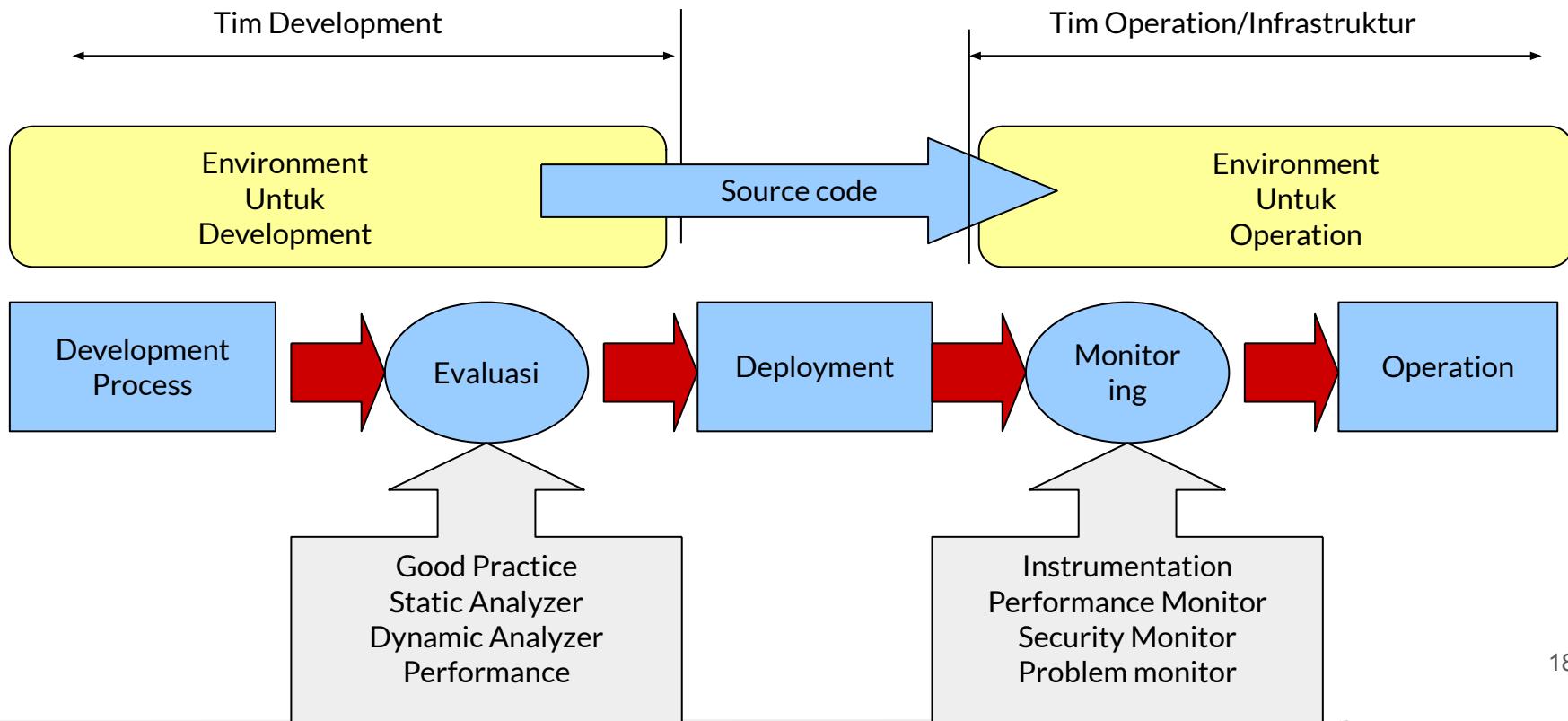


PostgreSQL

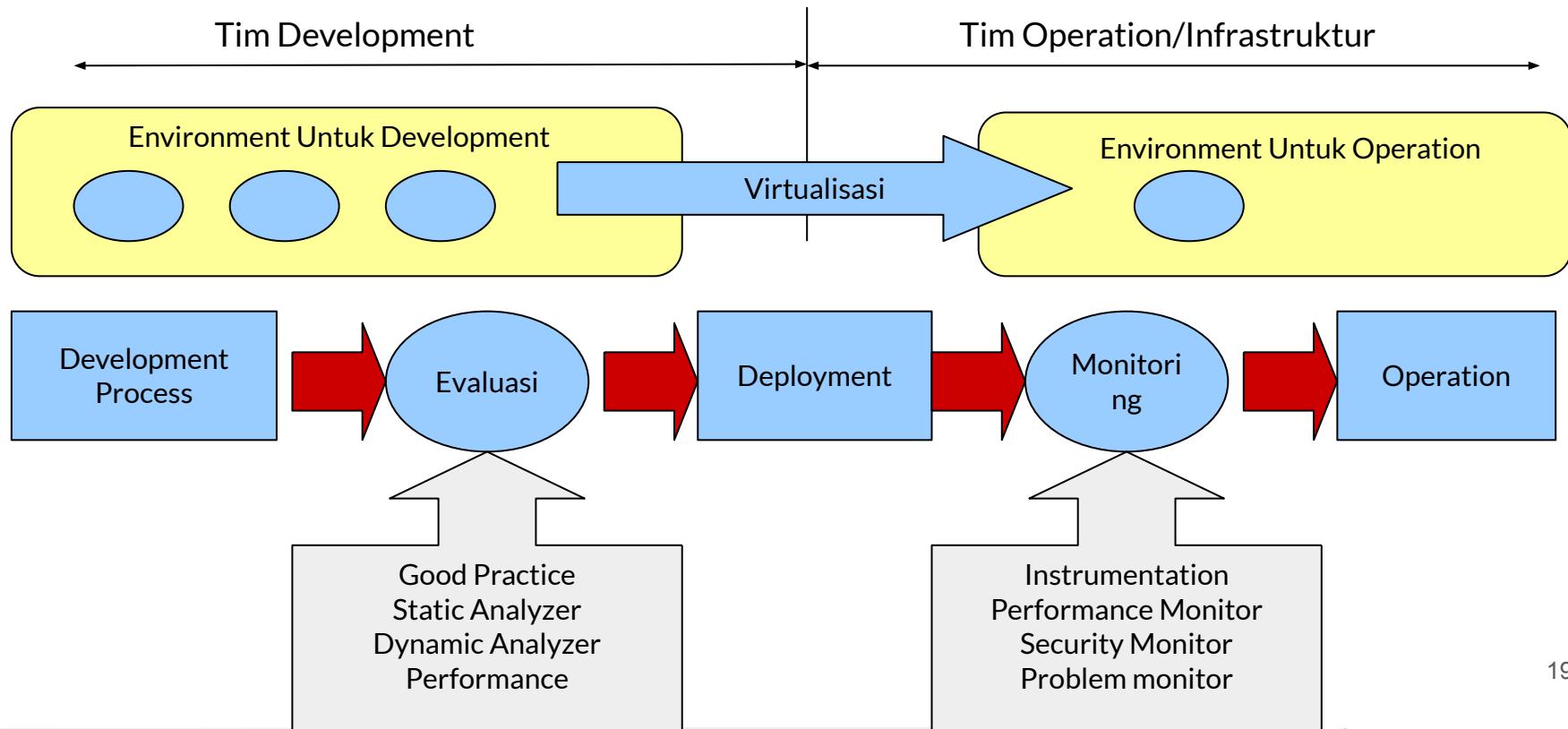




# Development → Deployment → Ops

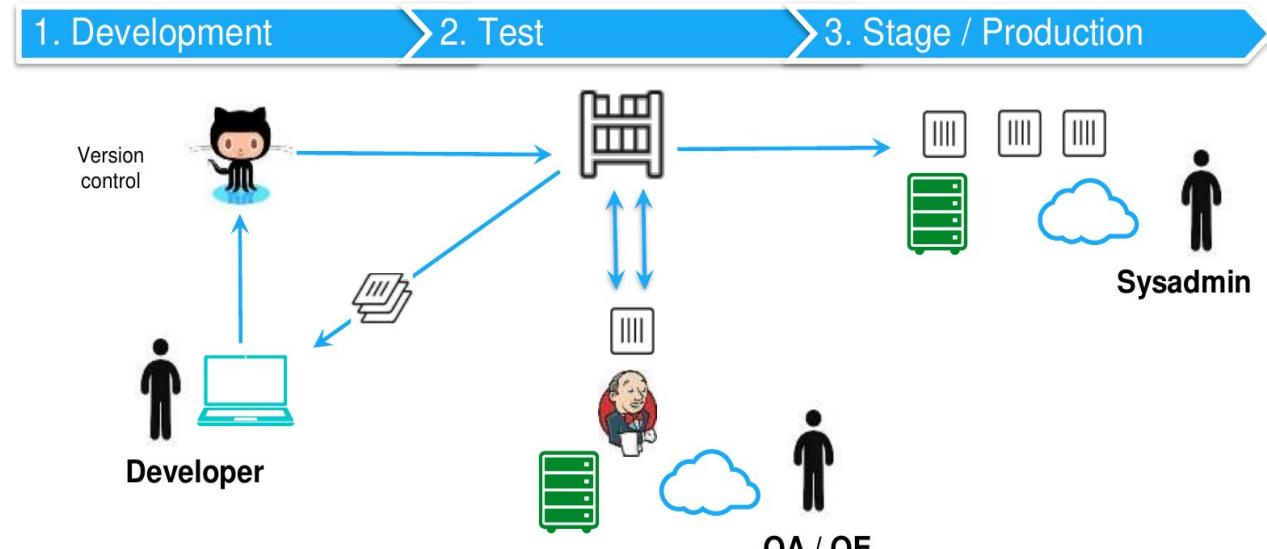


# Continue Development

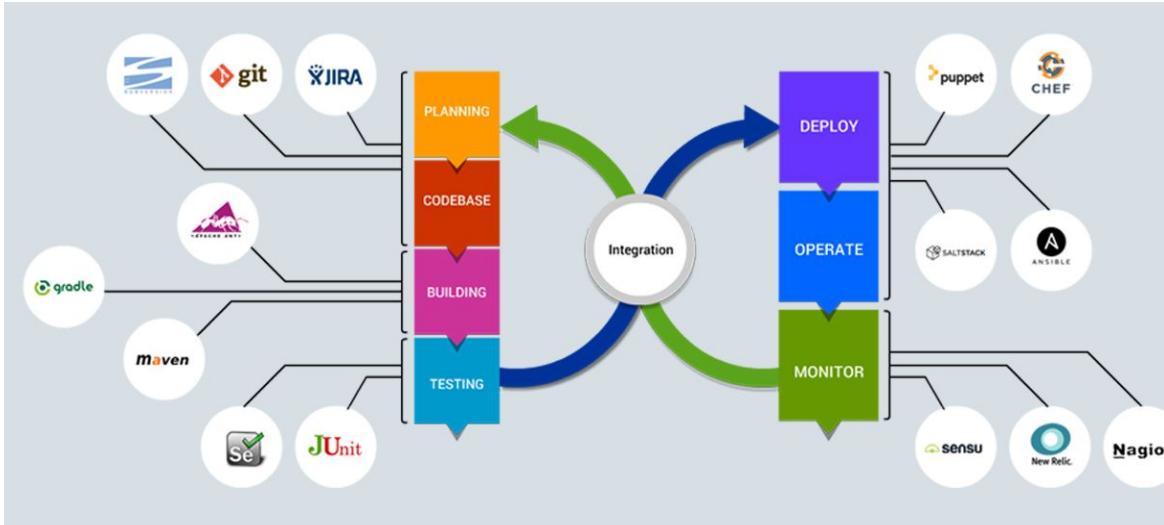


# CI/CD untuk Data Science

- 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



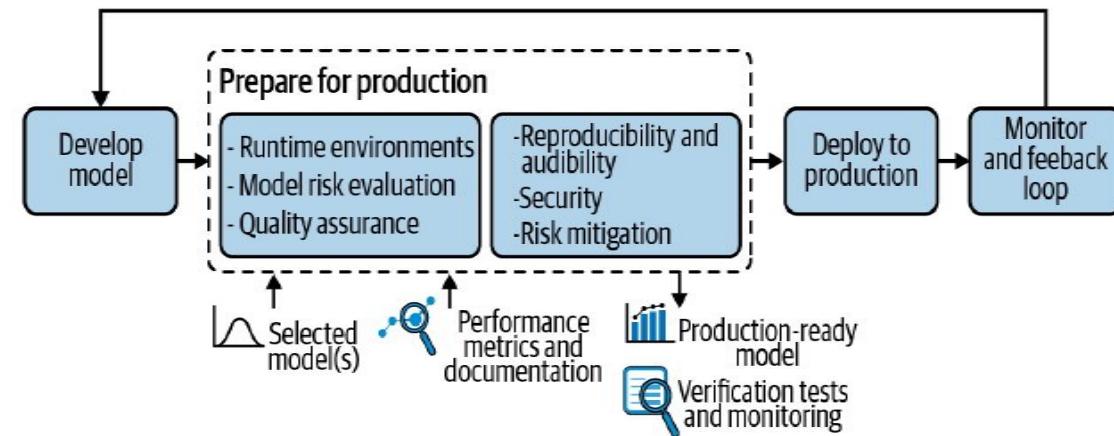
# Tools untuk DevOps



- 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

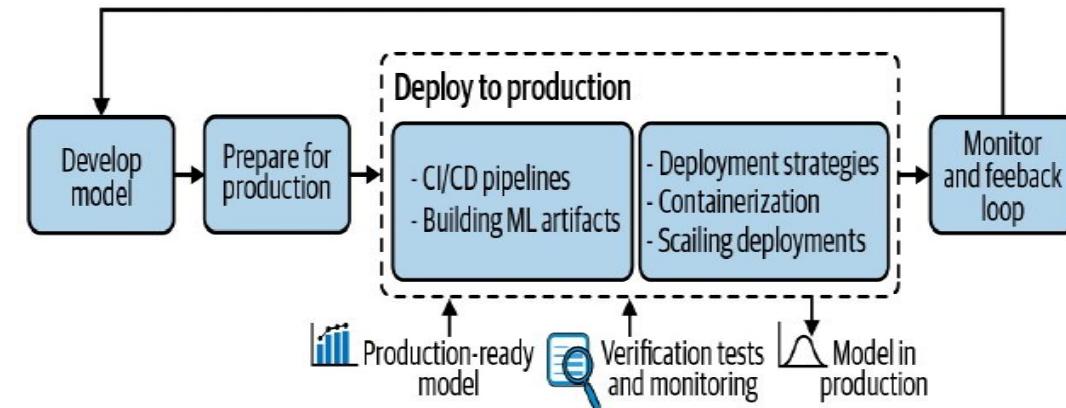
# Preparing for production

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

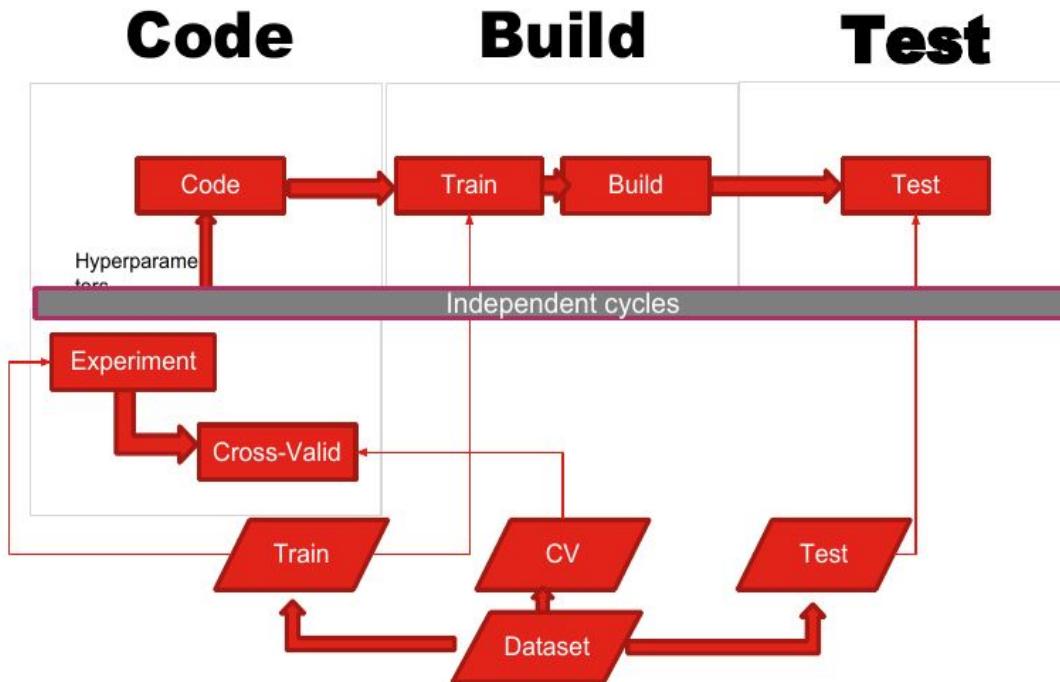


# Deploy to production

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



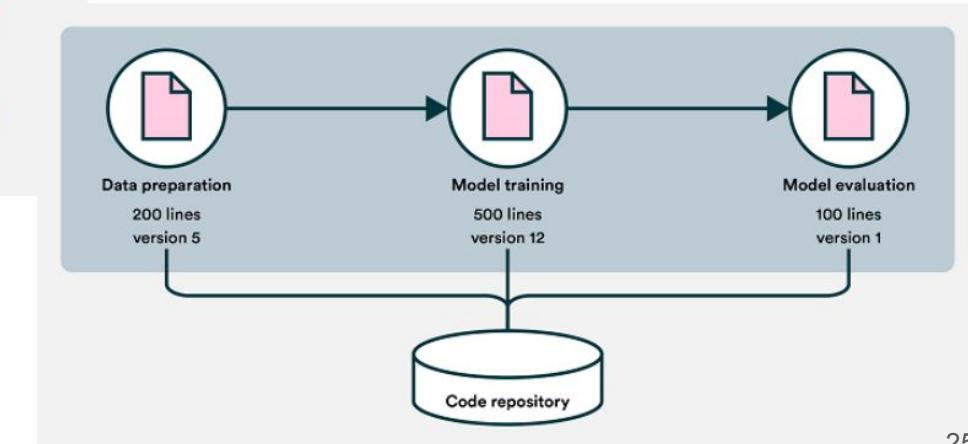
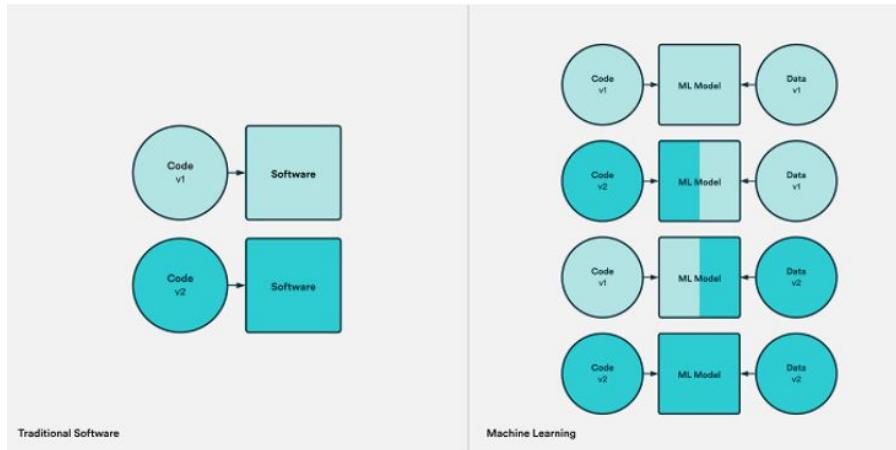
# Siklus pengembangan Solusi



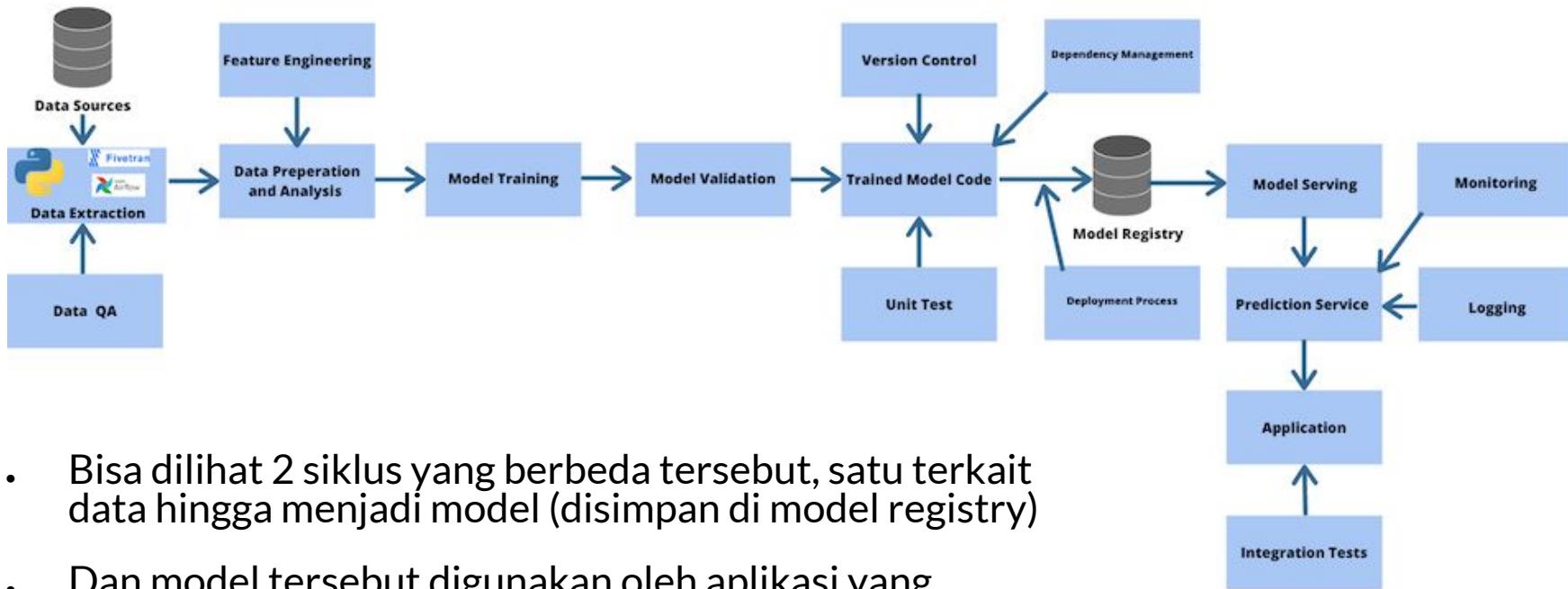
- Yang membedakan deployment solusi Data Science Machine Learning, adalah perlunya memastikan deployment data set dengan baik
- Proses versioning, dan deployment data juga harus dilakukan secara tepat.
- Karena dataset ini akan menentukan bagaimana solusi bekerja dengan baik
- Sehingga ada 2 siklus yang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



# Perbedaan Software ML



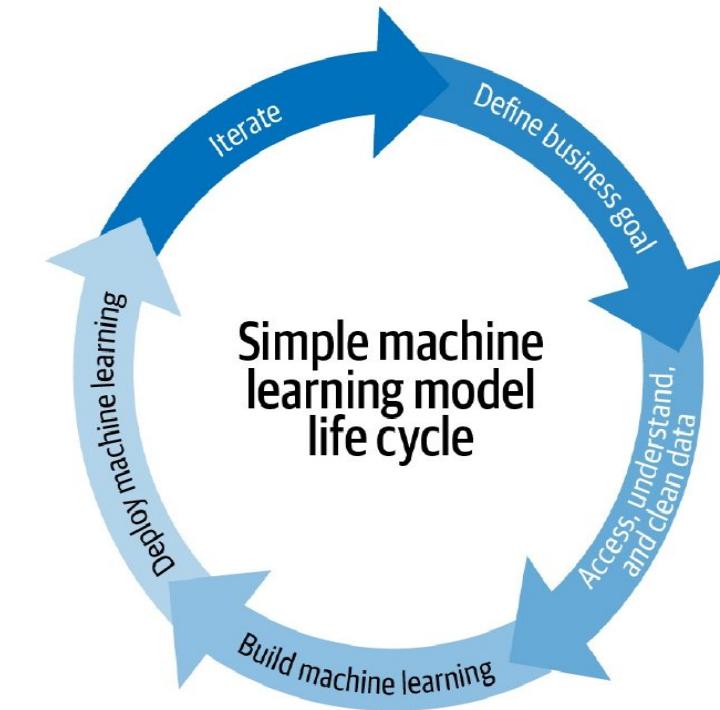
# Pembuatan Artefak



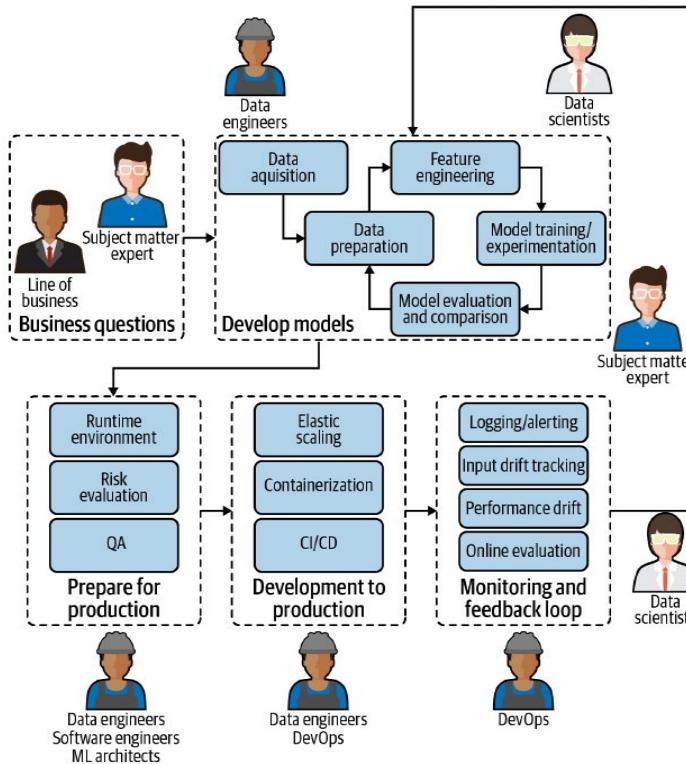
- Bisa dilihat 2 siklus yang berbeda tersebut, satu terkait data hingga menjadi model (disimpan di model registry)
  - Dan model tersebut digunakan oleh aplikasi yang dideploy

# 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



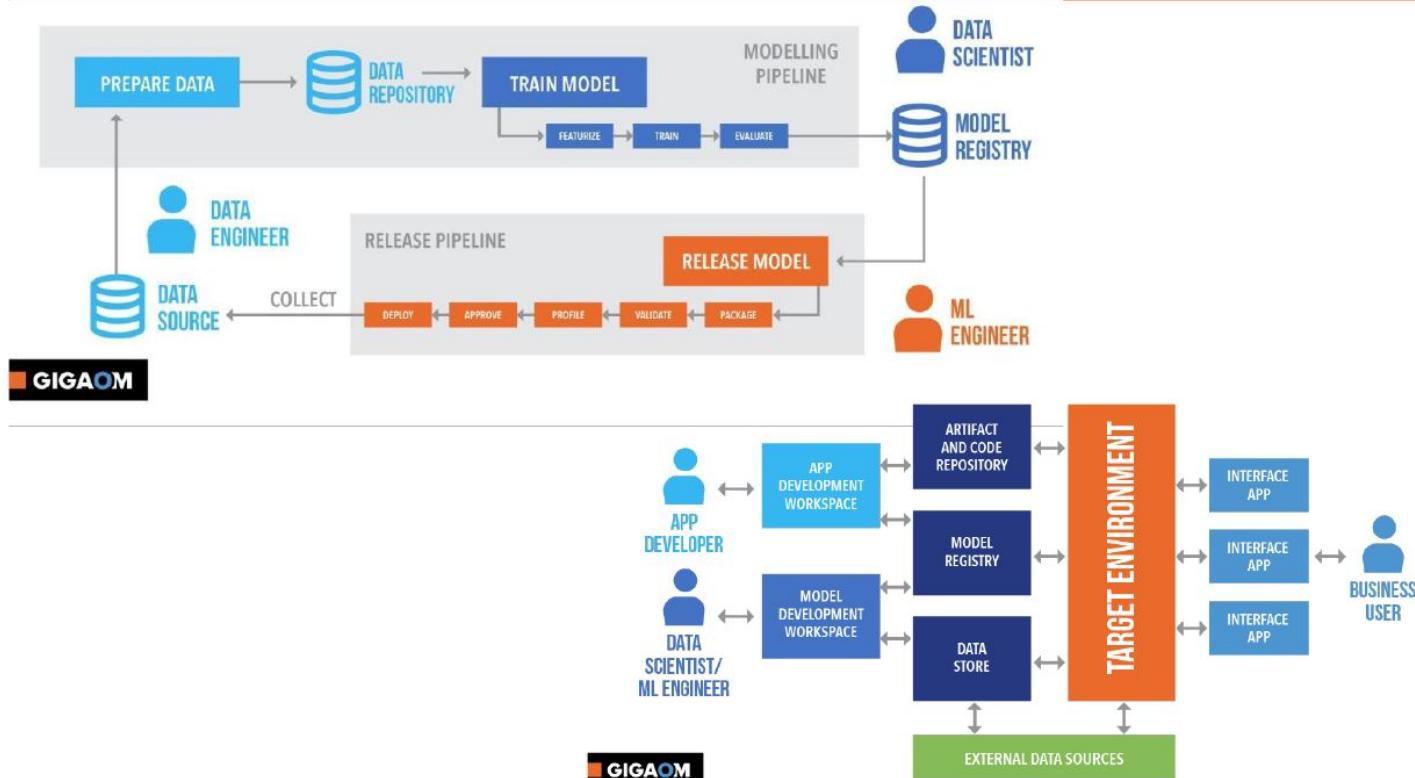
# Personal



- 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 Development & Data Science

MLOps & DevOps

Data Engineering



Connect with stakeholders

Low hanging fruits  
High-impact projects

Set performance indicators  
Set performance targets

Use Design Thinking, 10  
Types of Innovation,  
LEAN, Six Sigma, etc

Keep stakeholders updated

Build prototypes  
Log experiments

Iterate, iterate, iterate

Iteratively improve models

Feature engineering  
Hyperparameters

Track and archive models  
Track and archive data

Make it reproducible

Add models to workflows

Serve through APIs  
Monitor performance

Automate deployment  
Automate rollback

Spark, Docker, Kubernetes

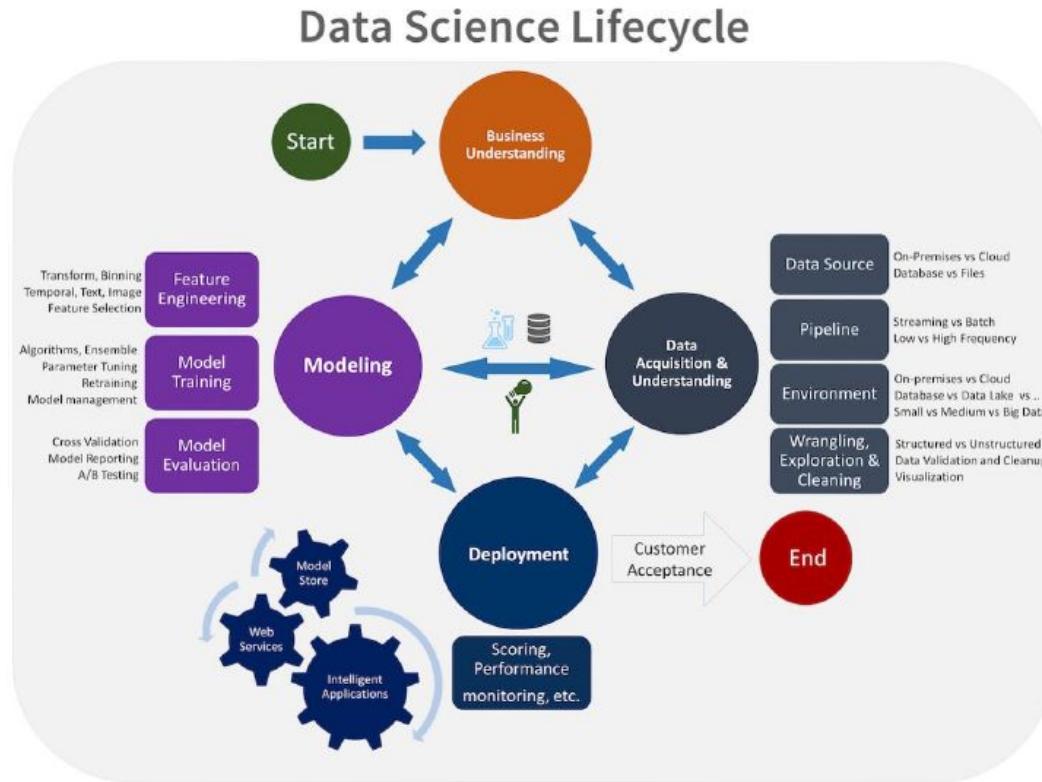
Build robust pipelines

Operate on real-time data  
Archive data

Python, R, Julia

Kafka, MQTT, Avro, Thrift

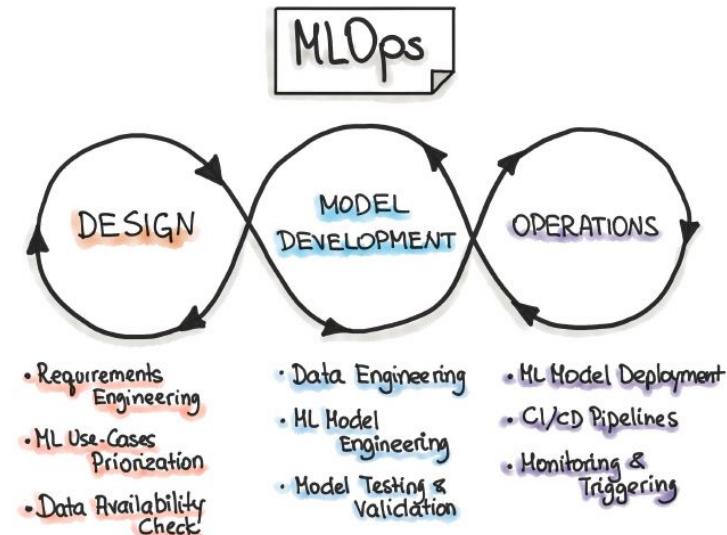
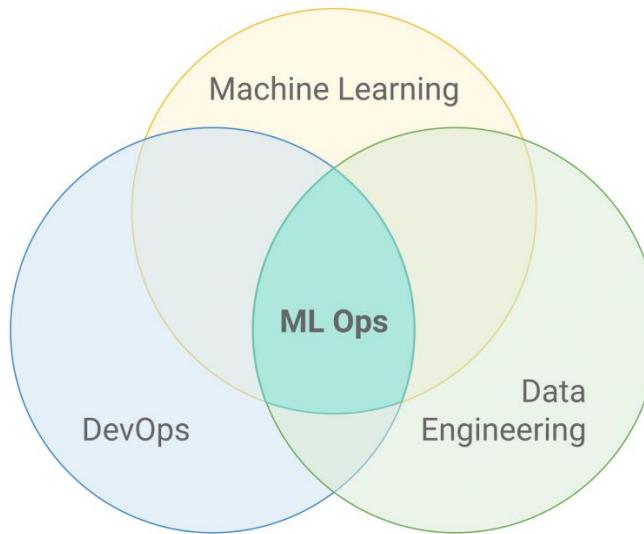
# Deployments





# ML Ops

## Iterative-Incremental Process in MLOps





# 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 disain, 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

$$\text{MLOps} = \text{ML} + \text{DEV} + \text{OPS}$$



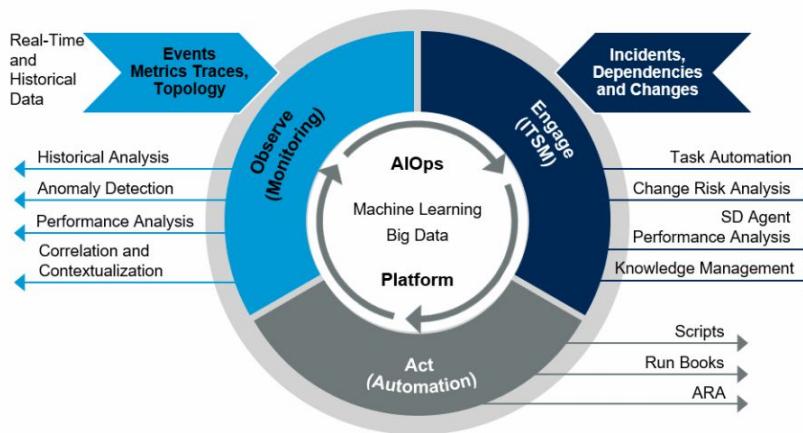
# ML Ops

- MLOps bertujuan untuk menyatukan skilus 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
- MLOPs 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 Metrics. 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

## AIOps Platform Enabling Continuous Insights Across IT Operations Monitoring (ITOM)



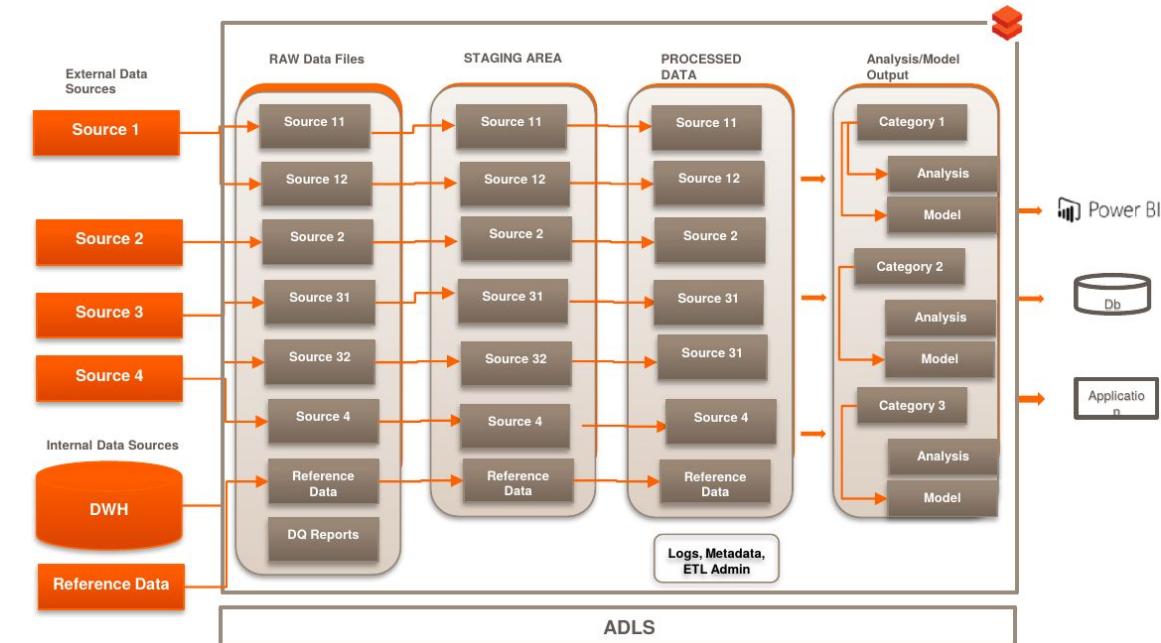
Source: Gartner  
ID: 378587

- 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 ModelOps 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
- 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 pro-aktif
  - Mempelajari secara terus menerus untuk kemampuan penanganan di masa depan.



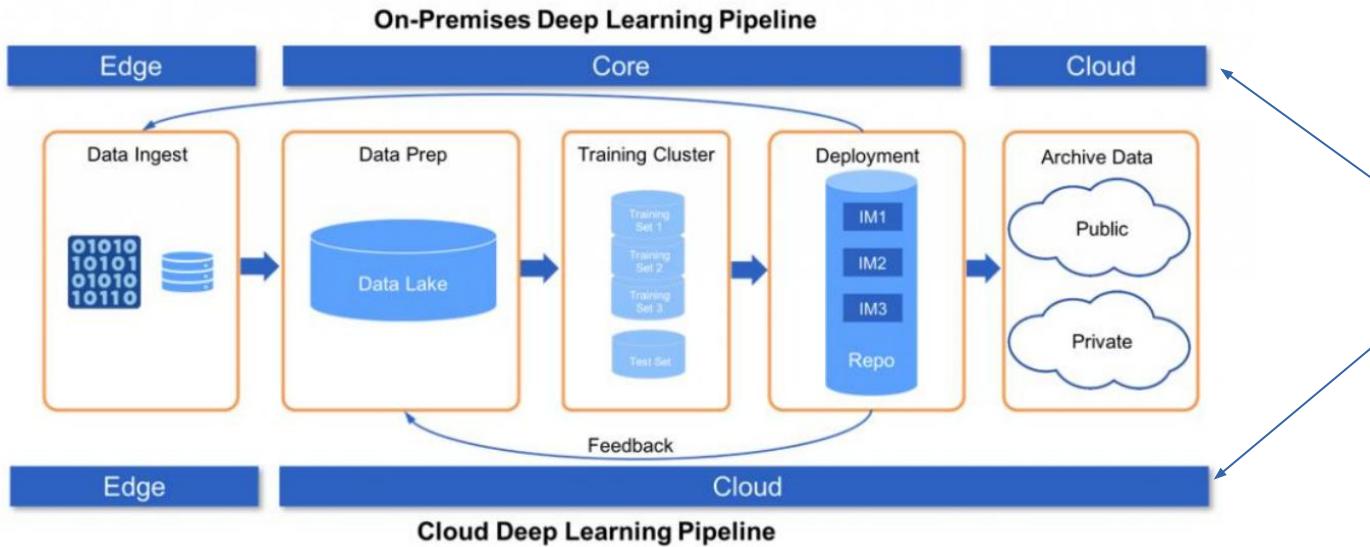
# Pipelining Data

- 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, Datawarehouse, 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 dideploy 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 diupload 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 memperbarui 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)

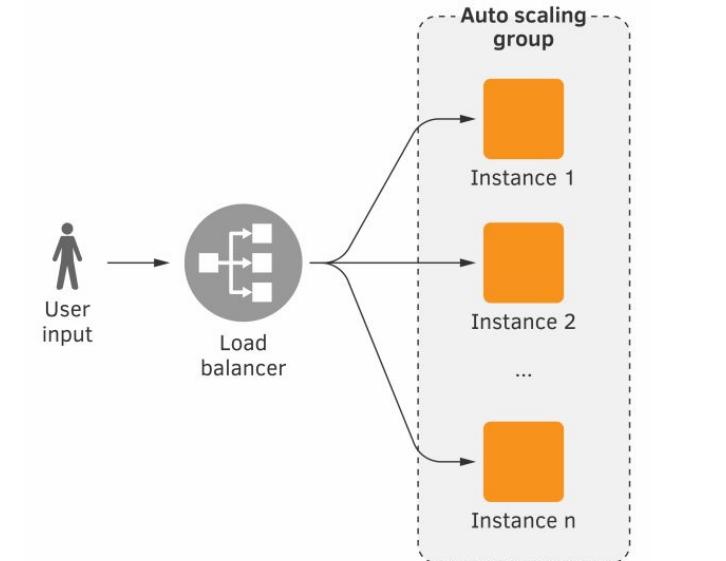
### Dynamic deployment on a server

- 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.



- Setiap instance berisi codes untuk menjalankan ekstraktor fitur dan model
- Layanan web yang memiliki akses code tersebut
- Dalam python, layanan web REST API biasanya diimplementasikan menggunakan Flask atau FastAPI



## Penerapan model berbasis pola: Dynamic Deployment(9/21)

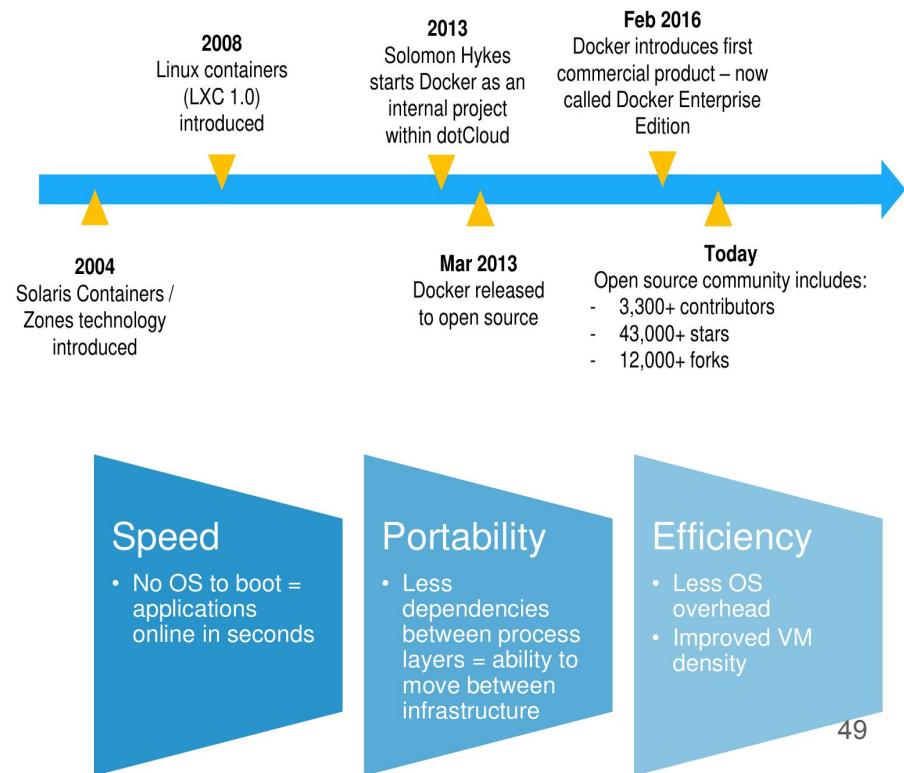
### Deployment pada virtual machine

- 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.

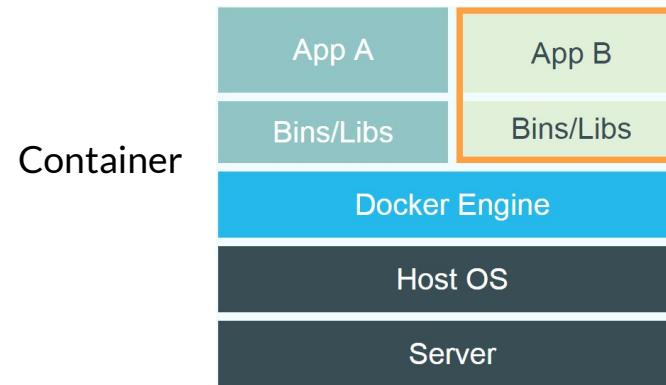
## Penerapan model berbasis pola: Dynamic Deployment(10/21)

### Deployment pada container

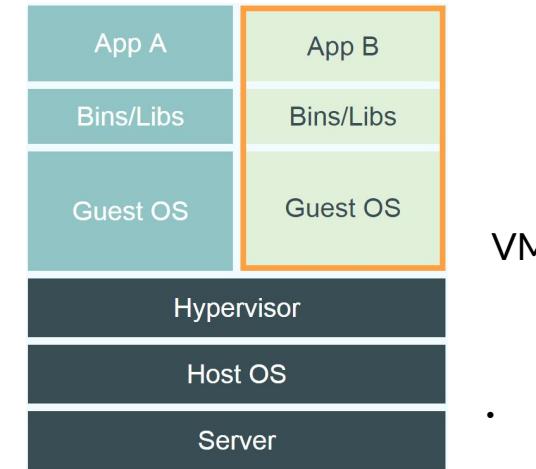
- Container mirip dengan VM
  - Memiliki runtime yang terisolasi dengan sistem file, CPU, Memory, dan ruang prosesnya sendiri.
- Container merupakan suatu alternatif modern dibanding dengan deployment pada VM.
  - Semua container berjalan pada VM atau fisik yang sama dan berbagi sistem operasi.
  - VM 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



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



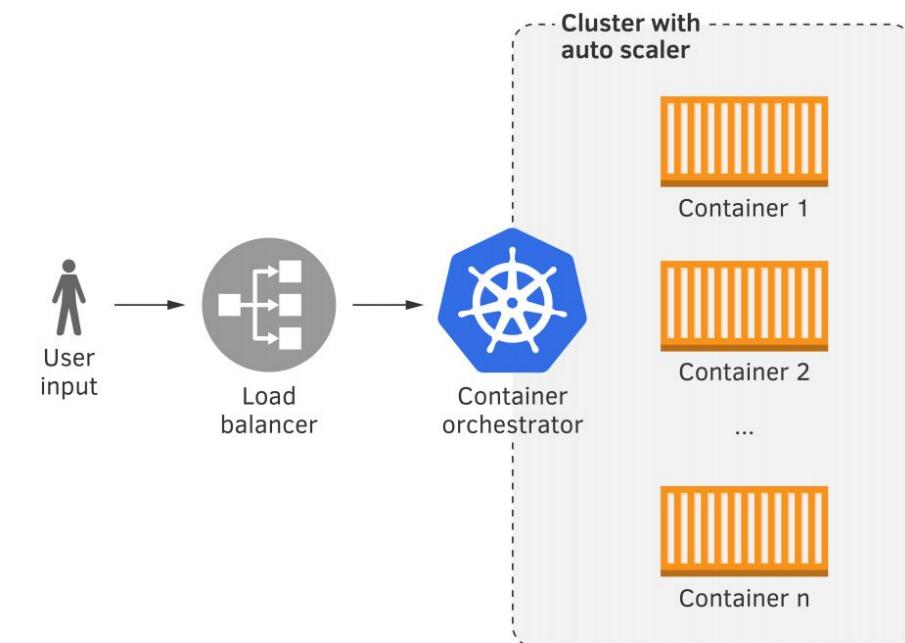
- Resource Polling yang lebih bagus
- Mudah di skalabilitas
- VM ada di cloud

- 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 diinstall pada container (**Docker Container**).
- **Container orchestrator** digunakan untuk menjalankan container pada sekelompok server fisik/virtual.
- **Contoh container orchestrator:** Kubernetes, AWS fargate, Google Kubernetes Enginee.



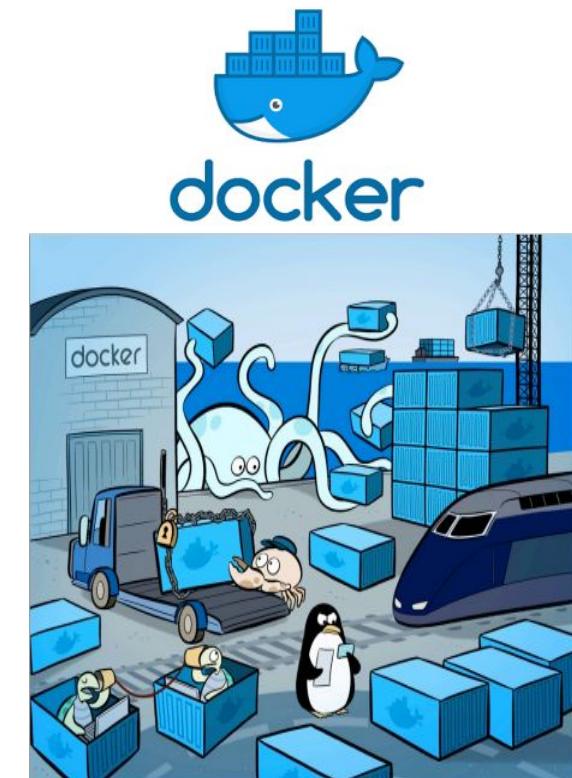
Deploying suatu model sebagai web service pada container yang berjalan pada cluster



## 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*»



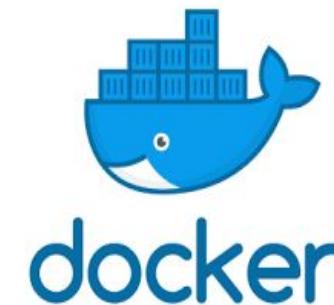


## Penerapan model berbasis pola: Dynamic Deployment(13/21)

### Deployment pada container

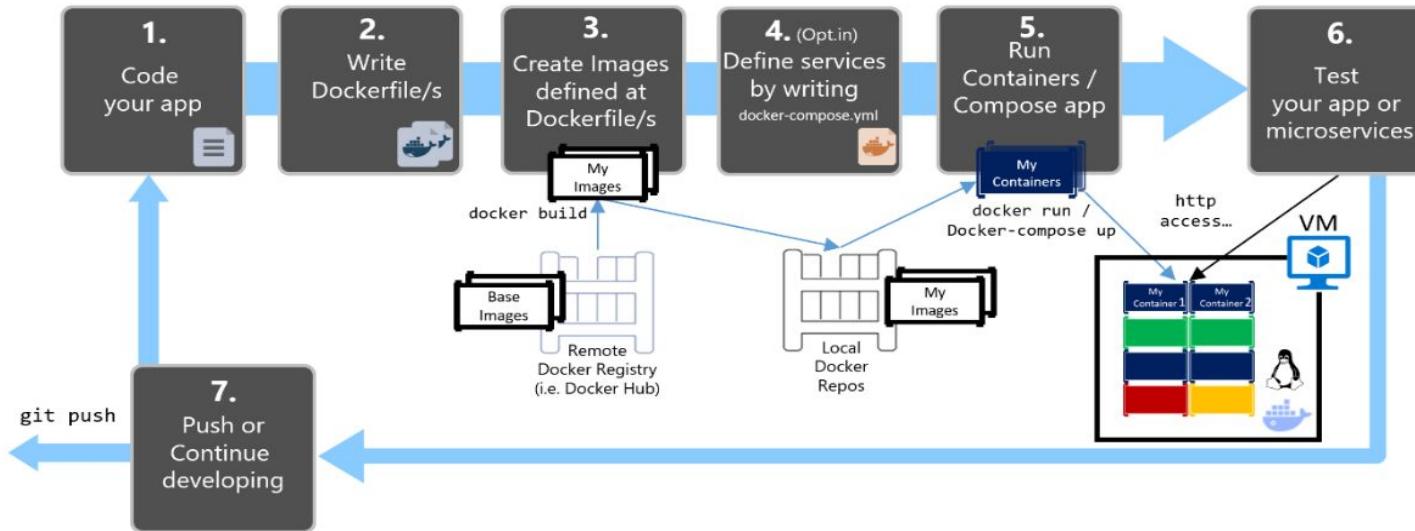
- **Fitur Docker:**

- **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**



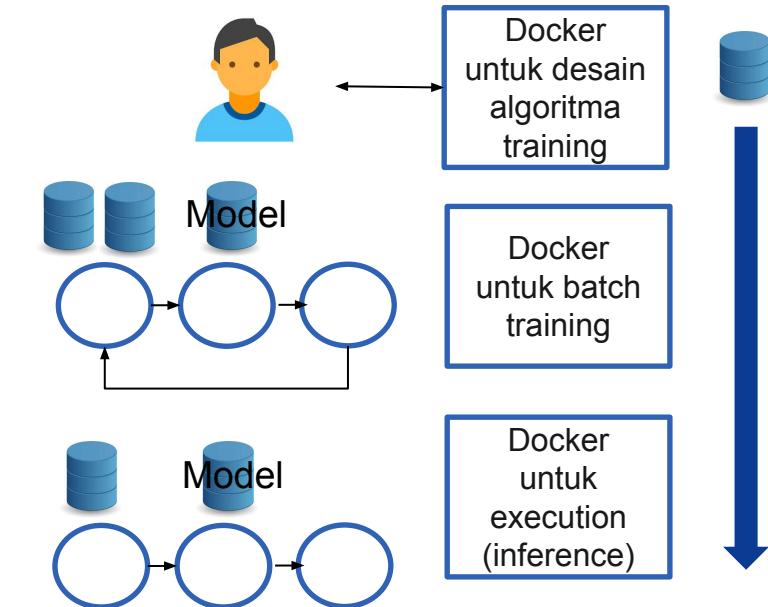
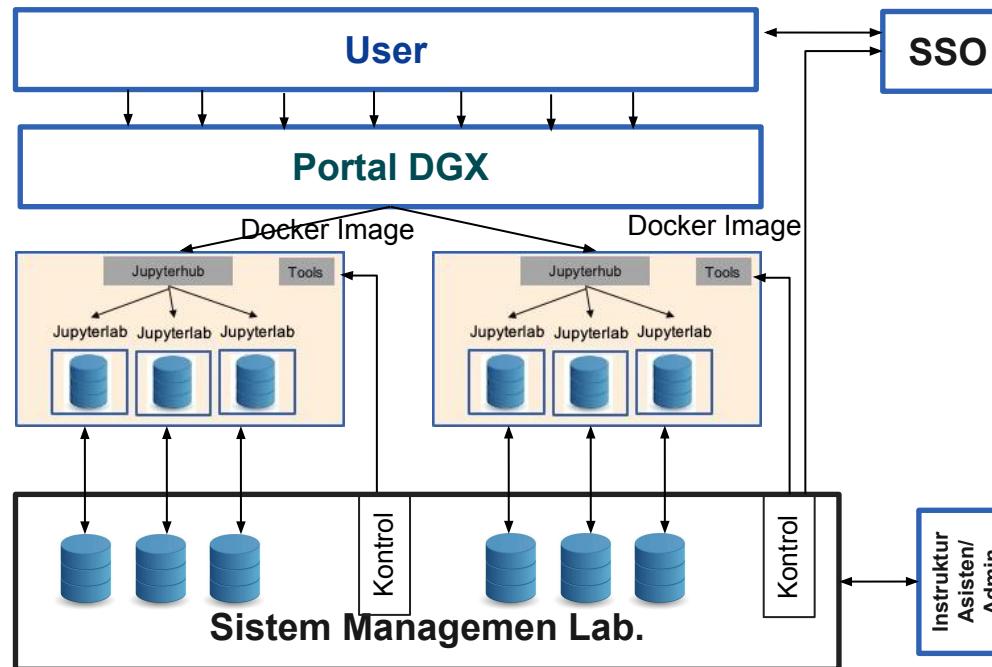
# Perubahan pola deployment

Inner-Loop development workflow for Docker apps



## 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 severless 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:**

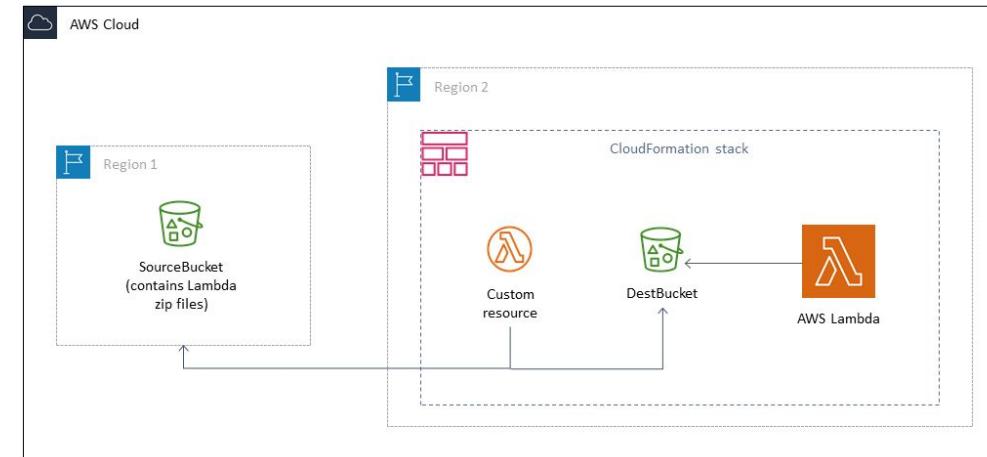
- Menyediakan API untuk mensubmit masukan ke Serverless Function.
- Menangani deploying codes dan model agar memadai pada sumber daya yang dimiliki.
- Mengeksekusi codes dan mengarahkan keluaran kembali ke client.
- Limitasi pada waktu eksekusi fungsi, ukuran file ZIP, dan jumlah RAM
  - Harus menyertakan 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)



Upload your code to AWS Lambda or write code in Lambda's code editor



Set up your code to trigger from other AWS services, HTTP endpoints, or in-app activity



AWS Lambda

Lambda runs your code only when triggered, using only the compute resources needed



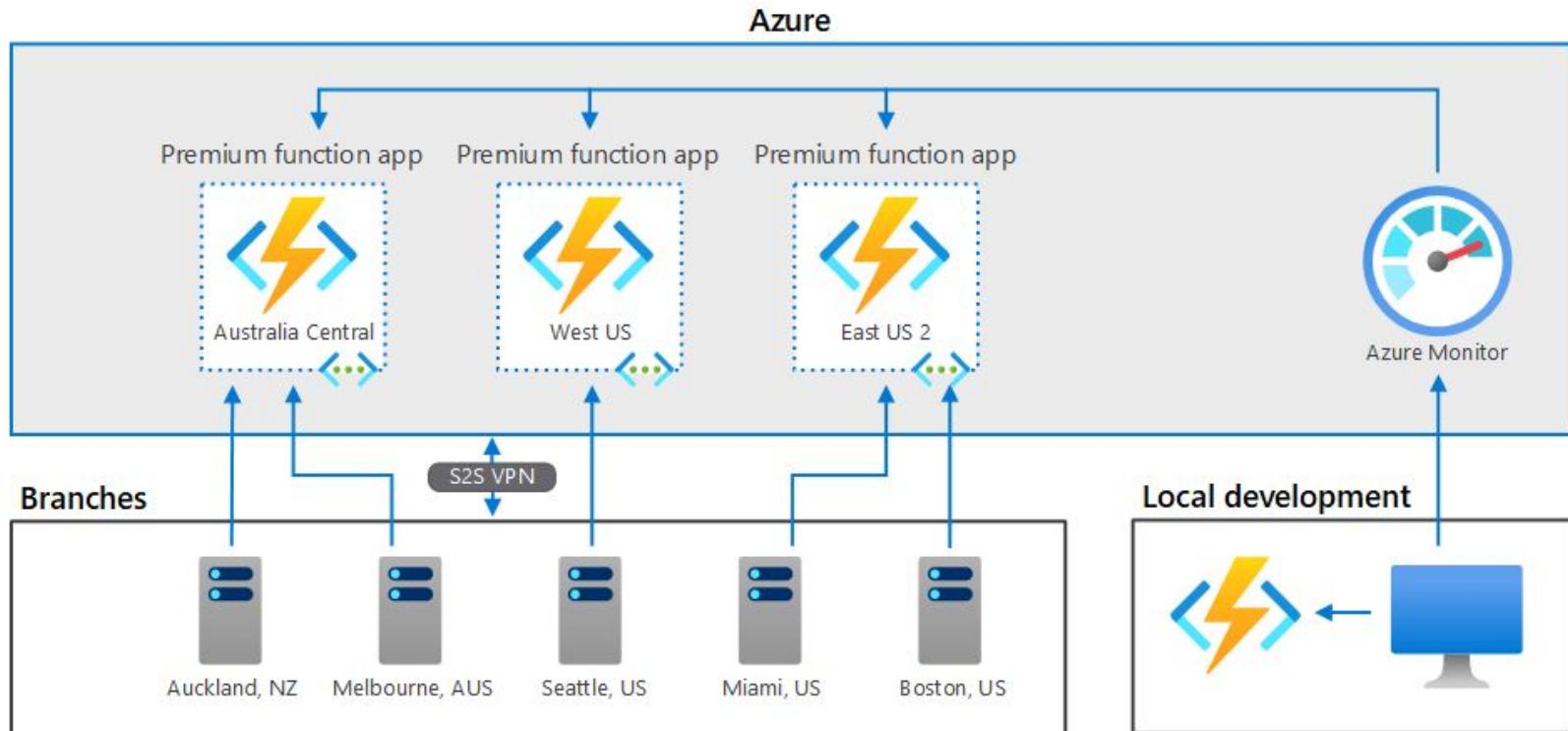
Just pay for the compute time you use



## Penerapan model berbasis pola: Dynamic Deployment(18/21)

### Serverless deployment

Azure Functions (Microsoft)

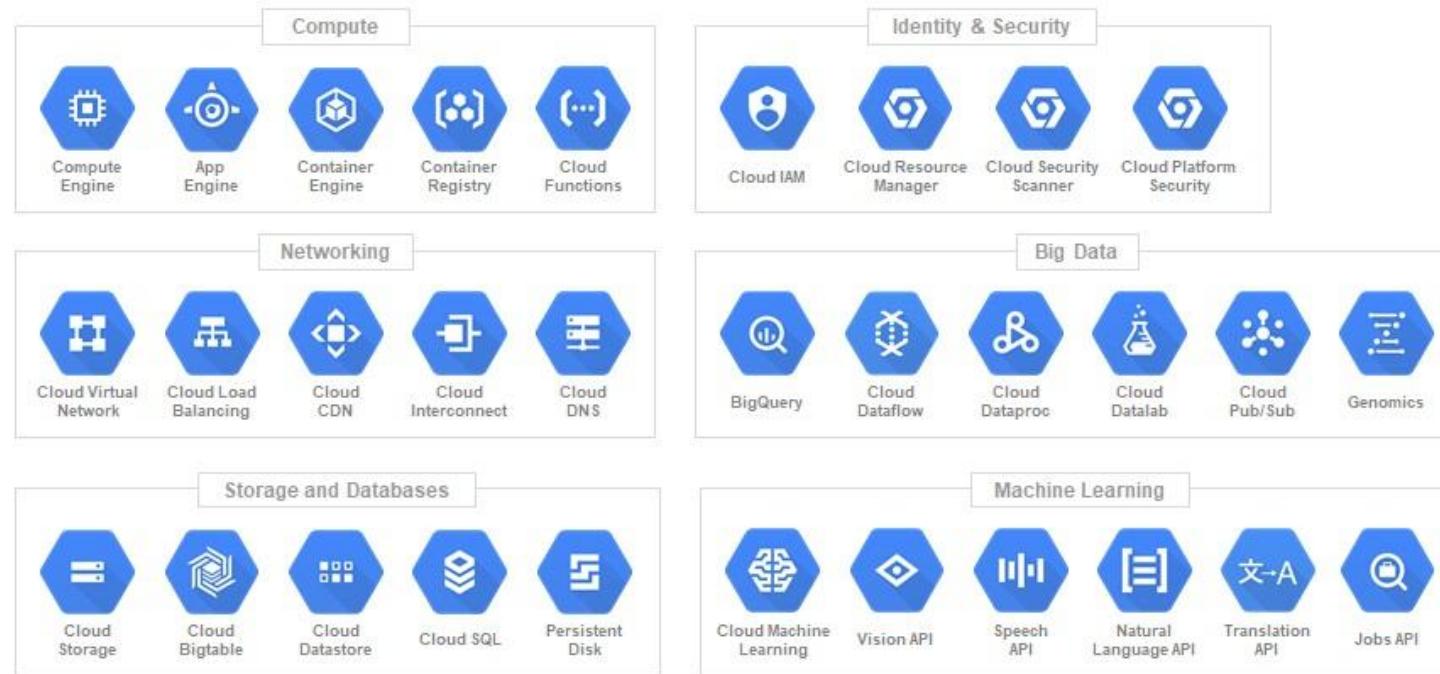


# Penerapan model berbasis pola: Dynamic Deployment(19/21)

## Serverless deployment

**Google Cloud Platform  
(Google)**

Google Cloud Platform Icons





## Penerapan model berbasis pola: Dynamic Deployment(20/21)

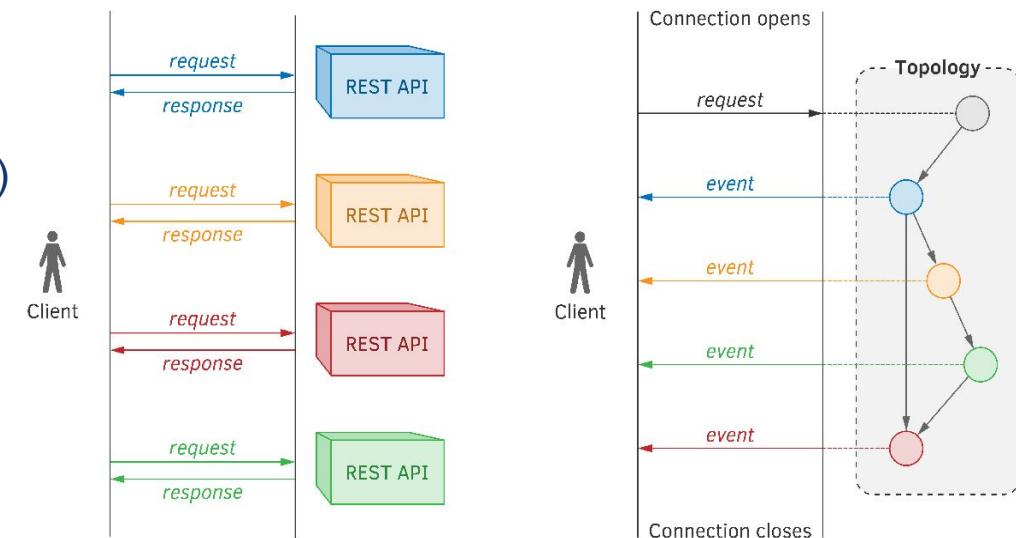
### Serverless deployment

- Kelebihan serverless deployment:
  - Tidak perlu menyediakan server maupun VM.
  - Tidak perlu install depedensi, 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.



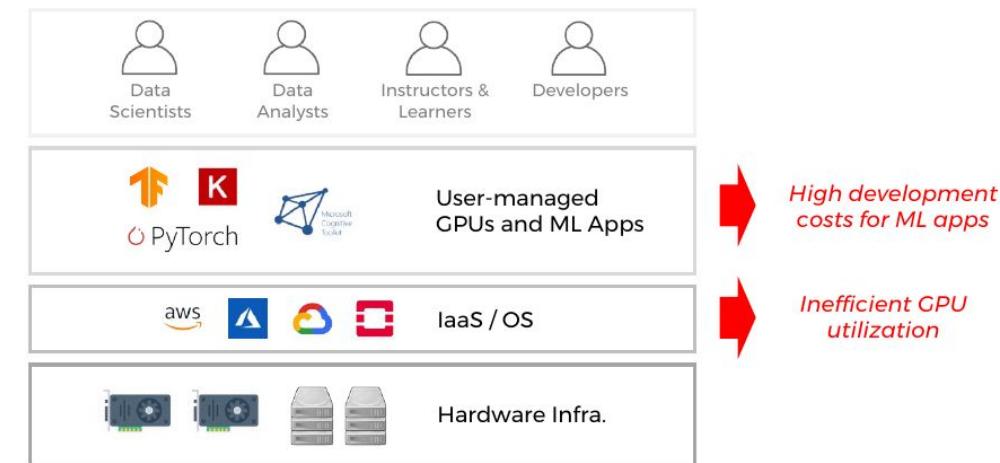
REST API: Memproses elemen data, client menggunakan REST API mengirimkan permintaan satu-per-satu dan menerima respon secara sinkronus

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



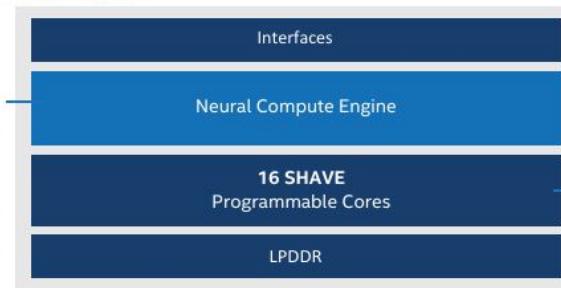
# OpenVINO - INTEL



- Bringing computer vision and AI to your IoT and edge device prototypes are now easier than ever with enhanced capabilities of the Intel® Neural Compute Stick 2 (Intel® NCS2).
- It's built on the latest Intel® Movidius™ Myriad™ X VPU which features the neural compute engine—a dedicated hardware accelerator for deep neural network inferences. With more compute cores than the original version and access to the Intel® Distribution of OpenVINO™ toolkit, the Intel® NCS2 delivers 8X\* performance boost over the previous generation.<sup>1</sup>
- The Intel Distribution of OpenVINO™ toolkit is the default software development kit<sup>1</sup> to optimize performance, integrate deep learning inference, and run deep neural networks (DNN) on Intel® Movidius™ Vision Processing Units (VPU).

## Vision Processing Unit Architecture Intel® Movidius™ Myriad™ X VPU

An entirely new deep neural network (DNN) inferencing engine that offers flexible interconnect and ease of configuration for on-device DNNs and computer vision applications



VLIW (DSP) programmable processors are optimized for complex visions and imaging workloads

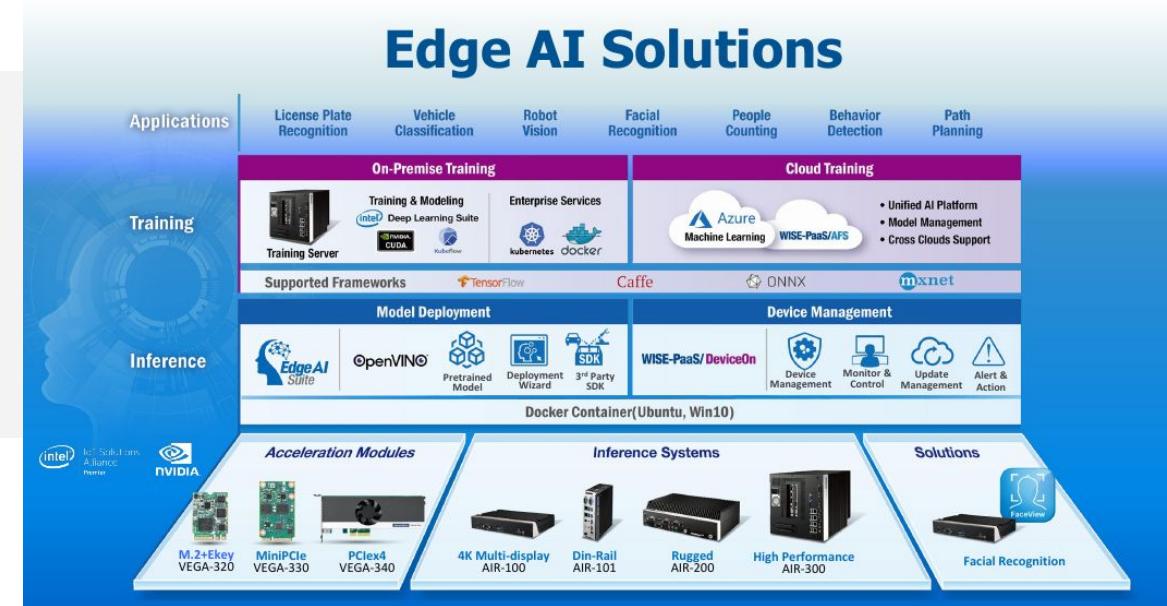
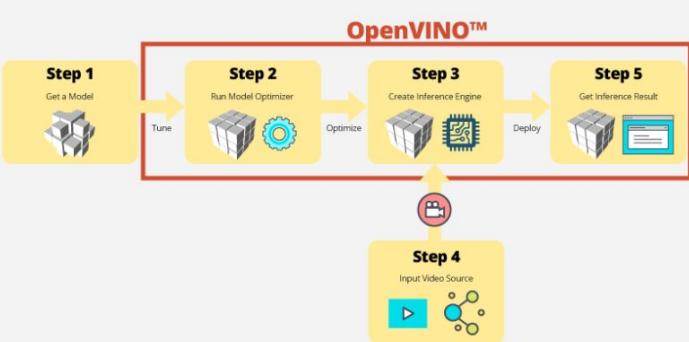
Homogeneous memory design for low-power, UL latency, sustained High Performance

## 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

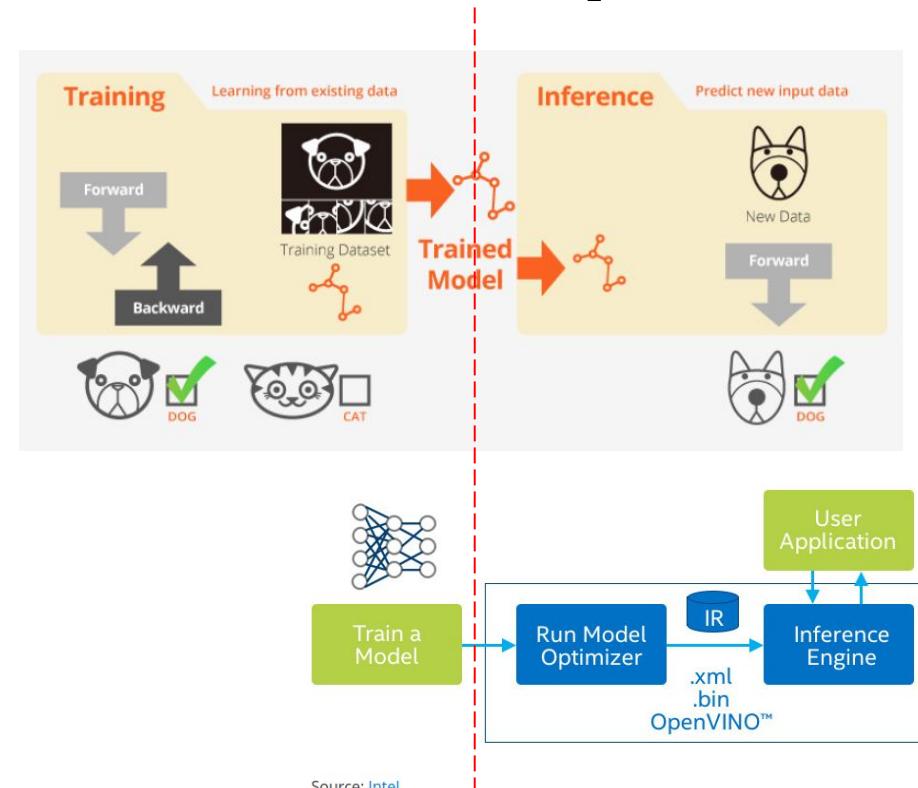


# Pengembangan dengan OpenVINO

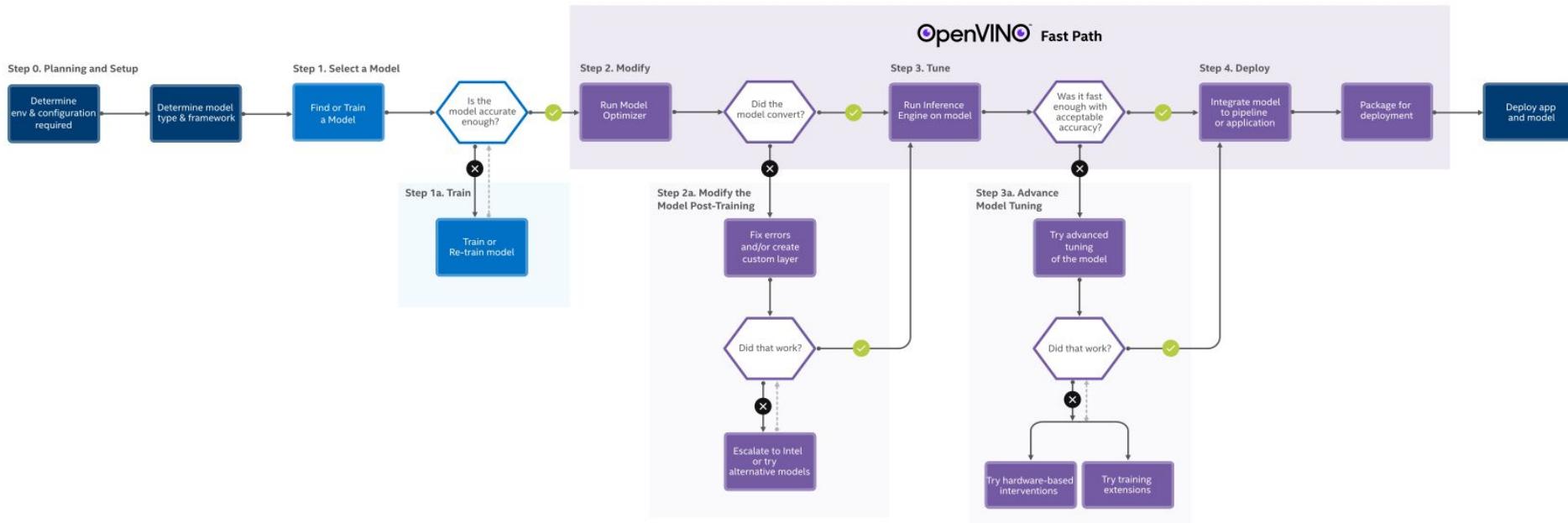




# Proses Inference di OpenVINO



# 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.

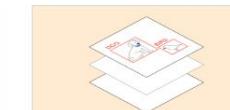
## Trained TensorFlow models for the Edge TPU



### Image classification

Models that recognize the subject in an image, plus classification models for on-device transfer learning.

[→ See models](#)



### Object detection

Models that identify multiple objects and provide their location.

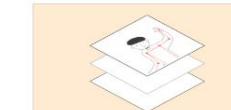
[→ See models](#)



### Semantic segmentation

Models that identify specific pixels belonging to different objects.

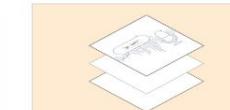
[→ See models](#)



### Pose estimation

Models that identify the location of several points on the human body.

[→ See models](#)



### Speech recognition

Models that recognize speech commands.

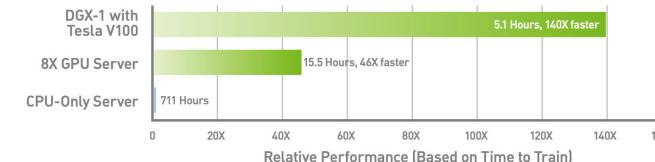
[→ See models](#)



# Penerapan model berbasis pola: AI infrastructure, platform (1/5)



## NVIDIA DGX-1 Delivers 140X Faster Deep Learning Training



## 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

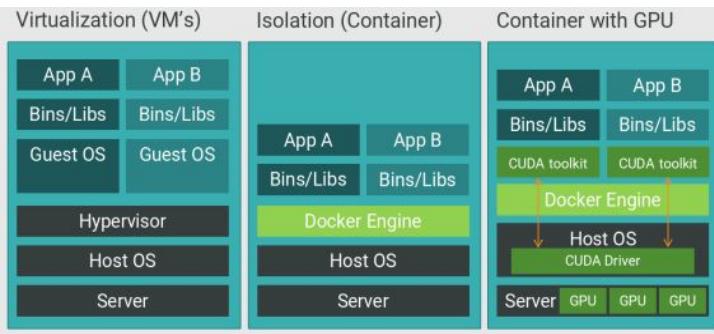
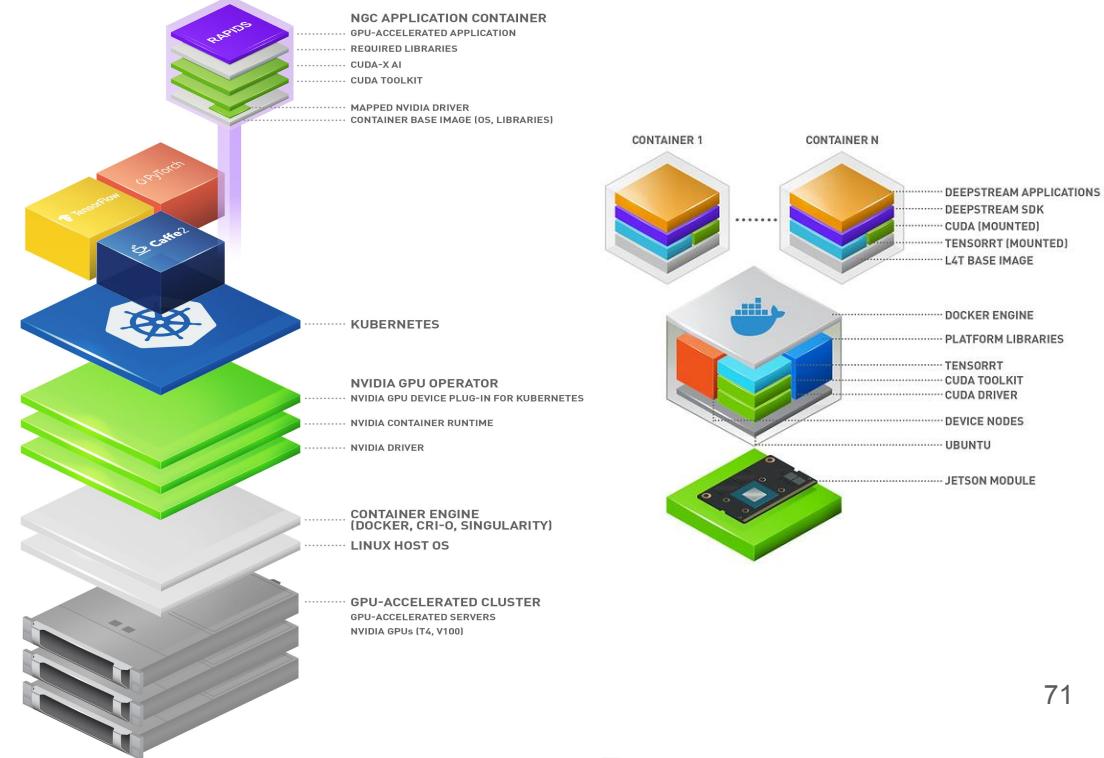
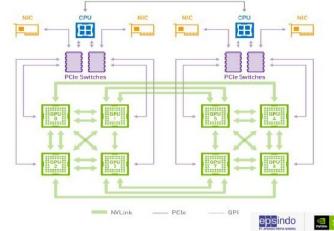
Dual 10GbE, Quad IB 100Gb

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

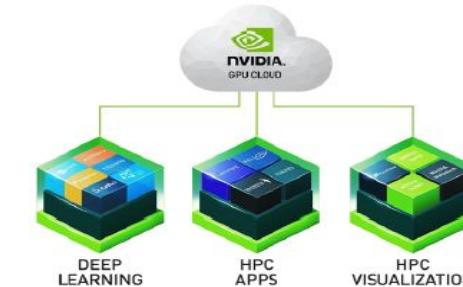
NAMD, Gromacs, LAMMPS, GAMESS, Relion, Chroma, MILC

### 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

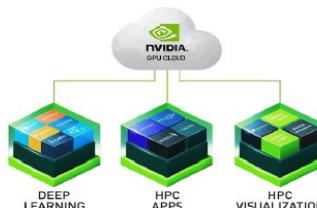
NAMD, Gromacs, LAMMPS, GAMESS, Relion, Chroma, MILC

## HPC Visualization

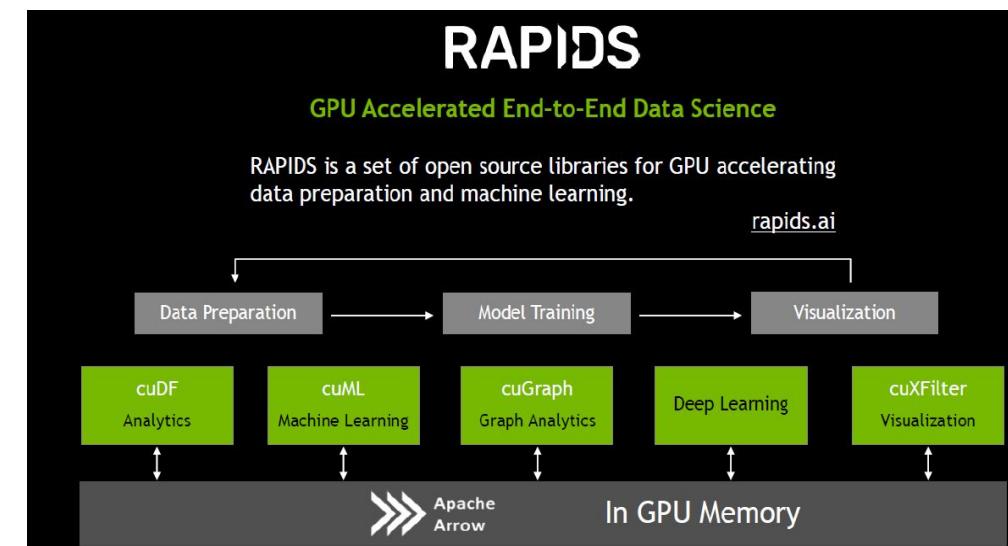
Paraview with Optix, Index and Holodeck with OpenGL visualization base on NVIDIA Docker 2.0, IndeX, VMD

## 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,74 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.



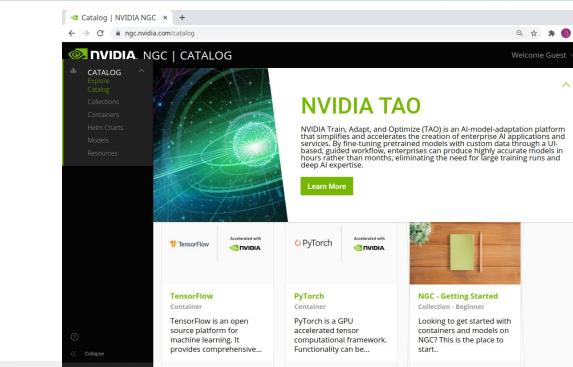
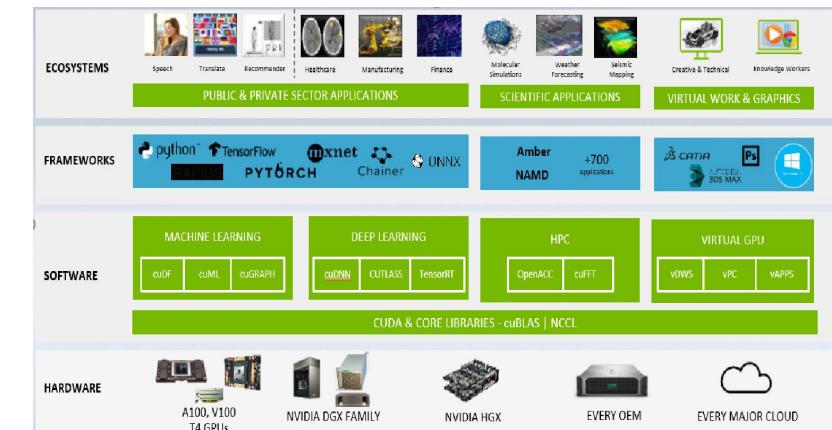
## SYSTEM SPECIFICATIONS

GPUs	8x NVIDIA A100 Tensor Core GPUs	Storage	OS: 2x 1.92TB M.2 NVME drives Internal Storage: 15TB (4x 3.84TB) U.2 NVME drives
GPU Memory	320 GB total	Software	Ubuntu Linux OS
Performance	5 petaFLOPS AI 10 petaOPS INT8	System Weight	271 lbs (123 kgs)
NVIDIA NVSwitches	6	Packaged System Weight	315 lbs (143kgs)
System Power Usage	6.5kW max	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
CPU	Dual AMD Rome 7742, 128 cores total, 2.25 GHz (base), 3.4 GHz (max boost)	Operating Temperature Range	5°C to 30°C (41°F to 86°F)
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		



# Software Architecture

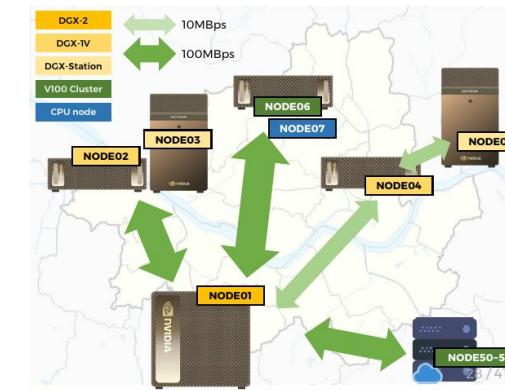
- Like previous generations, DGX-100 uses a containerized architecture:
  - A minimal operating system and driver install
  - All applications and SDKs provisioned as containers, accessible through NGC Privacy Registry
- The NGC Private Registry provides:
  - Containers enabling deep learning, machine 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:

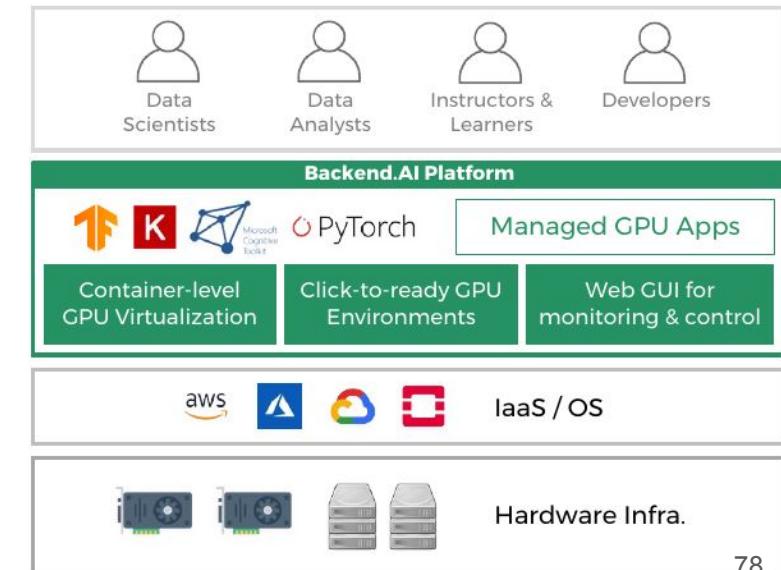
Computing Power	100-700 PetaFLOPS
Number of DGX A100 nodes	20-140
Number of GPUs	160-1120
Storage	1-10 Petabytes
NVIDIA Mellanox Bandwidth	200 Gbps





# 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)

Superior computing power | All-scenario deployment | Cloud-edge-device collaboration

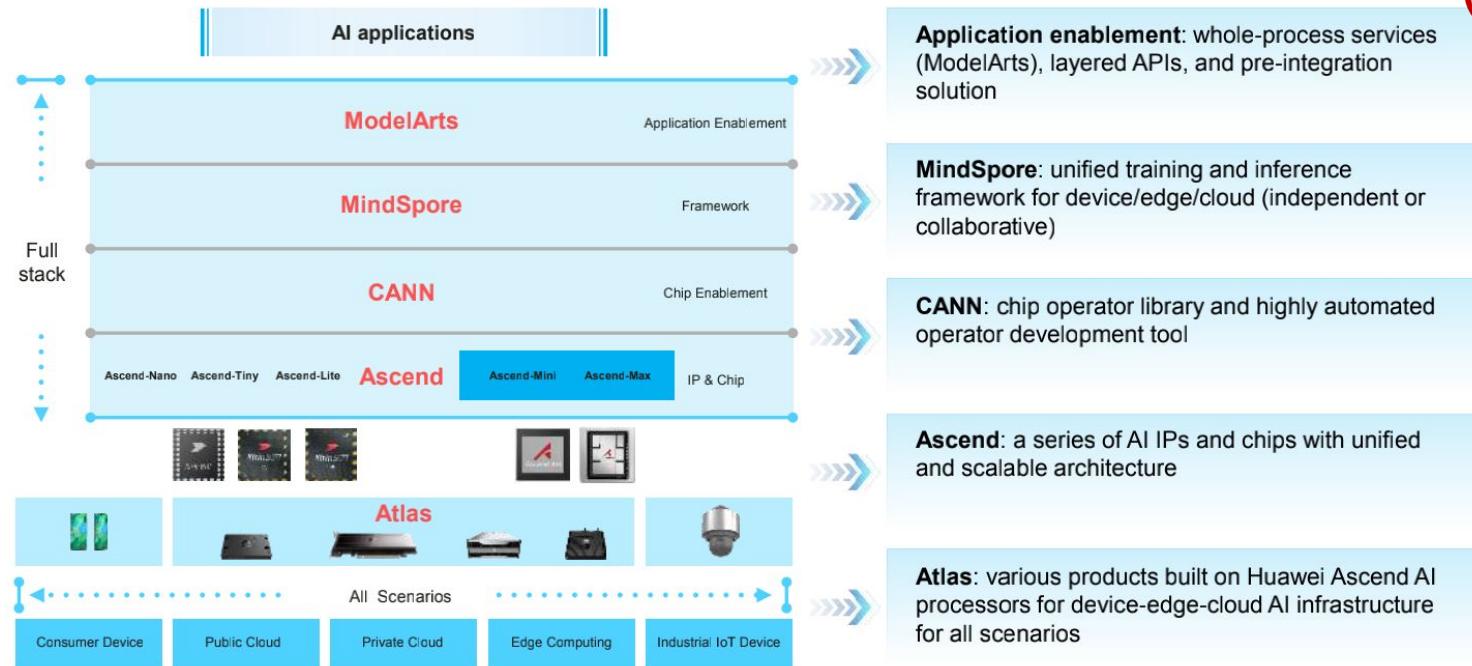




# Penerapan model berbasis pola: AI infrastructure, platform (3/5)

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

(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.

### 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	Up to 7 Atlas 300I inference cards
AI Computing Power	Up to 16 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 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: Dimensions 748 mm x 86.1 mm x 447 mm Chassis with 2.5" drives: 708 mm x 86.1 mm x 447 mm

### 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)

### 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

## 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.

### 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 <sup>1</sup>
Cooling Mode	Air cooling
Fan Modules	8 hot-swappable fan modules, supporting N+1 redundancy
Operating Temperature	5°C to 40°C (Liquid Cooling) 5°C to 35°C (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.



# Proses pada Ascend

Figure 3-1 Engine process

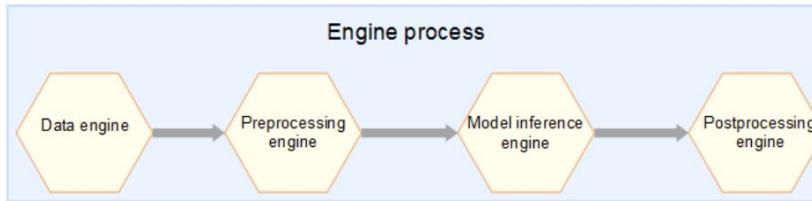
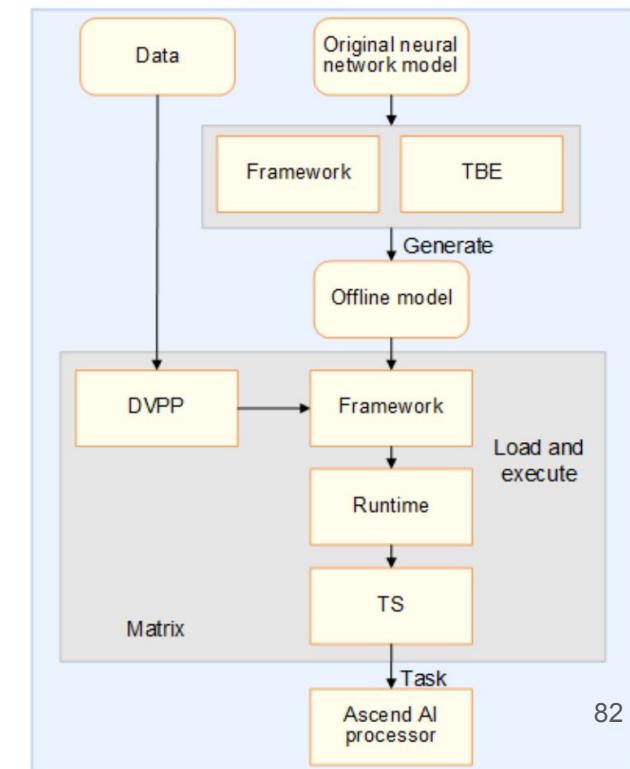
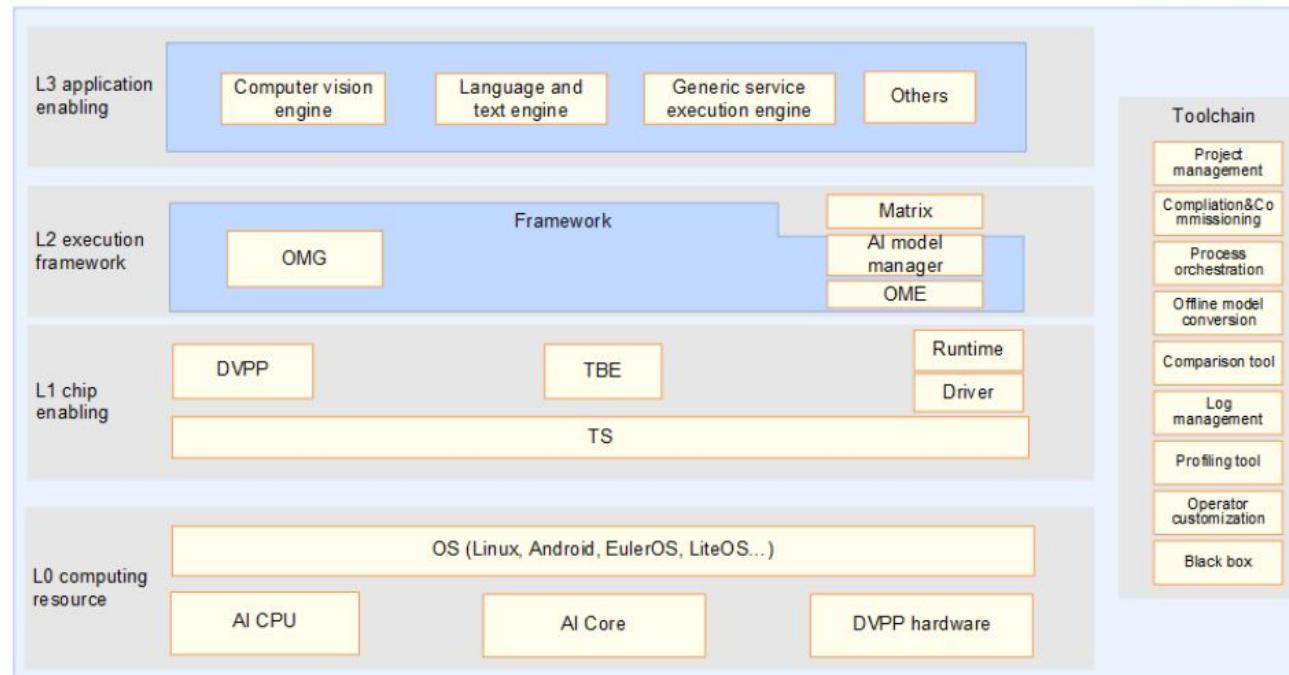


Figure 2-1 Neural network software architecture

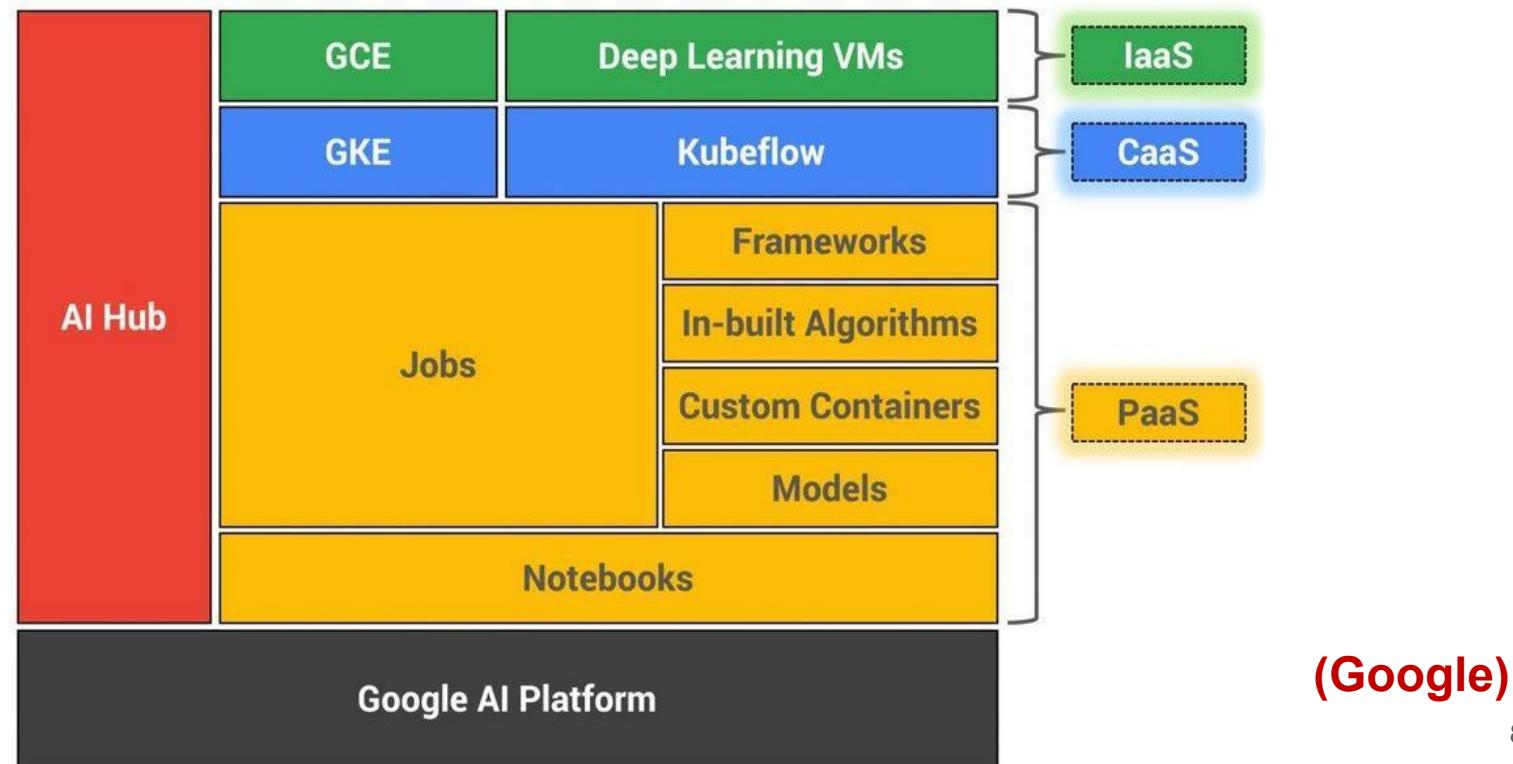


# Software Stack

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

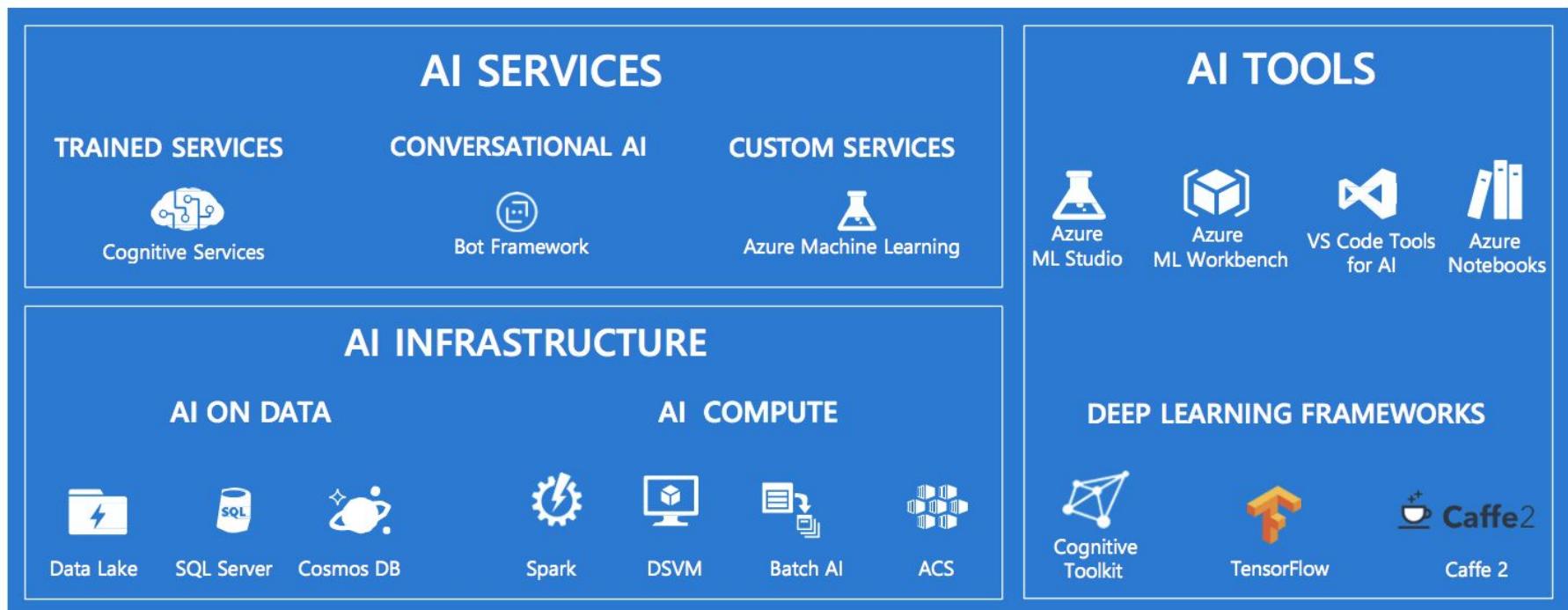


## Penerapan model berbasis pola: AI infrastructure, platform (4/5)



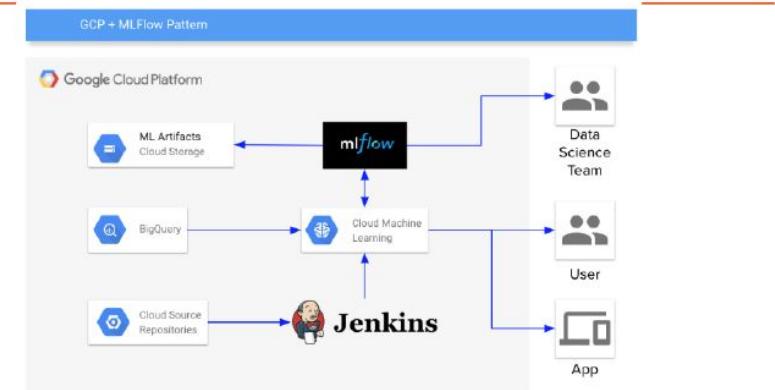
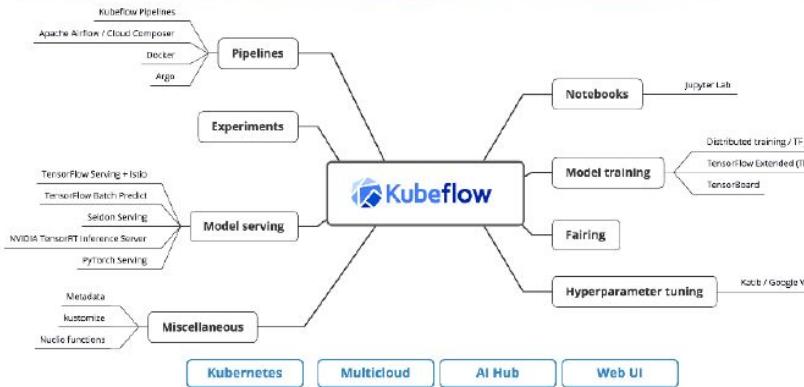


# Penerapan model berbasis pola: AI infrastructure, platform (5/5)

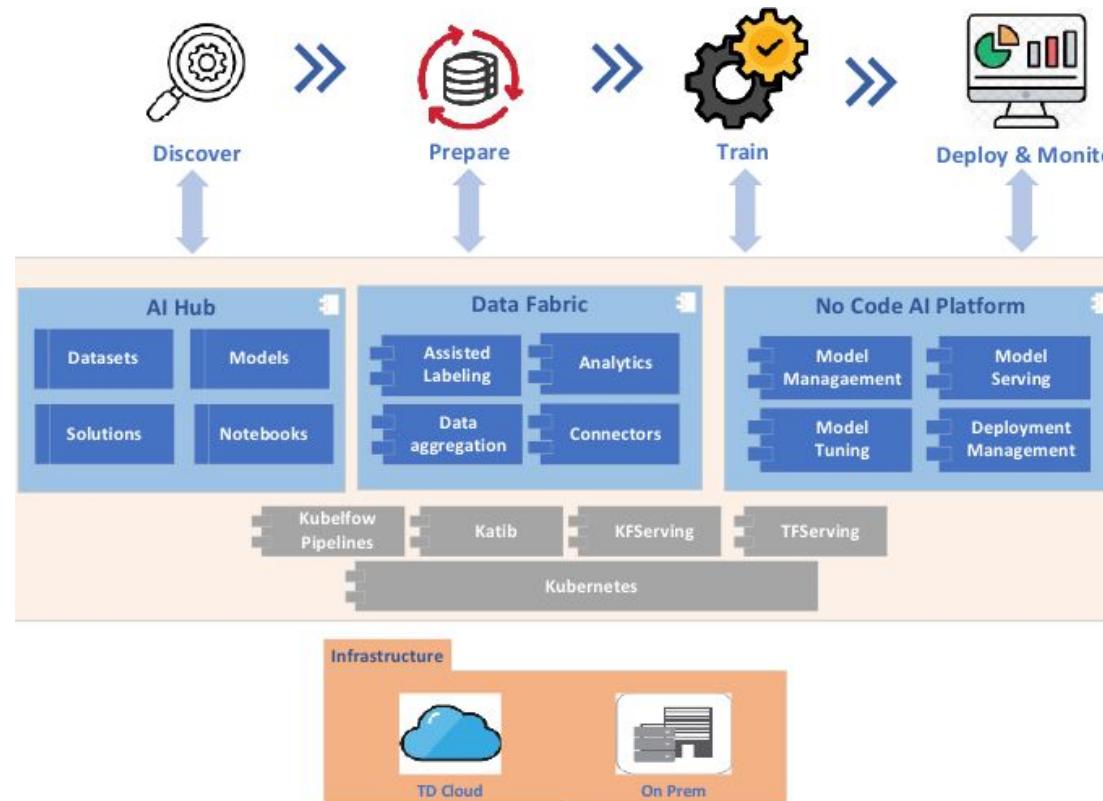




# Tools untuk MLOps

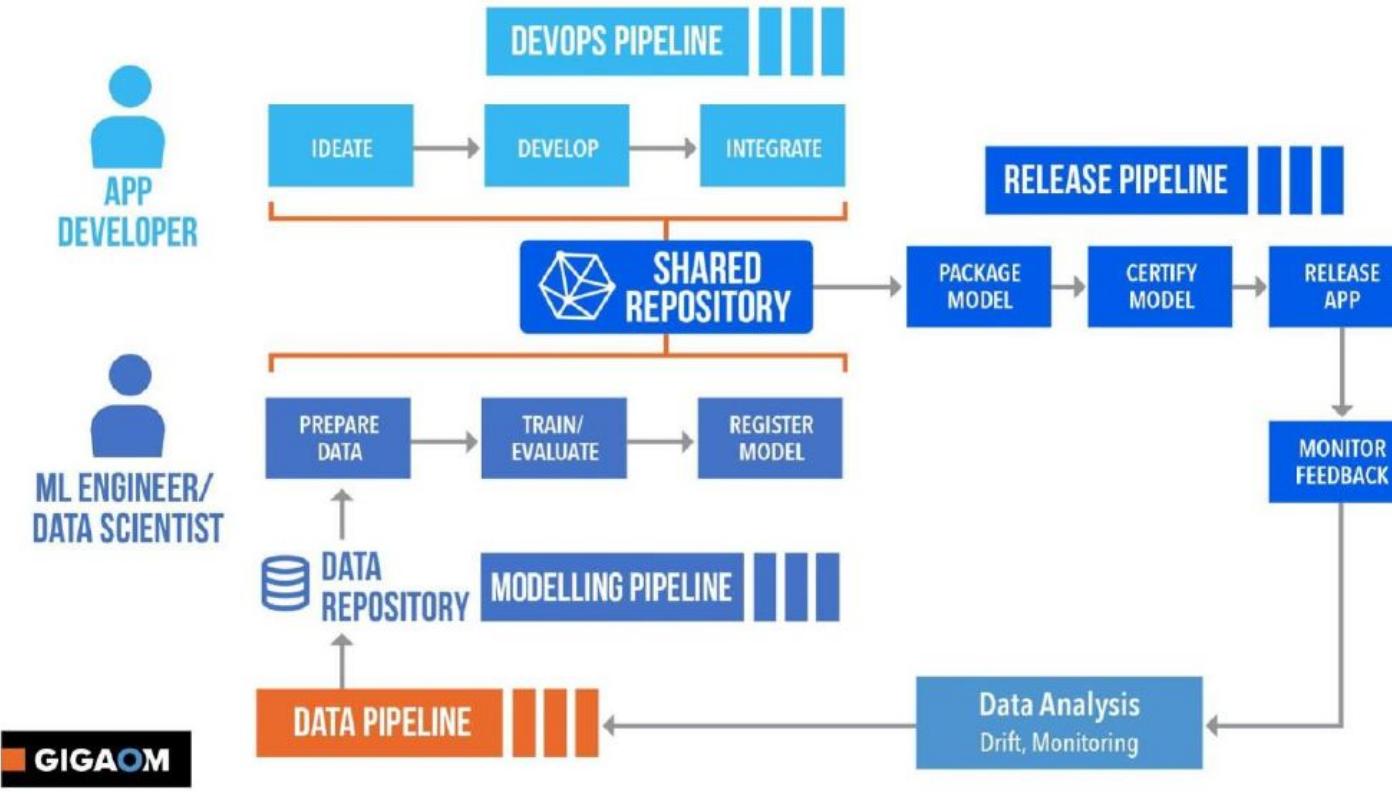


# Environment utk Produksi



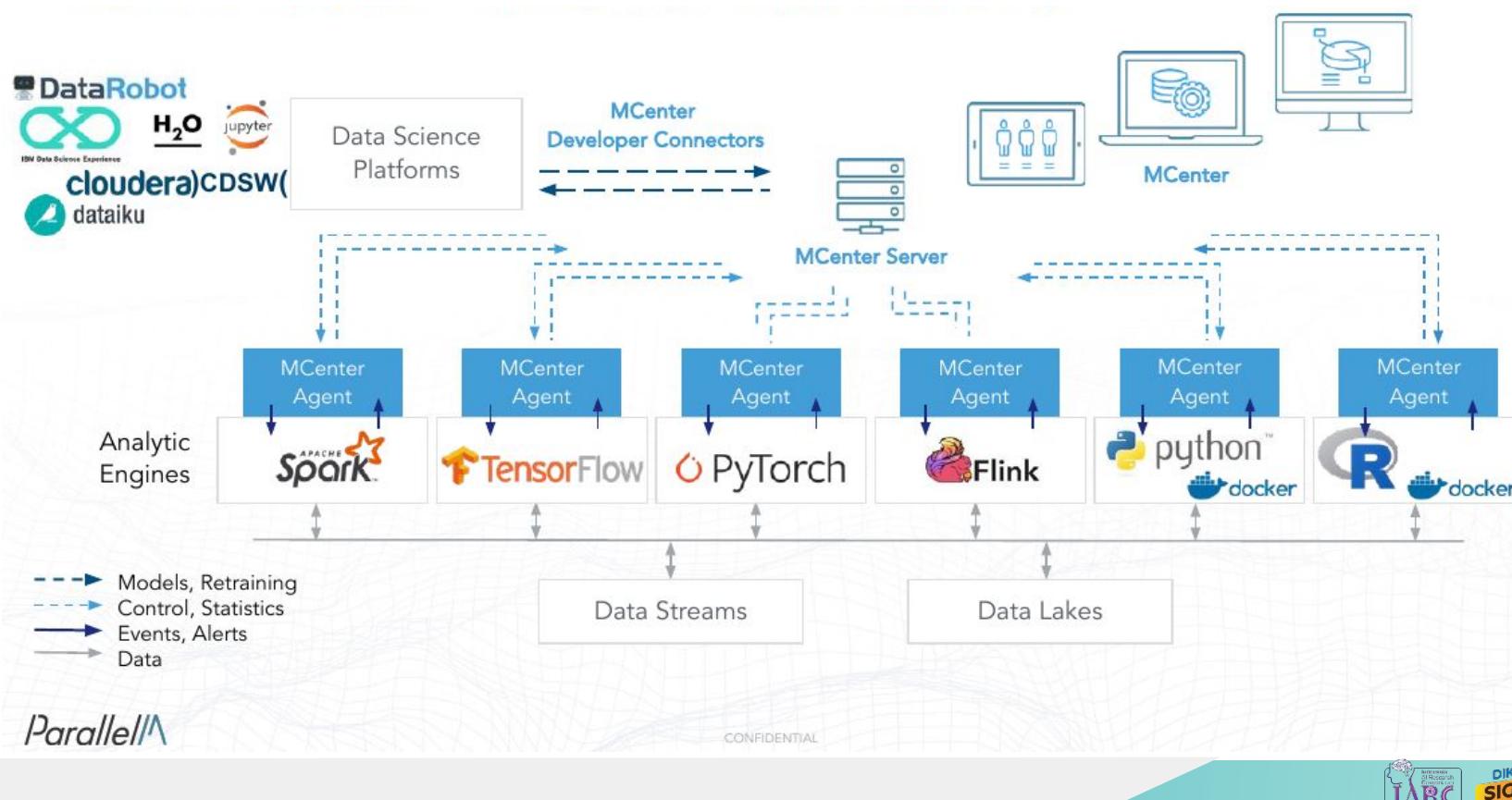


# Siklus terintegrasi



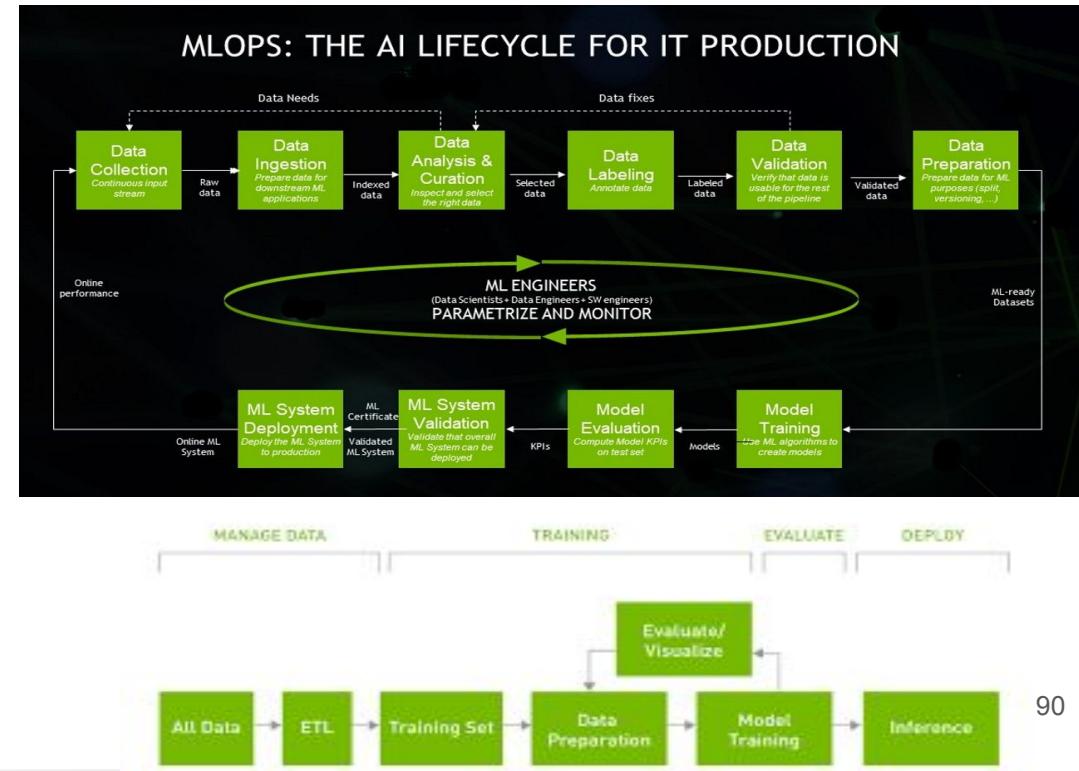
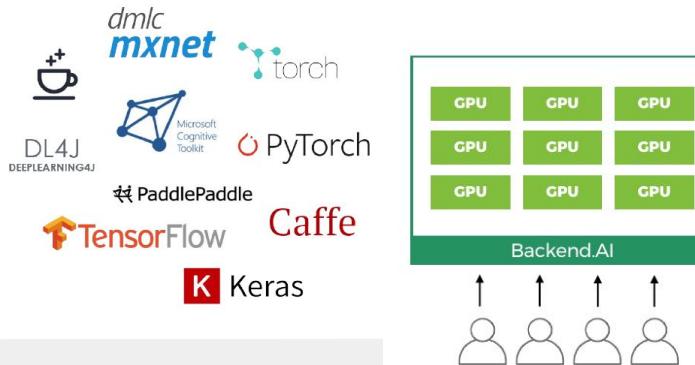


# Contoh Environment Produksi



# Pipe Lining

- DeepOps
- Slurm
- ClearML <https://clear.ml/>
- Run AI:
- Backend.AI <http://www.backend.ai>



# 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.

The screenshot shows a web browser window titled "SAN sandbox". The URL is "ui.neptune.ai/vhffm/sandbox/experiments?viewid=standard-view". The interface includes a top navigation bar with "Projects", "Explore", "Experiments", "Notebooks", "Wiki", and "Settings". Below this is a search bar and a filter section with dropdowns for "Time", "Tags", "Experiment Status", "Owned By", and "Select users". A link "Go to advanced search" is also present. The main area is a table titled "All experiments" with columns: ID, State, Name, Created, Finished, Running..., decay\_fac, n\_iterations, AUC, loss, and Tags. The table lists nine experiments, all labeled "SUCCEEDED". Each row contains a checkbox, the experiment ID, its state, name, creation and finish times, and numerical values for the other metrics. The "Tags" column shows "tag1" and "tag2" for some rows.

ID	State	Name	Created	Finished	Running...	decay_fac	n_iterations	AUC	loss	Tags
SAN-9	SUCCEEDED	ex6	17 APR 10:20:00	17 APR 10:20:00	< 1 min	0.5	117	0.96	0.00623214	tag1 tag2
SAN-8	SUCCEEDED	ex5	17 APR 10:20:00	17 APR 10:20:00	< 1 min	0.5	117	0.96	0.00623214	
SAN-7	SUCCEEDED	ex4	17 APR 10:20:00	17 APR 10:20:00	< 1 min	0.5	117	0.96	0.00623214	
SAN-6	FAILED	ex4	17 APR 10:20:00	17 APR 10:20:00	< 1 min	0.5	117	0.96	0.00623214	
SAN-5	FAILED	ex4	17 APR 10:20:00	17 APR 10:20:00	< 1 min	0.5	117	0.96	0.00623214	
SAN-4	SUCCEEDED	ex3	17 APR 10:20:00	17 APR 10:20:00	< 1 min	0.5	117	0.96	0.00623214	
SAN-3	SUCCEEDED	ex2	17 APR 10:20:00	17 APR 10:20:00	< 1 min	0.5	117	0.96	0.00623214	
SAN-2	SUCCEEDED	minimal_example	17 APR 10:20:00	17 APR 10:20:00	< 1 min	-	-	0.96	0.00623214	
SAN-1	SUCCEEDED	minimal_example	17 APR 10:20:00	17 APR 10:20:00	< 1 min	-	-	0.96	0.00623214	



```
$ 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"
```

# MLFlow

- MLFlow is an experiment tracker and generic model server. It can be self-hosted.
- 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). Artifacts can be logged to local, remote, or cloud storage (S3, GFS, etc).
- Results can be analyzed through a web UI and CSV export is available. Models are packaged as a wrapper around the underlying format (Sklearn, XGBoost, Torch, etc). They can be pushed to Spark for batch inference or served through REST.
- There are CLI, Python, R, Java, and REST APIs for further integration with CI/CD pipelines. Models can be pushed to cloud services (SageMaker, AzureML, ...).

The screenshot shows the MLflow web application running at localhost:5000. The main page displays a table of experiments under the 'Default' tab. Each experiment row includes columns for Start Time, Run Name, User, Source, Version, alpha, IT\_ratio, mse, r2, and lrmse. The table shows several runs, with one run highlighted in blue. At the top, there are search and filter options, and a note section indicating 'None'.

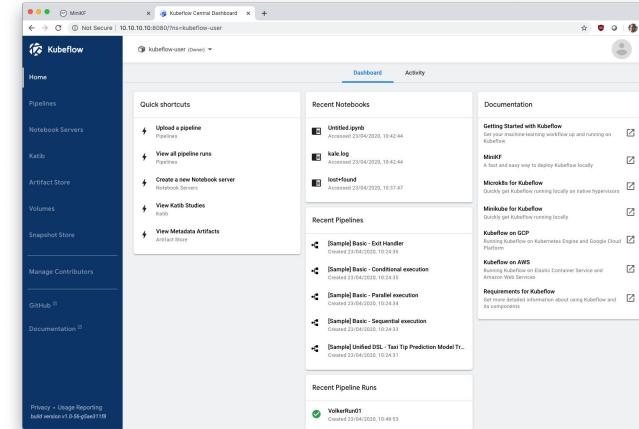
Start Time	Run Name	User	Source	Version	alpha	IT_ratio	mse	r2	lrmse
2020-04-17 11:1 -	volker	train.p	-	1.0	1.0	0.673	0.017	0.833	
2020-04-17 11:1 -	volker	train.p	-	1.0	0.5	0.668	0.017	0.833	
2020-04-17 11:1 -	volker	train.p	-	0.5	1.0	0.668	0.017	0.833	
2020-04-17 11:1 -	volker	train.p	-	0.5	0.5	0.627	0.109	0.793	
2020-04-17 11:2 -	volker	sklearn	-	0.42	0.1	0.572	0.22	0.742	
2020-04-17 11:1 -	volker	mlflow	7193f0	5	0.1	-	-	-	
2020-04-17 11:0 -	volker	main.p	-	-	-	-	-	-	



```
$ conda create --name mlflow python=3.6
$ conda activate mlflow
$ conda install -c conda-forge mlflow
$ cd /somewhere
$ mlflow ui
---> http://localhost:5000
---> https://mlflow.org/docs/latest/quickstart.html
```

# Kubeflow

- .Kubeflow is essentially a self-hosted version of the Google AI platform. It uses Kubernetes to abstract away infrastructure.
- .Kubeflow can deploy Jupyter notebooks, run pipelines for data processing and model training (scheduled, on-demand), organize runs, archive models and other artifacts, and expose models through endpoints.  
Pipelines are compute graphs and are described in Python with a DSL. Their components are wrapped as Docker images.
- .It integrates with GCP so it can elastically scale to the cloud compute and storage (e.g., distributed model training). It also integrates with offerings such as BigQuery or Dataproc.
- .The solution is heavy and complex but enables rapid scale-out. It is especially applicable if infrastructure is already managed through Kubernetes.



<https://www.kubeflow.org/docs/started/workstation/getting-started-minikf/>

```
$ cd /somewhere
$ vagrant init arrikto/minikf
$ vagrant up
```

--> <http://10.10.10.10>

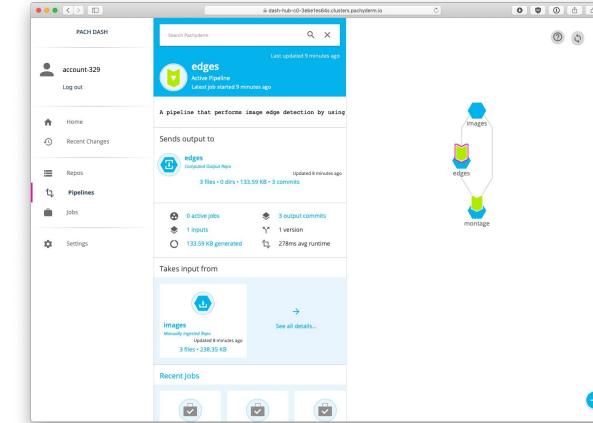
# Pachyderm

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.



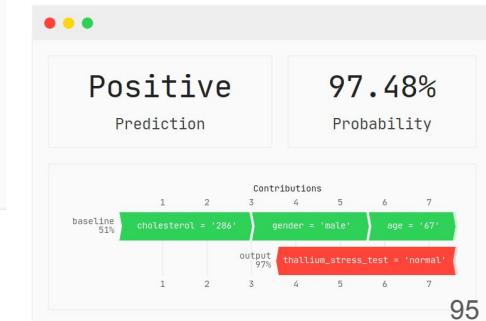
```
$ cd /somewhere
$ download from (https://github.com/pachyderm/pachyderm/releases)
$ tar xfvz release_filename_linux_amd64.tar.gz
$ pachctl version --client-only
```

---> [https://docs.pachyderm.com/latest/pachub/pachub\\_getting\\_started/](https://docs.pachyderm.com/latest/pachub/pachub_getting_started/) 94  
--- [https://docs.pachyderm.com/latest/getting\\_started/beginner\\_tutorial/](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 running `tangram train` 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 a `.tangram` file. If you want more control, you can provide a config file.

```
$ tangram train --file heart_disease.csv --target disease
✓ Loading train data.
✓ Loading test data.
✓ Shuffling train data.
✓ Shuffling test data.
✓ Inferring column types.
✓ Computing train stats.
✓ Computing test stats.
✓ Computing baseline metrics.
✓ Computing features.
info: Press ctrl-c to stop early and save the best model.
✓ Training model 1 of 8.
✓ Training model 2 of 8.
✓ Training model 3 of 8.
✓ Computing model comparison features.
✓ Computing comparison metric.
✓ Computing features.
info: Training model 4 of 8.
[=====>
```





# Solusi Cloud One-Stop



Amazon SageMaker



data  
iku



databricks



Azure ML



Google AI Platform

# Rekomendasi





# MLOps.org

## Motivation for MLOps



You will learn for what to use Machine Learning, about various scenarios of change that need to be managed and the iterative nature of ML-based software development. Finally, we provide the MLOps definition and show the evolution of MLOps.

[Read more](#)

## Designing ML-powered Software



This part is devoted to one of the most important phase in any software project — understanding the business problem and requirements. As these equally apply to ML-based software you need to make sure to have a good understanding before setting out designing things.

[Read more](#)

## End-to-End ML Workflow Lifecycle



In this section, we provide a high-level overview of a typical workflow for machine learning-based software development.

[Read more](#)

## Three Levels of ML-based Software



You will learn about three core elements of ML-based software — Data, ML models, and Code. In particular, we will talk about

- Data Engineering Pipelines
- ML Pipelines and ML workflows.
- Model Serving Patterns and Deployment Strategies

[Read more](#)

## MLOps Principles



In this part, we describe principles and established practices to test, deploy, manage, and monitor ML models in production.

[Read more](#)

## CRISP-ML(Q)



You will learn about the standard process model for machine learning development.

[Read more](#)

## MLOps Stack Canvas



In this part, you will learn how to specify an architecture and infrastructure stack for MLOps by applying a general MLOps Stack Canvas framework, which is designed to be application- and industry-neutral.

[Read more](#)

## ML Model Governance



This part presents an overview of governance processes, which are an integral part of MLOps.

[Read more](#)

## State of MLOps (Tools & Frameworks)



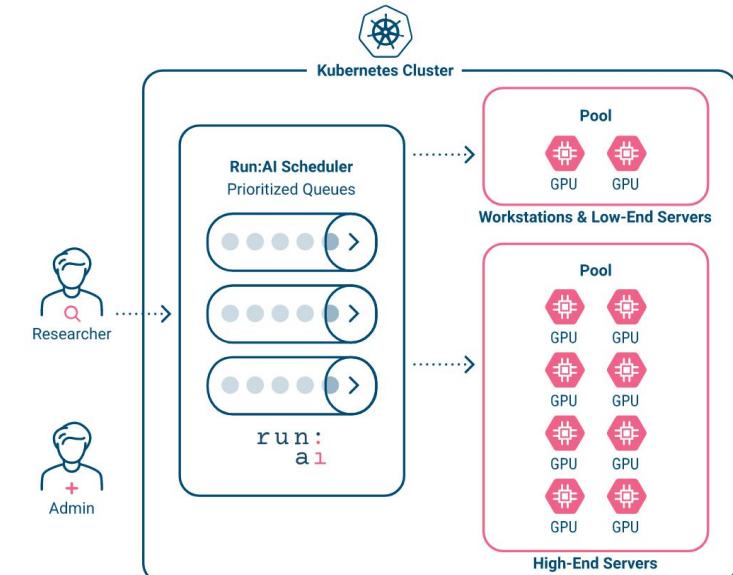
This part presents an overview of software tools and frameworks that manage ML artifacts, and covers the whole machine learning cycle.

[Read more](#)



# 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.

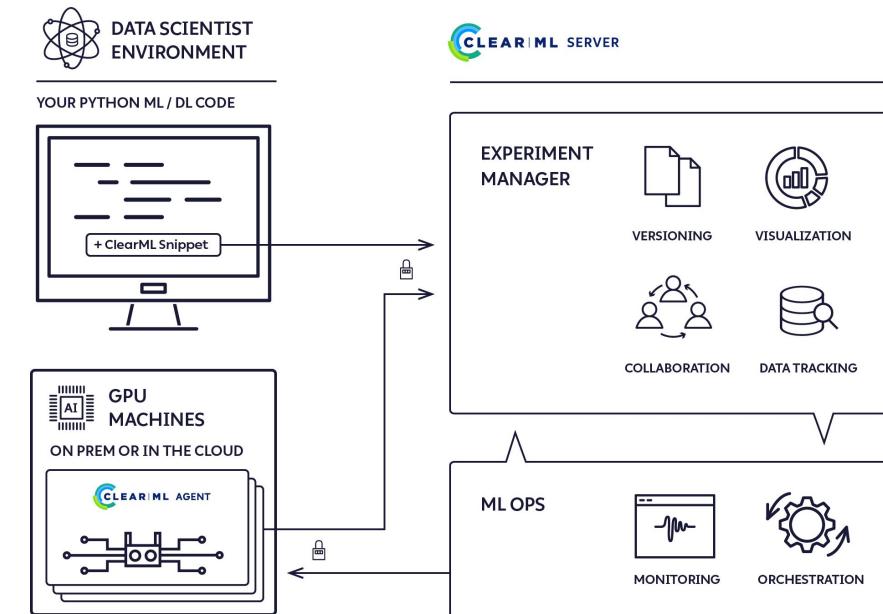
# Run AI

- 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.



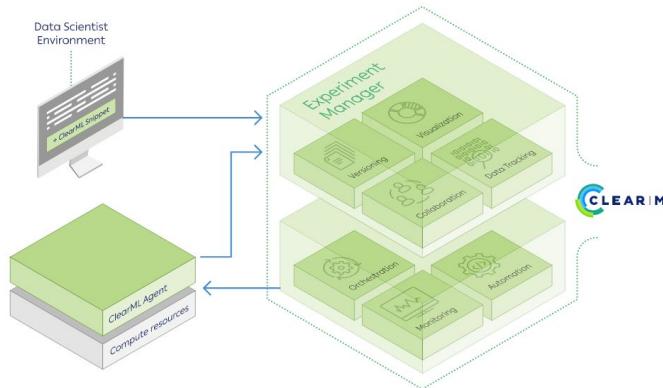
# What can we do with ClearML

- 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!

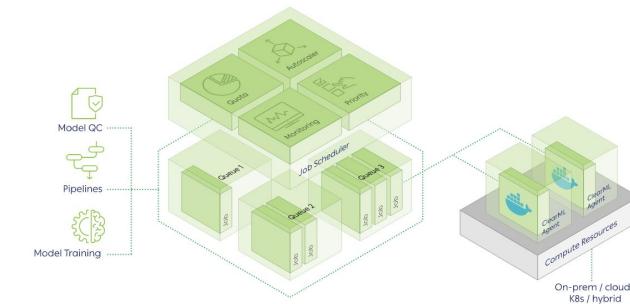




# ClearML

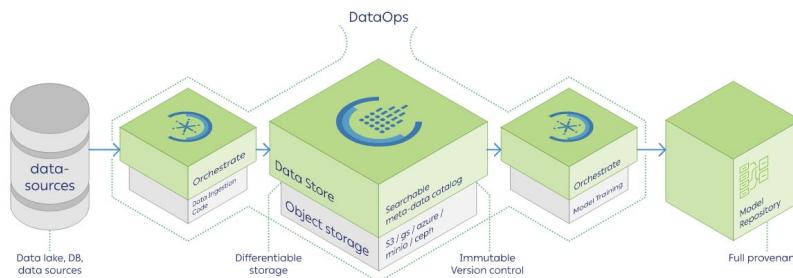


Orchestration with ClearML

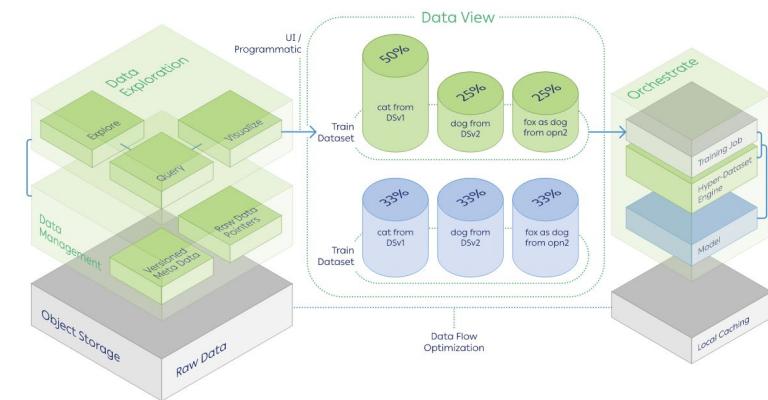


# ClearML

The DataOps process by ClearML

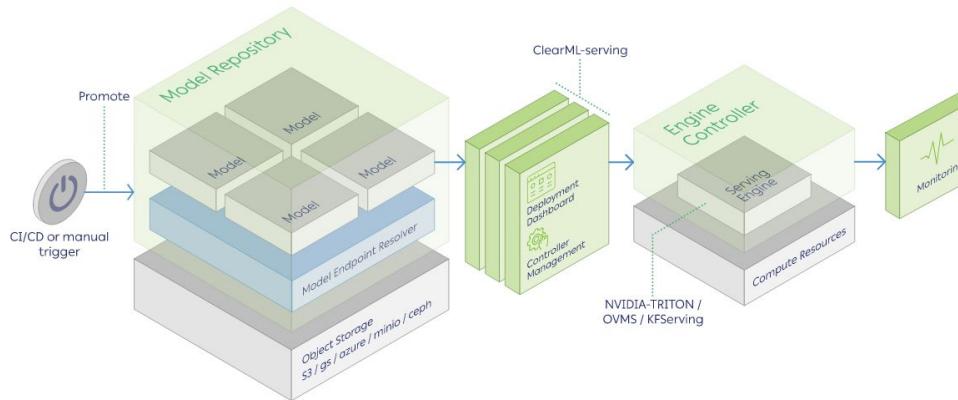


Hyper-Dataset workflow

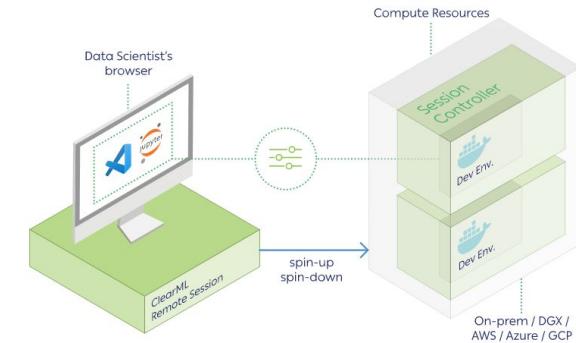


# ClearML

ModelOps workflow by ClearML



Remote Session by ClearML





# Penerapan model berbasis pola

- 



"Hey Cortana"

"Hey Alexa"

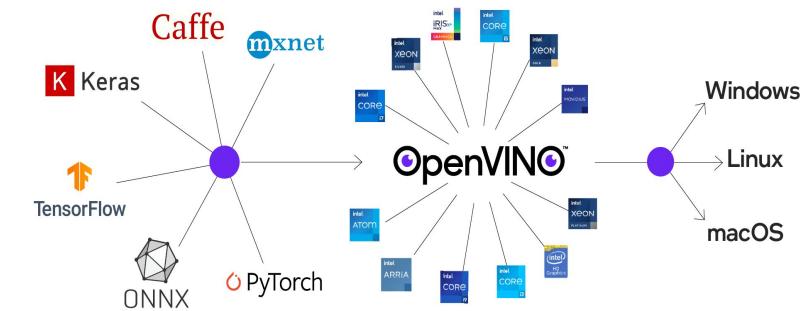
"Hey Siri"

"Hey Google"



# 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, dll (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



108



## Deployment strategy(2/6)

### Single Deployment

- Single deployment merupakan strategi yang paling sederhana.
- Model dibuat serial menjadi file, file lama bisa direplace 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 direplace
  - 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)

### Silent 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'))
```



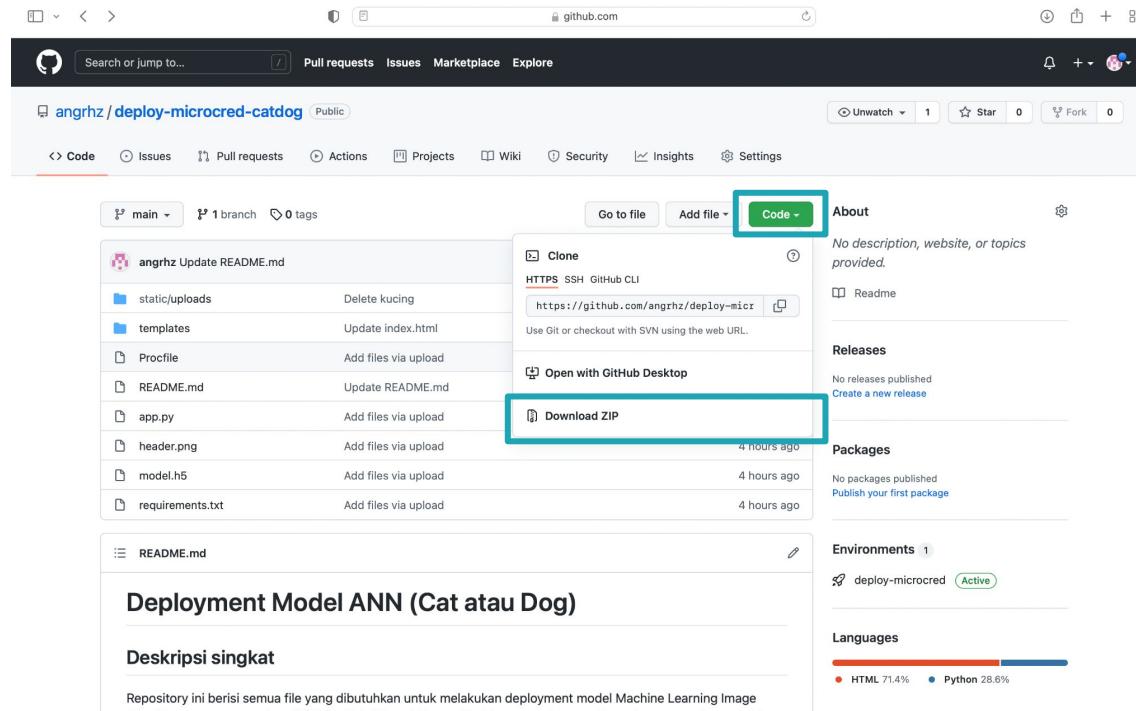
## Deployment Model: Langkah-Langkah model deployment (inferencing)

0. Download dan lakukan installasi perangkat lunak di bawah ini



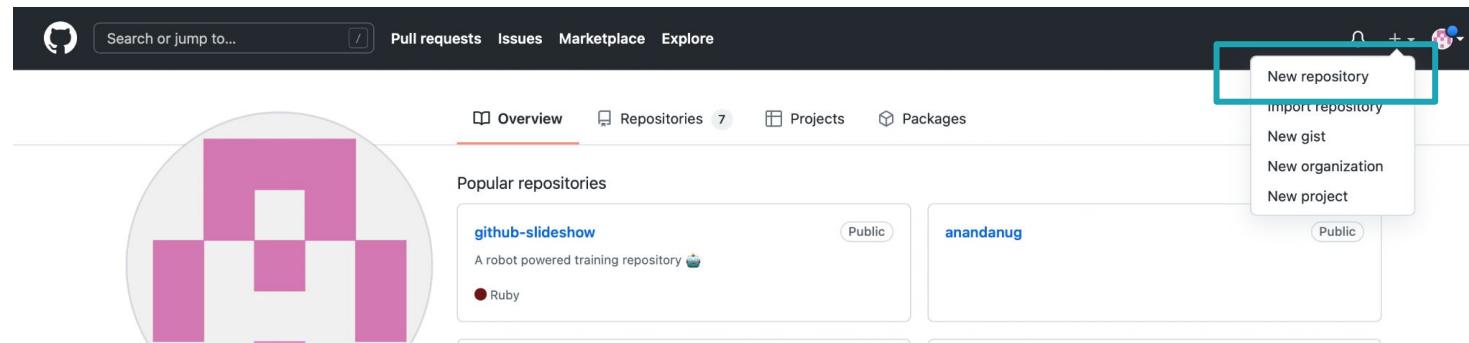
# Deployment Model: Langkah-Langkah model deployment

## 1. Download source code dari repository Github



# Deployment Model: Langkah-Langkah model deployment

## 2. Membuat repository baru



# Deployment Model: Langkah-Langkah model deployment

## 3. Tulis nama repository

Owner \* Repository name \*

 angrhz / deploy-micrcrd-2021 ✓

Great repository names are short and memorable. Need inspiration? How about [miniature-memory](#)?

Description (optional)

## 4. Pilih create repository

 **Public**  
Anyone on the internet can see this repository. You choose who can commit.

 **Private**  
You choose who can see and commit to this repository.

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

**Add a README file**  
This is where you can write a long description for your project. [Learn more](#).

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more](#).

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more](#).

**Create repository**



# Deployment Model: Langkah-Langkah model deployment

## 5. Pilih uploading an existing file

The screenshot shows a GitHub repository page for 'angrhz / deploy-micorcred-2021'. The 'Code' tab is selected. A prominent callout box highlights the 'Uploading an existing file' option under the 'Quick setup' section. Below this, there are sections for command-line deployment and importing code from other repositories.

**Quick setup — if you've done this kind of thing before**

Set up in Desktop or [HTTPS](https://github.com/angrhz/deploy-micorcred-2021.git) [SSH](https://github.com/angrhz/deploy-micorcred-2021.git)

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

**...or create a new repository on the command line**

```
echo "# deploy-micorcred-2021" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/angrhz/deploy-micorcred-2021.git
git push -u origin main
```

**...or push an existing repository from the command line**

```
git remote add origin https://github.com/angrhz/deploy-micorcred-2021.git
git branch -M main
git push -u origin main
```

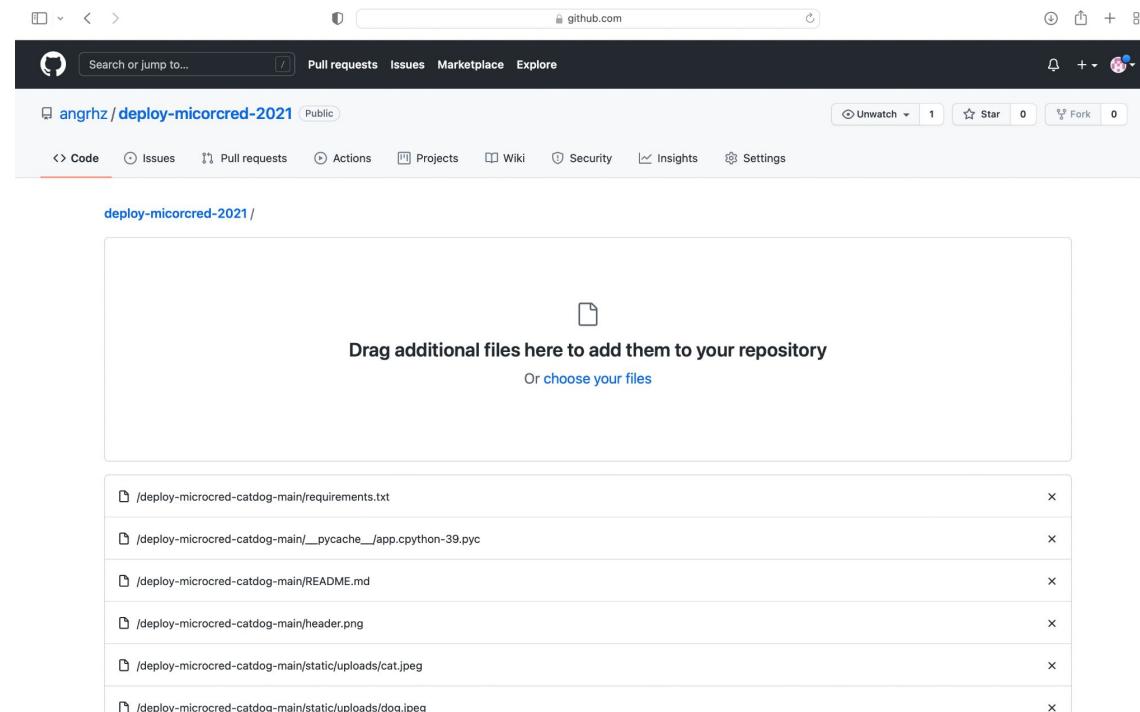
**...or import code from another repository**

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

# Deployment Model: Langkah-Langkah model deployment

## 6. Upload file repository yang telah di download tadi





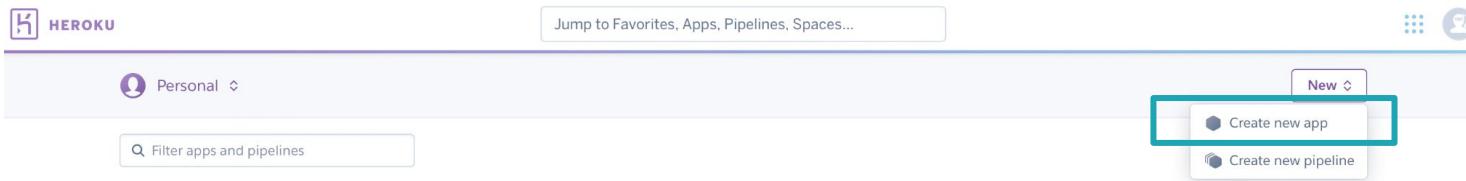
# Deployment Model: Langkah-Langkah model deployment

7. Masuk ke heroku.com. Jika belum punya akun silahkan sign up terlebih dahulu

The screenshot shows the Heroku homepage. At the top, there's a navigation bar with links for Cloud Application Platform, Products, Marketplace, Pricing, Documentation, Support, COVID-19, and More. Below the navigation is a search bar and a 'Log in or Sign up' button. The main visual is a large, stylized purple illustration of a winding path through clouds and trees. To the right of the illustration, the text 'MOVE FAST' is displayed above the heading 'Unleash your inner startup'. A subtext explains: 'Choose Heroku for the same reasons disruptive startups do: it's the best platform for building with modern architectures, innovating quickly, and scaling precisely to meet demand.' Below this are two buttons: 'SIGN UP FOR FREE' and 'Explore Heroku Customers'. At the bottom of the page, there's a section titled 'OFFICIALLY SUPPORTED LANGUAGES' featuring icons for Node.js, Ruby, Java, PHP, Python, Go, Scala, and Clojure. A note below the icons states: 'In addition to our officially supported languages, you can use any language that runs'.

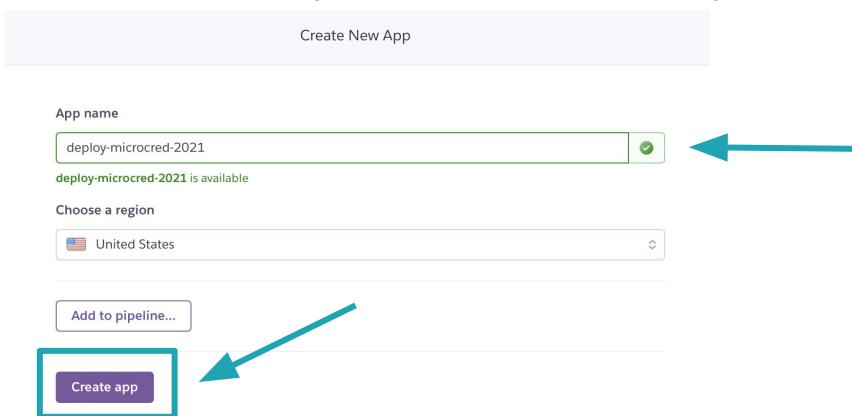
# Deployment Model: Langkah-Langkah model deployment

8. Klik New dan pilih Create new app



The screenshot shows the Heroku dashboard interface. At the top right, there is a 'New' button with a dropdown arrow. A teal box highlights the 'Create new app' option under this button. Below it, another option 'Create new pipeline' is visible.

9. Masukkan nama aplikasi. Kemudian create aplikasi



Create New App

App name  
 

deploy-microcred-2021 is available

Choose a region

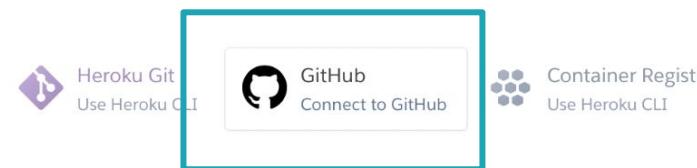
Add to pipeline...



# Deployment Model: Langkah-Langkah model deployment

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

Deployment method



Connect to GitHub

Connect this app to GitHub to enable code  
diffs and deploys.

Search for a repository to connect to

The screenshot shows the 'Connect to GitHub' interface. It includes a search bar with 'angrhz' and a dropdown menu showing 'deploy-micorcred-2021' (highlighted by a blue box). A 'Search' button is also visible.

Missing a GitHub organization? [Ensure Heroku Dashboard has team access](#).

angrhz/deploy-micorcred-2021

Connect

# Deployment Model: Langkah-Langkah model deployment

## 11. Pilih Enable Automatic Deploys dan lakukan deploy dengan memilih Deploy Branch

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions [here](#).

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch is always in a deployable state and any tests have passed before you push. [Learn more](#).

Choose a branch to deploy

Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

---

Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more](#).

Choose a branch to deploy

# Deployment Model: Langkah-Langkah model deployment

12. Tunggu setelah proses selesai. Jika sudah pilih **view**

## Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more.](#)

Choose a branch to deploy



Deploy Branch

Receive code from GitHub

Build main 40992dd0

Release phase

Deploy to Heroku

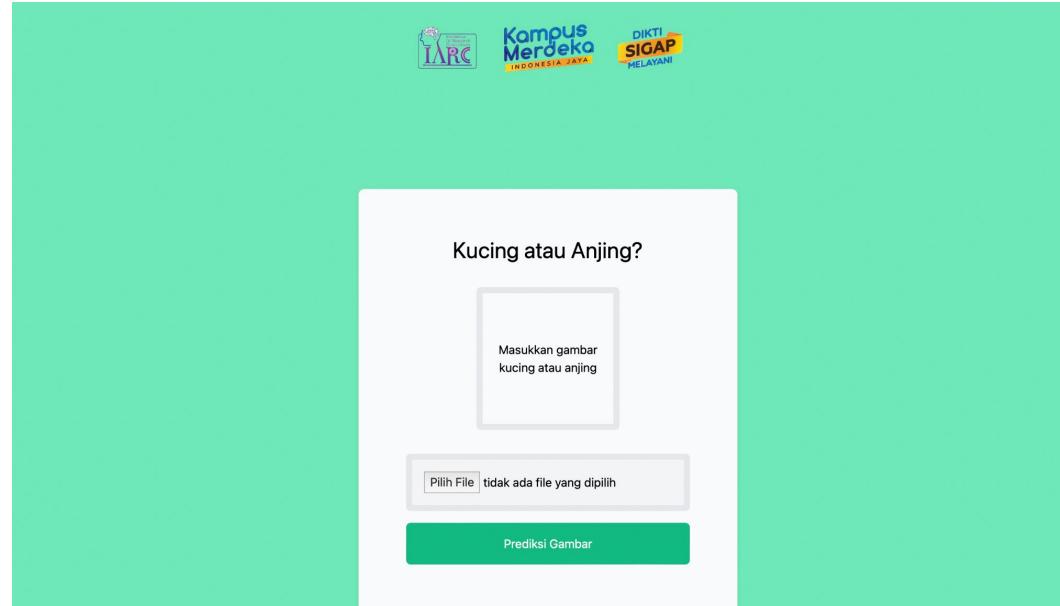


Your app was successfully deployed.

View

# Deployment Model: Langkah-Langkah model deployment

13. Berikut hasilnya : <https://deploy-microcred-2021.herokuapp.com>



# 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.
- 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-linreq>
- <https://github.com/alfanme/dts-deployment-ann>



# Quiz / Tugas

Quiz dapat diakses melalui <https://spadadikti.id/>



*Terima kasih*