

Praktikum Rekayasa Perangkat Lunak

Pertemuan 12

→ Hari ini belajar apa??

DevOps dan CI/CD

continuous integration, continuous delivery

Pola Deployment

pola klasik, pola modern

Deployment Diagram

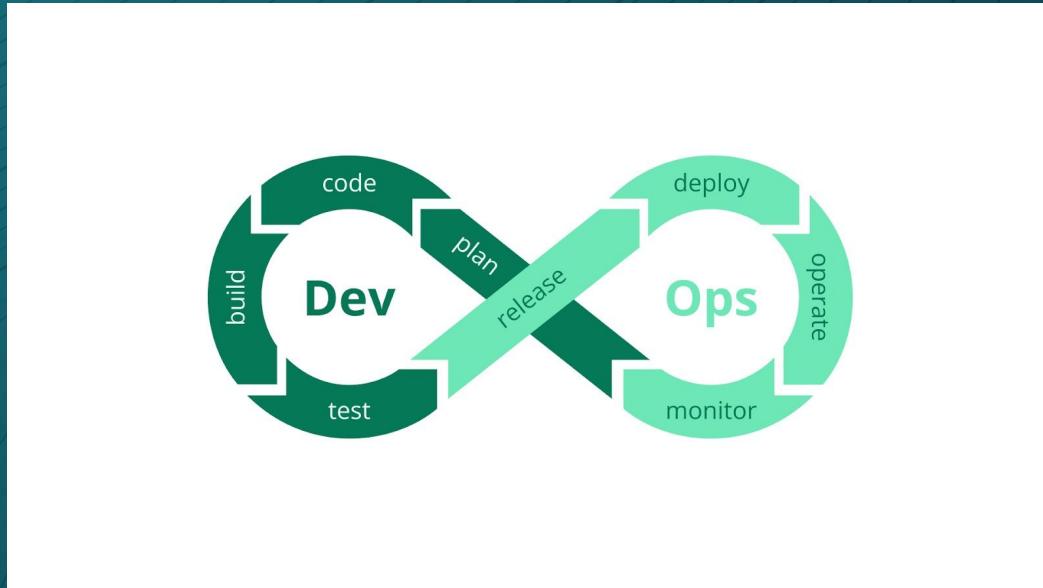
definisi, komponen, contoh



1. DevOps dan CI/CD

Pengenalan DevOps

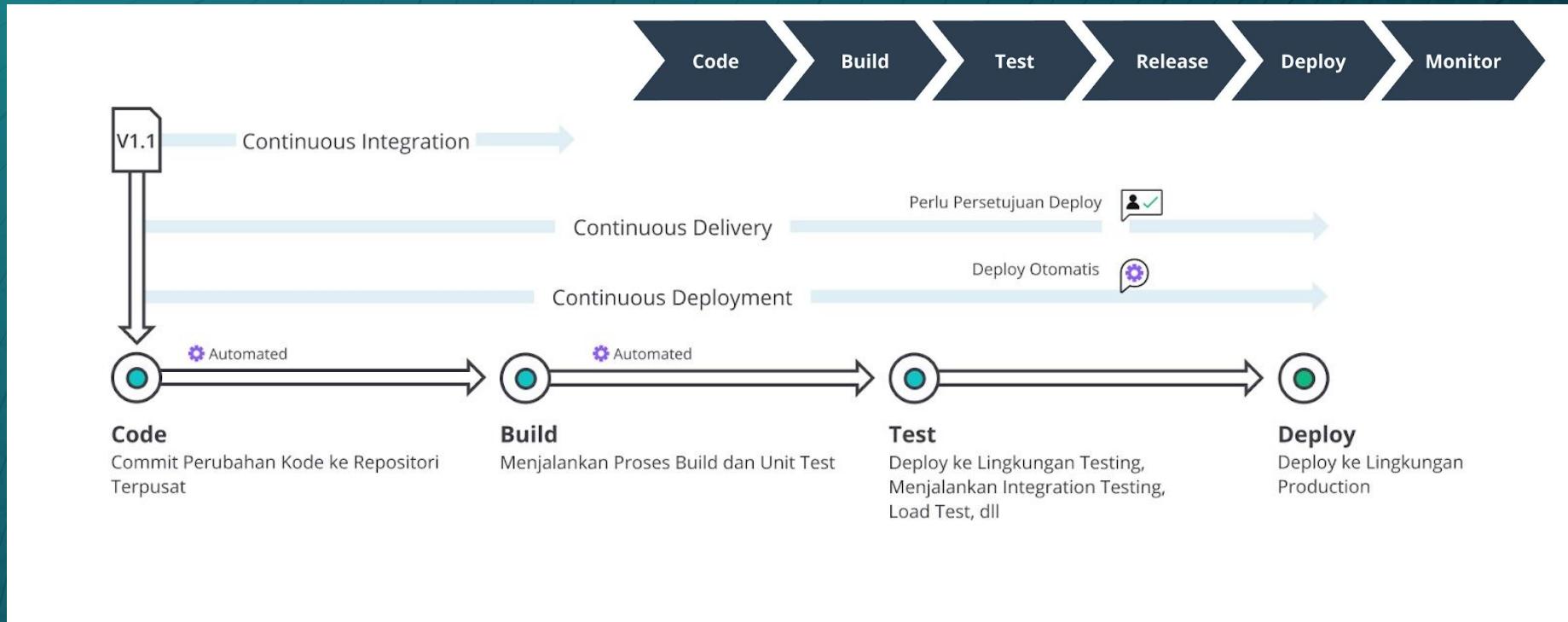
"DevOps is the union of people, process, and products to enable continuous delivery of value to our end users." - Donovan Brown dalam [What is DevOps?](#)



Manfaat DevOps

- Proses rilis aplikasi yang cepat
- Ketangkasan terhadap kebutuhan pasar dan inovasi produk
- Keandalan melalui kepastian kualitas
- Membantu skalabilitas
- Peningkatan kolaborasi
- Meningkatkan keamanan

CI/CD Pipeline



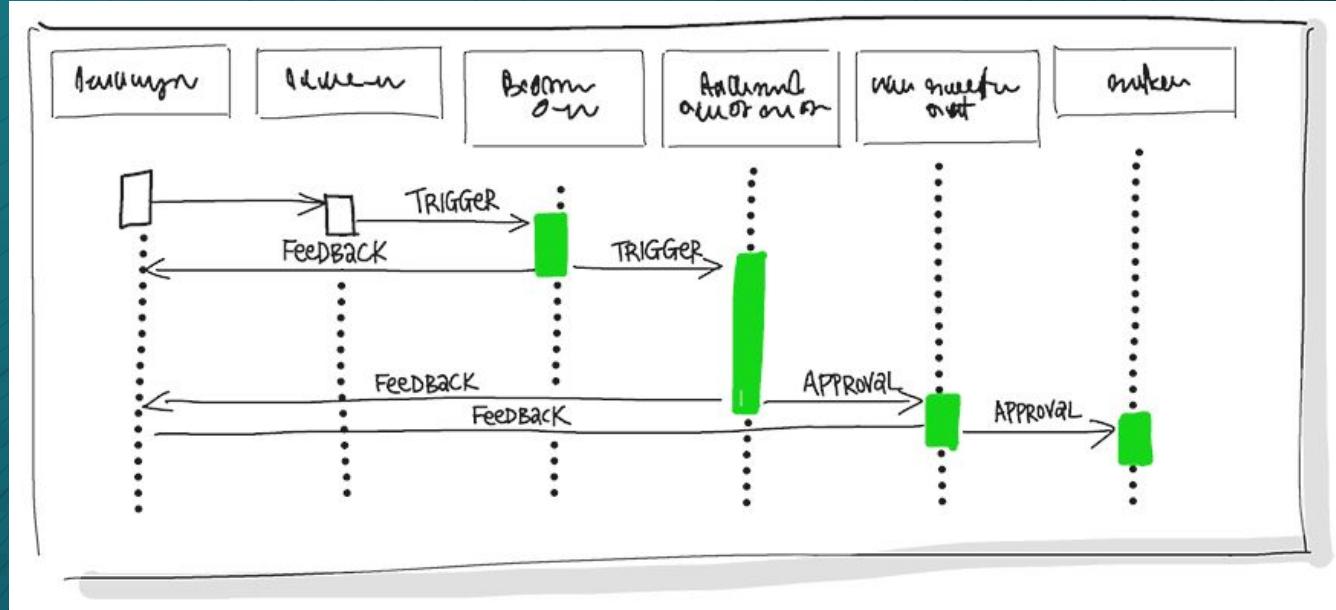
Sumber: Belajar dasar-dasar DevOps - Dicoding Academy

Praktik DevOps



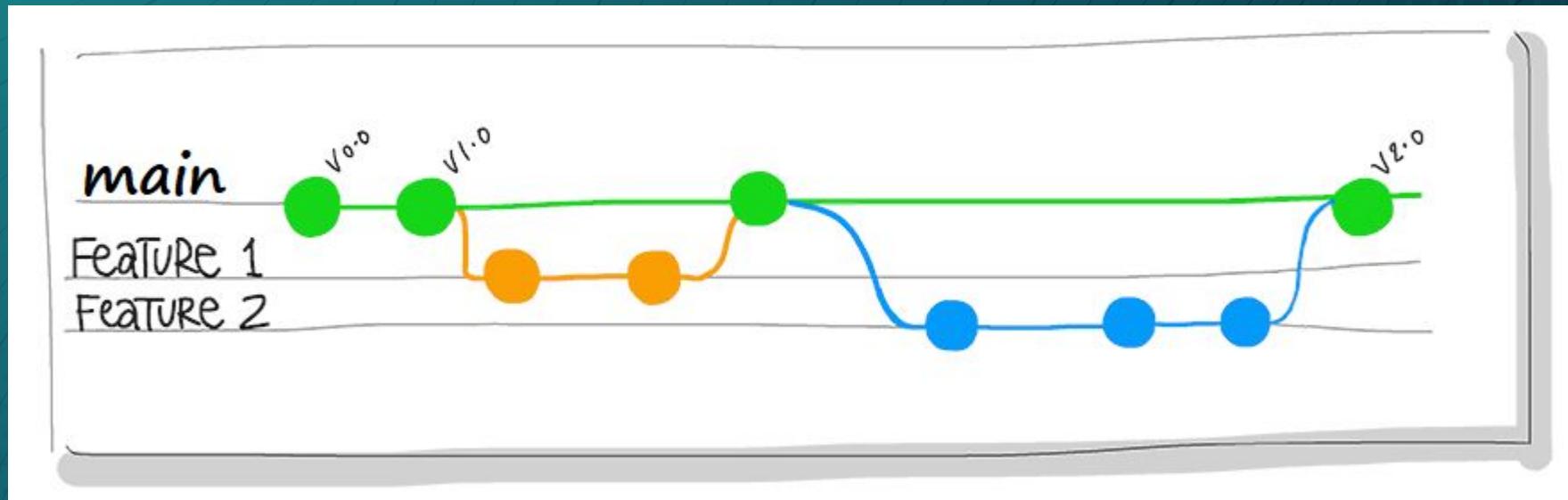
Continuous Integration mendorong penggabungan dan pengujian kode yang berkelanjutan untuk menemukan kecacatan sedini mungkin. Manfaat lainnya adalah mengurangi waktu menyelesaikan *merge issue* dan mempercepat *feedback* ketika *development*.

Praktik DevOps



Continuous Delivery pada lingkungan produksi dan pengujian membantu tim memperbaiki *bug* dengan cepat dan merespons kebutuhan bisnis yang terus berubah.

Praktik DevOps

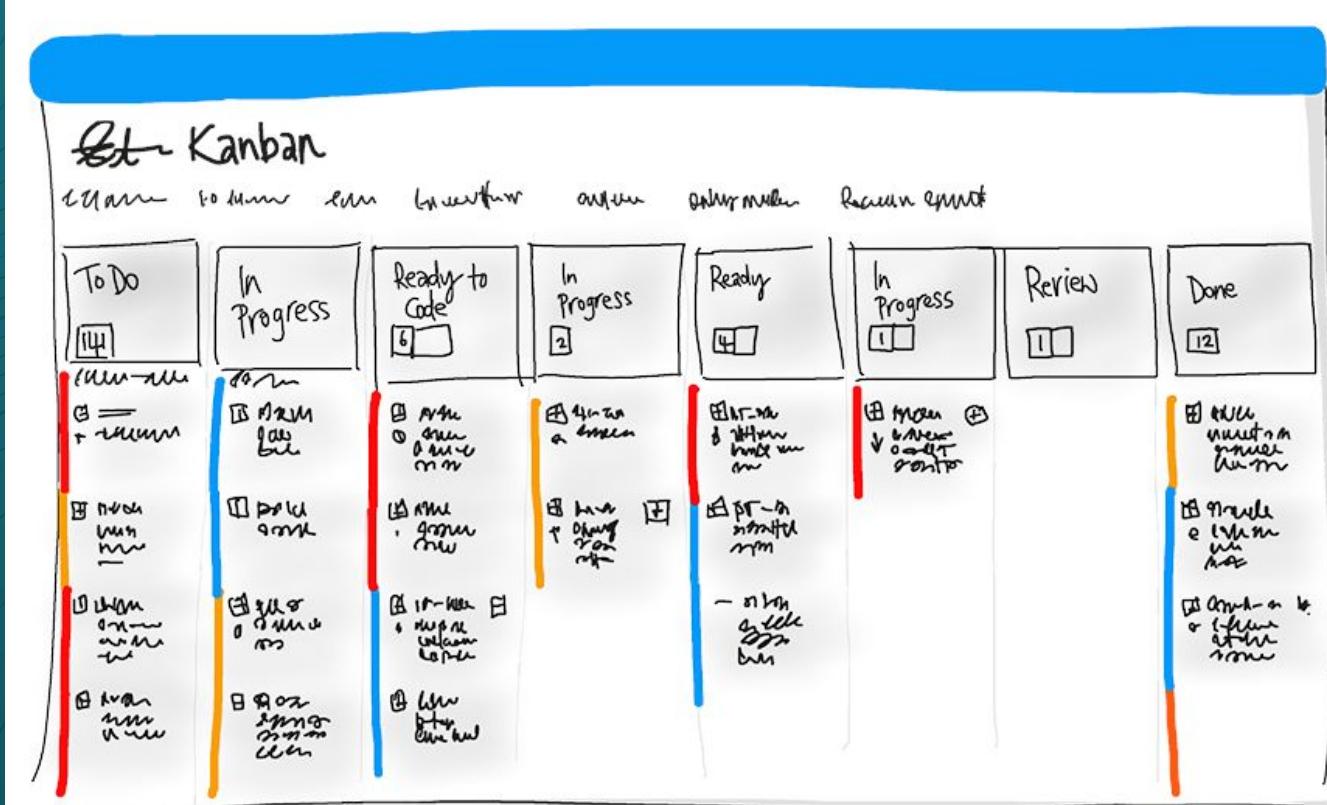


Version Control, dengan repositori berbasis Git memungkinkan tim untuk berkomunikasi secara efektif selama aktivitas *development* harian.

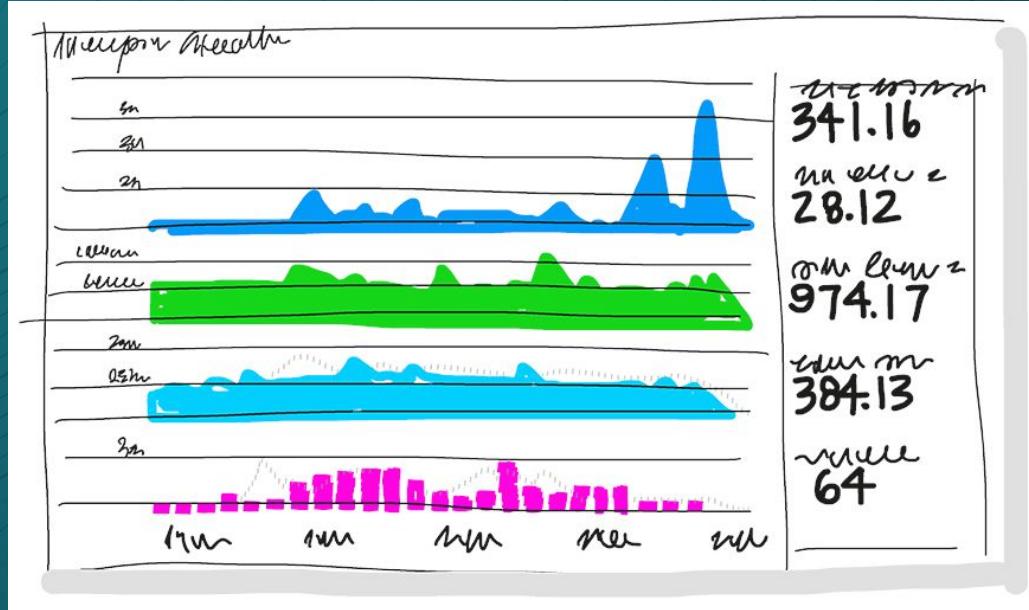
Praktik DevOps

Gunakan *Agile planning* dan *lean technique* untuk:

- Merencanakan dan memisahkan pekerjaan menjadi sprint.
- Mengelola kapasitas tim dan membantu tim beradaptasi dengan cepat terhadap perubahan.
- Definisi DevOps dari **Done** adalah ketika *software* telah mengumpulkan telemetri untuk mencapai tujuan bisnis yang dimaksud.

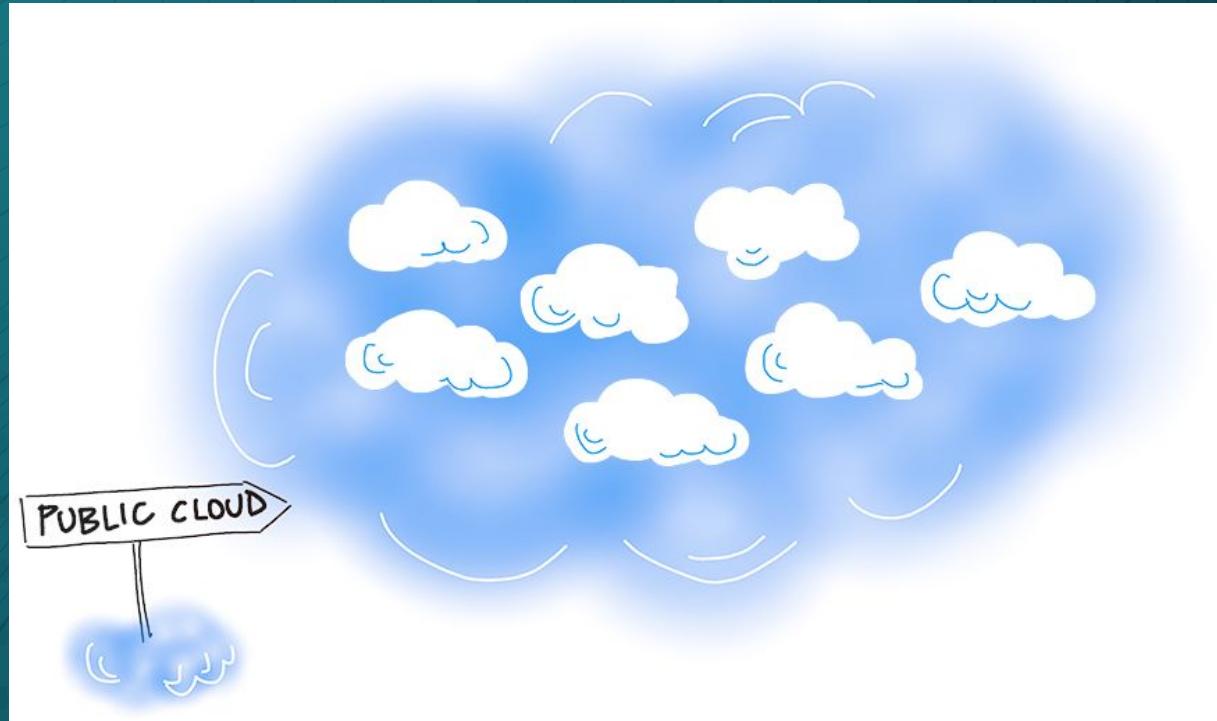


Praktik DevOps



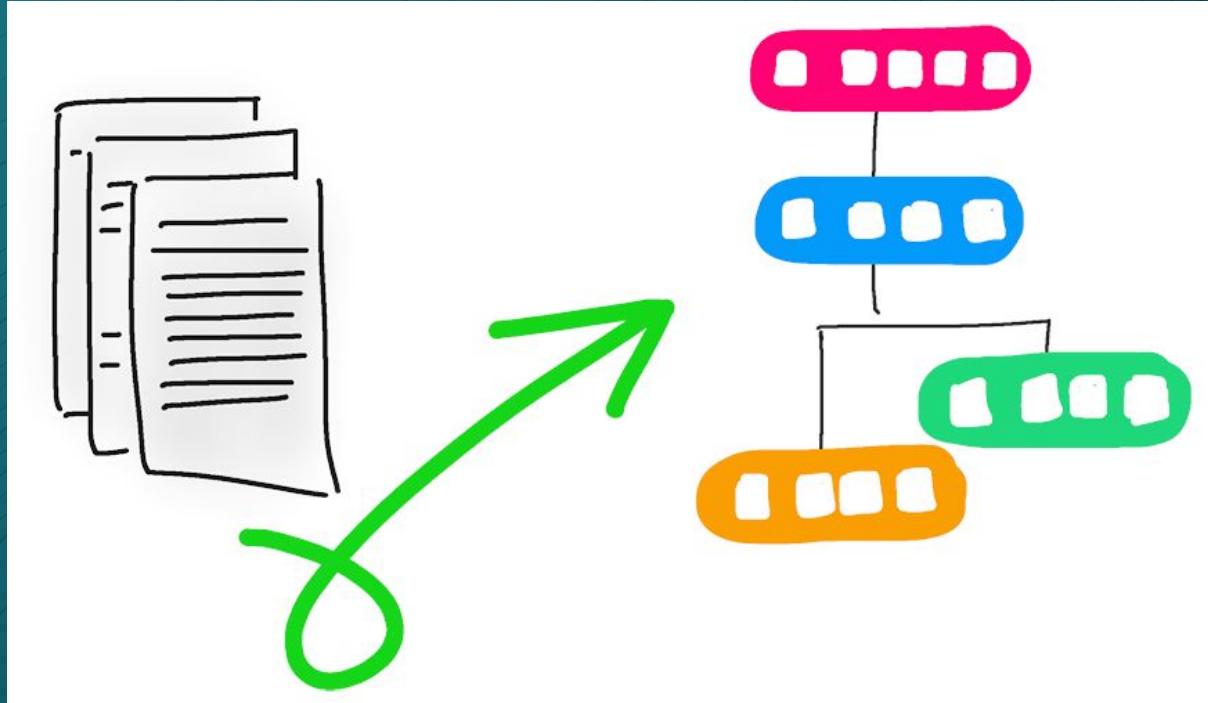
Monitoring dan Logging aplikasi yang sedang berjalan pada lingkungan produksi untuk memperoleh data *health check* dan *usage*. Digunakan untuk membantu tim membuat hipotesis dan dengan cepat membuat strategi berikutnya. Data disimpan dalam berbagai format *logging*.

Praktik DevOps



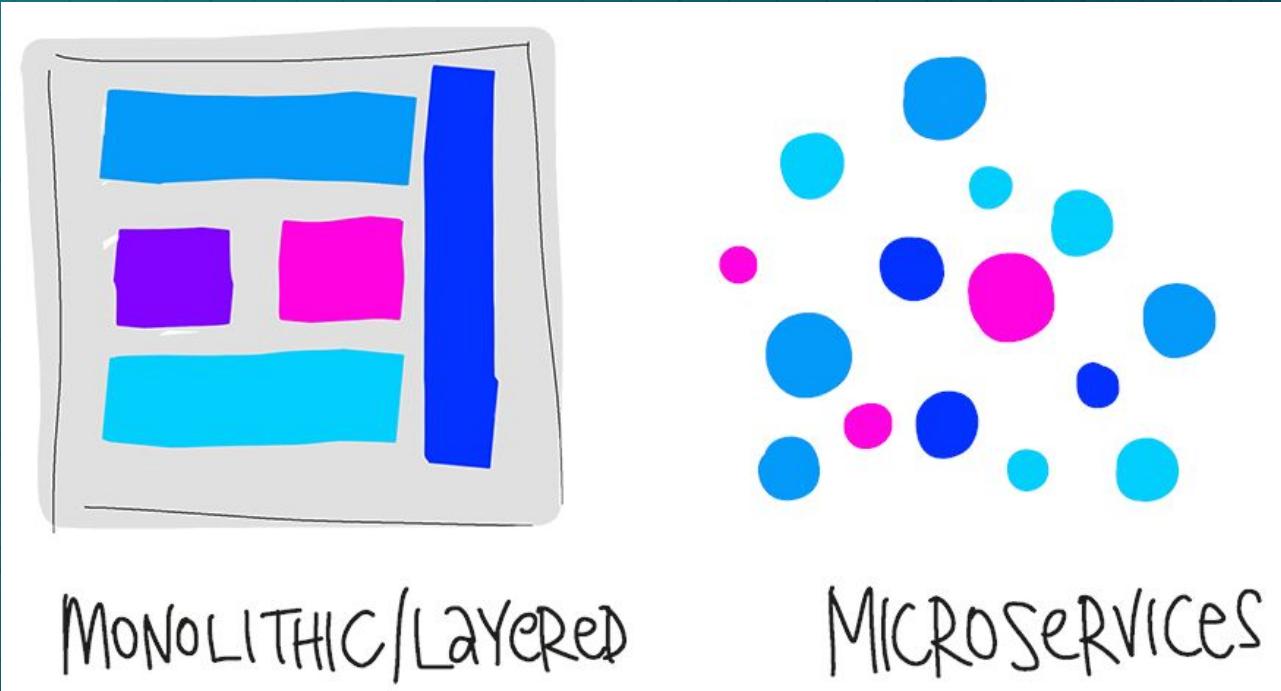
Pemanfaatan *Public* dan *Hybrid Cloud* untuk membantu mengakomodasi infrastruktur dengan mudah.

Praktik DevOps



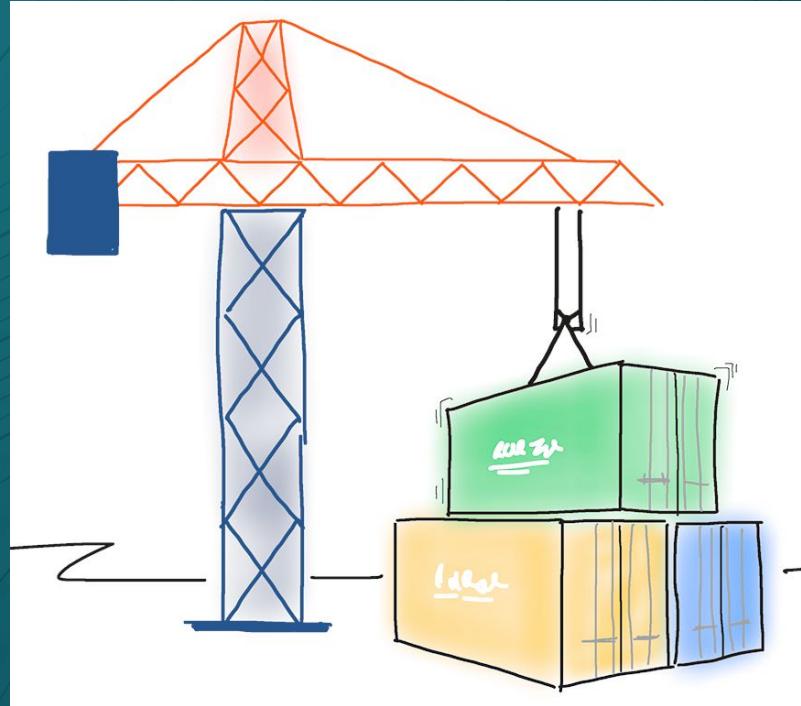
Infrastructure as Code (IaC) memungkinkan otomatisasi dan validasi pembuatan dan penghancuran lingkungan aplikasi yang lebih aman dan stabil.

Praktik DevOps



Gunakan arsitektur *Microservices* untuk mengisolasi *business process* menjadi *service-service* kecil yang *reusable*.

Praktik DevOps



Manfaatkan *container*/ kontainerisasi untuk membungkus aplikasi dengan ringan dan cepat.

Tools DevOps

Komunikasi



Version Control



IaC



Logs



Cloud



Automation



Build



Monitoring



IDE



Test





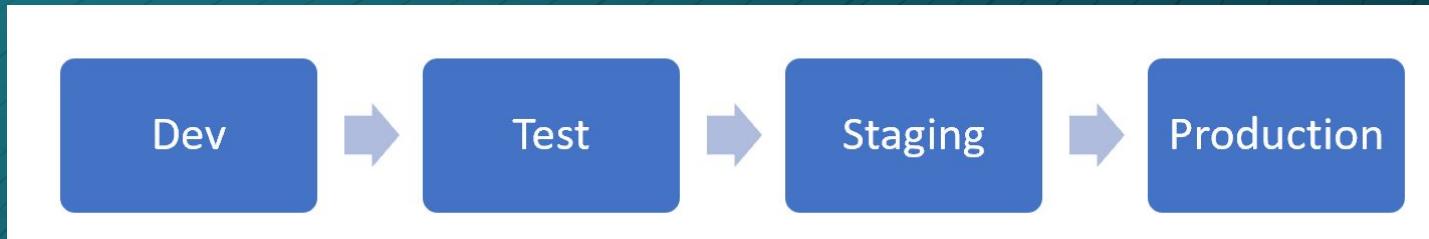
2. Pola Deployment

Fase Deployment

- Fase *deployment* ditujukan untuk memindahkan *software* ke lingkungan produksi agar dapat digunakan oleh *end user*.
- Fase ini memiliki beberapa tahapan, yaitu persiapan dan prosedur *deployment*, *deployment* produk, memindahkan kepemilikan produk, dan menyelesaikan fase *deployment*.
- Terdapat beberapa pola *deployment*, yakni pola tradisional dan pola modern.

Pola Deployment Tradisional

- Pola ini cenderung sederhana karena hanya memindahkan *software* ke tahapan/ lingkungan yang berbeda.



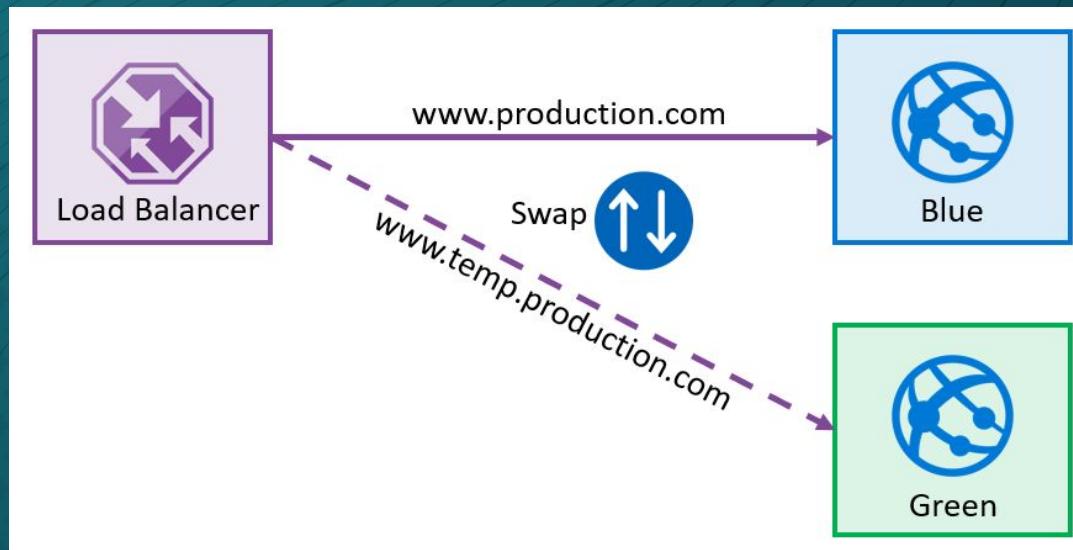
- *Software* berpindah sebagai satu bagian utuh dalam setiap tahapan.
- Dapat menimbulkan banyak resiko, dikarenakan pengujian dan validasi berada di lingkungan non-prod.

Pola Deployment Modern

- Untuk mengantisipasi hal yang tidak diinginkan di lingkungan produksi, diperlukan adanya pengujian setelah *software* dirilis.
- Untuk melakukan hal ini, kita dapat melakukan *deploy* fitur tanpa memperlihatkannya ke semua *user*.
- Beberapa konsep *deployment* modern antara lain:
 - Blue-green deployments.
 - Canary releases.
 - Dark launching.
 - A/B testing.
 - Progressive exposure or ring-based deployment.
 - Feature toggles.

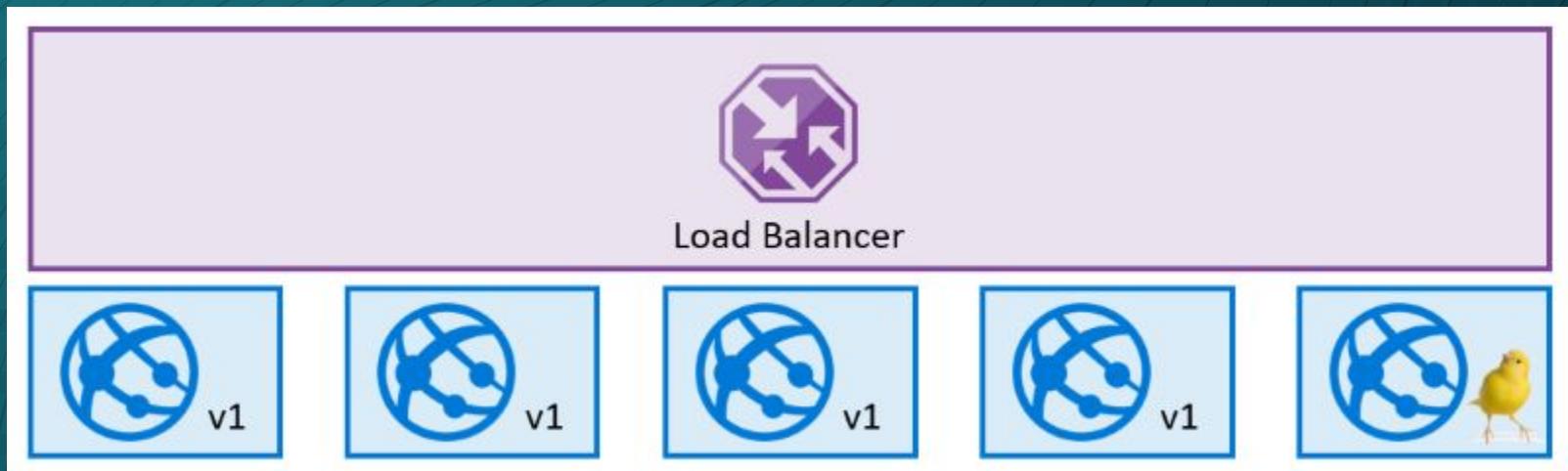
Blue-Green Deployment

- Teknik untuk mengurangi resiko dan *downtime* aplikasi dengan menjalankan dua lingkungan produksi yang identik.
- Hanya satu lingkungan yang *live*. Dilakukan penukaran jika ada versi baru.



Canary Releases

- Dilakukan dengan memperkenalkan fitur baru pada sebagian *user*.
- Berfungsi untuk memonitor tingkah laku fitur baru apabila digunakan serta mengidentifikasi masalah performa atau skalabilitas yang muncul.

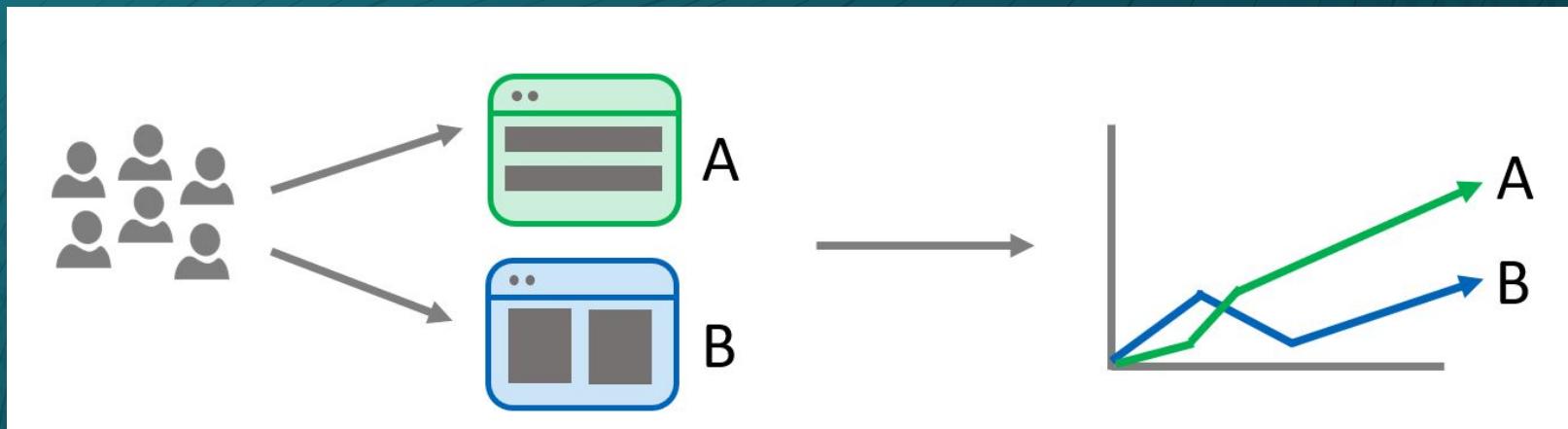


Dark Launching

- Mirip dengan *canary releases*, namun perbedaannya adalah difokuskan pada respons *user* untuk fitur baru di frontend.
- Fitur tidak diberitahu kepada *user*, sehingga dinamakan “dark” launching.

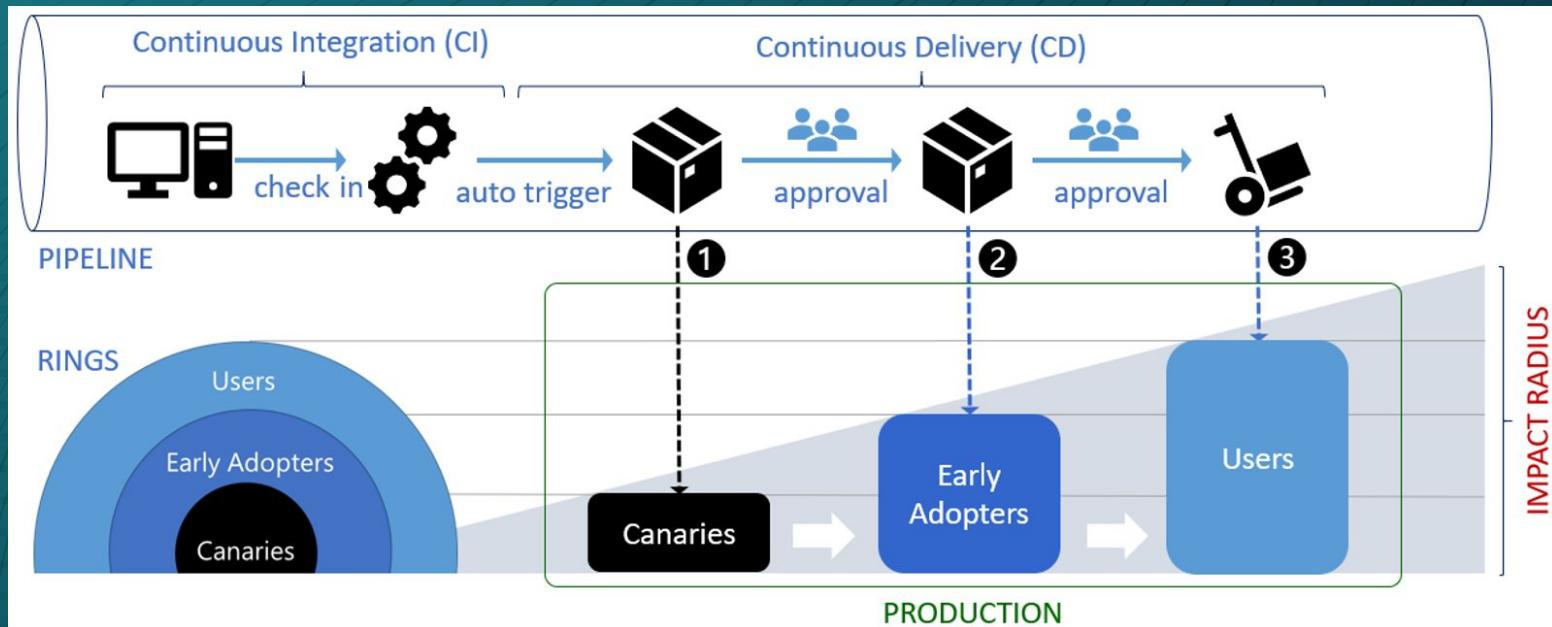
A/B Testing

- Dapat disebut juga *split testing* atau *bucket testing*.
- Dilakukan untuk membandingkan dua versi laman aplikasi satu sama lain.
- Berguna untuk melihat variasi yang menghasilkan *insight* terbaik.



Ring Based Deployment

- Melibatkan *production-first DevOps mindset* untuk mengurangi dampak perilisan pada *user* dengan *deploy* dan validasi secara bertahap.



Feature Toggles/ Feature Flag

- Dilakukan untuk memodifikasi tingkah laku sistem tanpa mengubah kode melalui penggunaan *toggle*.
- Dapat menggunakan *dashboard* khusus untuk mengontrol *toggle*.



Pertimbangan Arsitektur

- Kita dapat memanfaatkan *Continuous Delivery* pada CI/ CD untuk *deployment*, tapi apa pertimbangannya?
 - Apakah *software* berbentuk monolitik atau dibagi menjadi beberapa komponen?
 - Apakah masing-masing bagian dari *software* dapat dirilis secara terpisah?
 - Adakah jaminan jika *software* di-*deploy* berkali-kali dalam satu minggu tetap memiliki kualitas yang baik?
 - Bagaimana pengujian *software* dilakukan?
 - Dapatkah satu atau beberapa versi *software* dijalankan bersamaan?
 - Apa yang harus ditingkatkan jika ingin mengimplementasi *Continuous Delivery*?



3. Deployment Diagram

→ Definisi

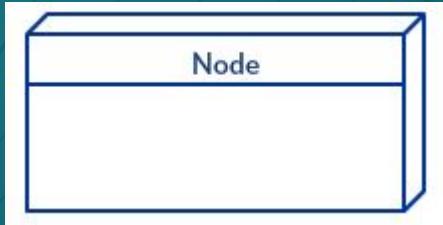
Deployment diagram adalah jenis diagram UML yang menunjukkan arsitektur eksekusi sistem, termasuk *node* seperti untuk lingkungan eksekusi *hardware* atau *software* dan *middleware* yang menghubungkannya.

Untuk menggambarkan *web apps*, komponen *hardware* (mis. server web, server aplikasi, dan server *database*), komponen *software* yakni artefak apa yang berjalan di setiap node (mis. aplikasi web, *database*), dan bagaimana setiap bagian dapat terhubung (mis. dengan JDBC, REST, RMI).

→ Tujuan

- Menunjukkan struktur sistem *run-time*.
- Memodelkan elemen *hardware* dan jalur komunikasinya.
- Digunakan untuk merencanakan arsitektur sistem.
- Berguna untuk mendokumentasikan *deployment* komponen atau *node software*.

→ Komponen

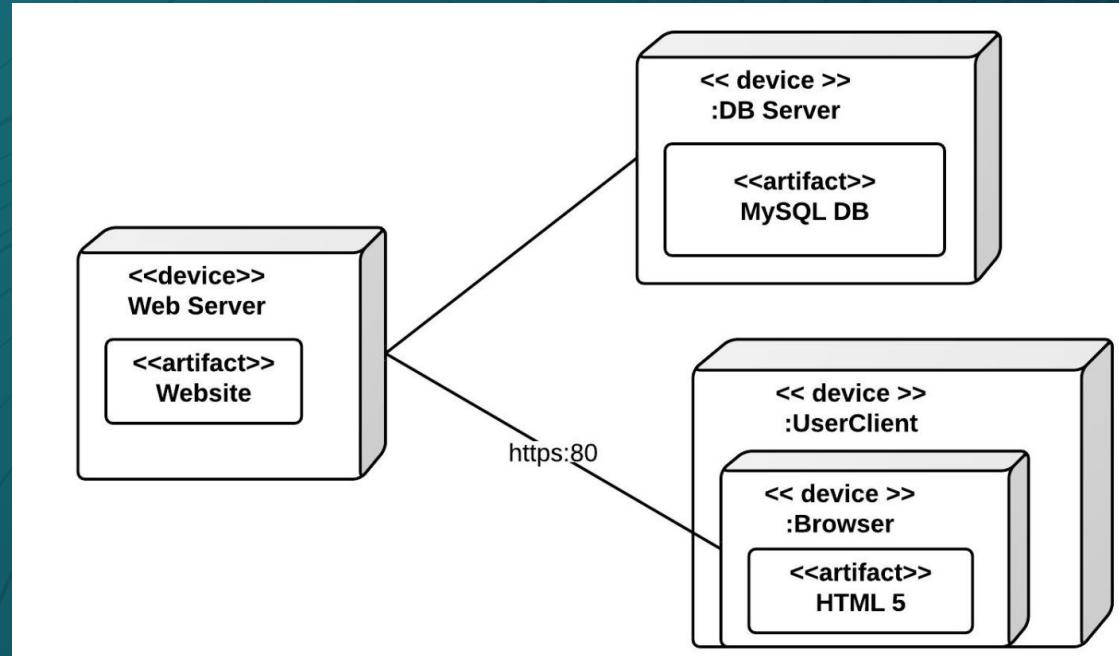


Node adalah entitas fisik yang mengeksekusi satu atau lebih komponen, subsistem, atau hal yang dapat dieksekusi. Node bisa berupa elemen *hardware* atau *software*.

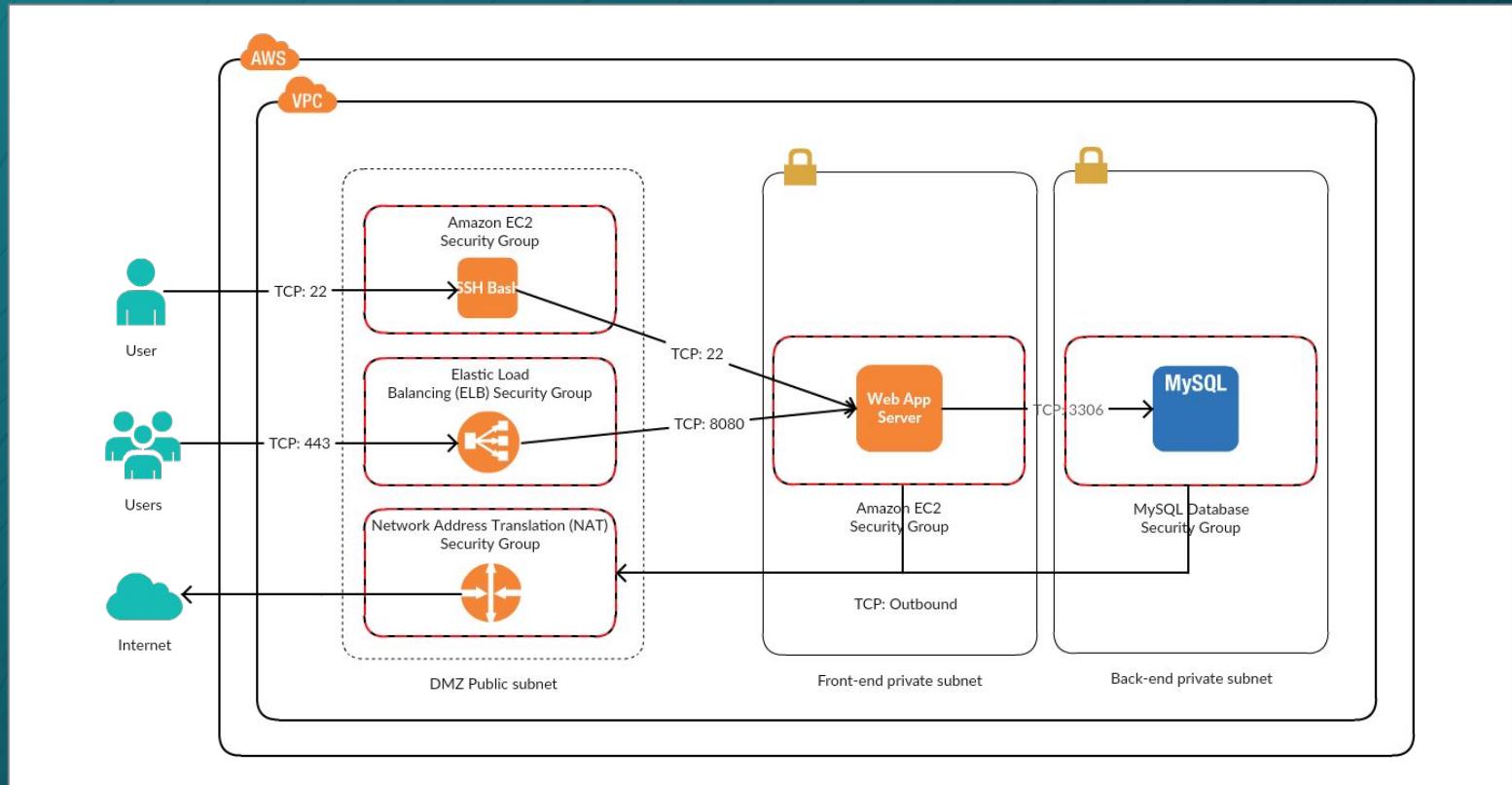


Artefak adalah elemen konkret yang dihasilkan dari proses *development*. Contoh artefak adalah *library*, *archive*, file konfigurasi, file *executable*, dll.

Contoh



Jika melakukan *deployment* ke cloud, kita dapat menggunakan diagram arsitektur cloud.





Tugas

tugas segan ngganggur tak mau

Tugas

Buat **deployment diagram** aplikasi sesuai proposal kelompok yang telah dibuat dan berikan penjelasannya. Jika menggunakan infrastruktur cloud, bisa membuat diagram arsitektur cloudnya. Kumpulkan di Google Classroom

Format:

DeploymentDiagram_namaprojek.pdf

Deadline : H-1 Praktikum Berikutnya, 23.59 (**KUMPUL PERWAKILAN SAJA!**)

Terima kasih