BAB III

DevOps

A. Pengetahuan yang diperlukan dalam menerapkan DevOps

1. Memahami konsep dasar Lean dan Agile dalam DevOps
2. Memahami konsep dasar penggunaan version control
3. Mengidentifikasi penggunaan CI/CD
4. Memahami pentingnya IaC

B. Ketrampilan yang diperlukan dalam menerapkan DevOps

1. Keterampilan komunikasi
2. Mengeksekusi source code
3. Keterampilan bekerjasama dengan tim
4. Keterampilan berfikir kritis
5. Keterampilan mengidentifikasi masalah

C. Sikap yang diperlukan dalam menerapkan DevOps

1. Disiplin
2. Teliti
3. Objektif
4. Bertanggung jawab
5. Kreatif
6. Keterbukaan
7. *Critical Thinking*

D. DevOps

1. Devops: Fondasi Dasar *Lean* dan *Agile*

A. *Lean Management*

Lean adalah suatu upaya terus menerus (*continuous improvement effort*) untuk menghilangkan pemborosan (*waste*), meningkatkan nilai tambah (*value added*) produk (barang dan/ jasa) dan memberikan nilai kepada pelanggan *(customer value*) (Gaspersz, 2008). Sedangkan pendapat yang lain mengatakan bahwa prinsip

dari *lean thinking* adalah mencari cara untuk proses penciptaan nilai dengan urutan terbaik yang dimungkinkan, menyusun aktivitas ini tanpa interupsi, dan menjelaskan secara lebih dan lebih efektif (Hines & Tylor, 2000).

Sebelum Anda mulai dengan prinsip dasar Lean, Anda perlu menyadari bahwa metodologi Lean adalah tentang terus meningkatkan proses kerja, tujuan, dan orang-orang.

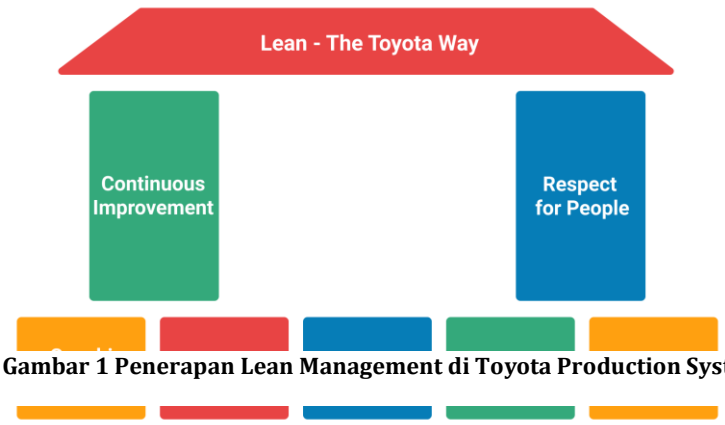
Metodologi Lean mengandalkan 3 gagasan yang sangat sederhana:

- Memberikan nilai dari perspektif pelanggan Anda
- Hilangkan pemborosan (hal-hal yang tidak memberi nilai pada produk akhir)
- Perbaikan terus-menerus

Manajemen Lean mendorong tanggung jawab bersama dan kepemimpinan bersama. Inilah mengapa dua pilar utama metodologi Lean adalah:

- Menghormati orang lain
- Perbaikan berkelanjutan

Bagaimanapun, ide atau inisiatif yang baik dapat lahir di semua tingkat hierarki dan Lean mempercayai orang-orang yang melakukan pekerjaan untuk mengatakan bagaimana hal itu harus dilakukan. Saat ini, manajemen Lean adalah konsep yang diadopsi secara luas di berbagai industri. Namun, itu sebenarnya berasal dari Toyota Production System, yang didirikan sekitar 70 tahun yang lalu.



Gambar 1 Penerapan Lean Management di Toyota Production System

Terdapat lima prinsip dasar Lean, yaitu:

1. Identifikasi Nilai (*Identify Value*)

   Apa yang berusaha dilakukan oleh setiap perusahaan? Jawabnnya adalah menawarkan produk / layanan yang siap dibayar oleh pelanggan. Untuk melakukannya, perusahaan perlu menambahkan nilai yang ditentukan oleh kebutuhan pelanggannya.

   Nilainya terletak pada masalah yang Anda coba selesaikan untuk pelanggan. Lebih khusus lagi pada bagian solusi yang bersedia dibayar oleh pelanggan Anda secara aktif. Setiap aktivitas atau proses lain yang tidak memberi nilai pada produk akhir dianggap pemborosan.

   Jadi, Anda harus terlebih dahulu mengidentifikasi nilai yang ingin Anda berikan dan kemudian melanjutkan ke langkah berikutnya.

2. Pemetaan Aliran Nilai (*Map the Value Stream*)

   Ini adalah titik di mana Anda benar-benar perlu memetakan alur kerja perusahaan Anda. Hal ini harus mencakup semua tindakan dan orang-orang yang terlibat dalam proses pengiriman produk akhir kepada pelanggan. Dengan demikian, Anda akan dapat mengidentifikasi bagian mana dari proses yang tidak bernilai.

   Menerapkan prinsip Lean dari pemetaan aliran nilai akan menunjukkan kepada Anda di mana nilai dihasilkan dan dalam proporsi apa proses yang berbeda menghasilkan atau tidak menghasilkan nilai. Ketika aliran nilai Anda dipetakan, akan lebih mudah bagi Anda untuk melihat proses mana yang dimiliki oleh tim dan siapa yang bertanggung jawab untuk mengukur, mengevaluasi, dan meningkatkan proses itu. Gambaran besar ini akan memungkinkan Anda untuk mendeteksi langkah-langkah yang tidak membawa nilai dan menghilangkannya.

3. Buat Alur Kerja Berkelanjutan (*Create Flow*)

   Setelah menguasai aliran nilai, Anda perlu memastikan bahwa alur kerja setiap tim tetap lancar. Ingatlah bahwa ini mungkin membutuhkan waktu beberapa saat. Mengembangkan produk/layanan sering kali mencakup kerja tim lintas fungsi. Kemacetan dan gangguan dapat muncul kapan saja. Namun, dengan memecah pekerjaan menjadi beberapa kelompok kecil dan

memvisualisasikan alur kerja, Anda akan dapat dengan mudah mendeteksi dan menghapus penghalang pandang proses.

4. *Establish Pull*

   Memiliki alur kerja yang stabil adalah jaminan bahwa tim Anda dapat melakukan tugas kerja lebih cepat dengan sedikit usaha. Namun, untuk mengamankan alur kerja yang stabil, pastikan untuk membuat *Establish Pull.* Dalam sistem seperti itu, pekerjaan ditarik hanya jika ada permintaan. Hal ini memungkinkan Anda untuk mengoptimalkan kapasitas sumber daya dan mengirimkan produk / layanan hanya jika memang benar-benar dibutuhkan.

   Kita ambil sebuah restoran sebagai contoh. Anda pergi ke sana dan memesan pizza. Tukang roti menarik pesanan Anda dan mulai membuat pizza. Dia tidak menyiapkan banyak hidangan sebelumnya karena sebenarnya tidak ada permintaan dan banyak hidangan ini bisa menjadi pemborosan sumber daya.

5. Perbaikan Berkelanjutan (*Continuous Improvement*)

   Setelah melalui semua langkah sebelumnya, Anda sudah membangun sistem manajemen Lean Anda. Namun, jangan lupa untuk memperhatikan langkah terakhir ini, mungkin ini yang paling penting. Ingat masalah dapat terjadi di salah satu langkah sebelumnya. Inilah mengapa Anda perlu memastikan bahwa karyawan di setiap level terlibat dalam setiap proses yang terus berjalan (peningkatan berkelanjutan).

   Ada beberapa teknik berbeda untuk mendorong peningkatan berkelanjutan. Misalnya, setiap tim melakukan pertemuan harian untuk membahas apa yang telah dilakukan, apa yang perlu dilakukan dan kemungkinan hambatan. Hal tersebut merupakan cara mudah untuk memproses peningkatan setiap hari.

Gambar 2 Prinsip dasar Leans

Pelajari topik tentang lean managemen ini selengkapnya di https://kanbanize.com/lean-management/what-is-lean-management

B. Agile

Agile adalah pendekatan berulang untuk manajemen proyek dan pengembangan perangkat lunak yang membantu tim memberikan nilai kepada pelanggan mereka lebih cepat dan dengan lebih sedikit kesalahan. Sementara pendekatan *waterfall*, memiliki satu disiplin ilmu yang berkontribusi pada proyek, kemudian "membuangnya" kepada kontributor berikutnya. Komunikasi terbuka, kolaborasi, adaptasi, dan kepercayaan di antara anggota tim merupakan inti dari *agile*. *Agile* tidak ditentukan oleh serangkaian upacara atau teknik pengembangan khusus. *Agile* adalah sekelompok metodologi yang menunjukkan komitmen terhadap siklus umpan balik yang ketat dan peningkatan berkelanjutan.

Mengapa memilih *agile*? Tim memilih *agile* sehingga mereka dapat menanggapi perubahan di pasar atau umpan balik dari pelanggan dengan cepat tanpa menggagalkan rencana selama setahun. Perencanaan dan pengiriman "Cukup" dalam peningkatan kecil dan sering memungkinkan tim Anda mengumpulkan umpan balik pada setiap perubahan dan mengintegrasikannya ke dalam rencana masa depan dengan biaya minimal.

Tapi ini bukan hanya permainan angka. Pertama dan terpenting, hal ini tentang manusia. Seperti yang dijelaskan oleh Agile Manifesto, interaksi manusia yang otentik lebih penting daripada proses yang kaku. Berkolaborasi dengan pelanggan dan rekan tim lebih penting daripada pengaturan yang telah ditentukan

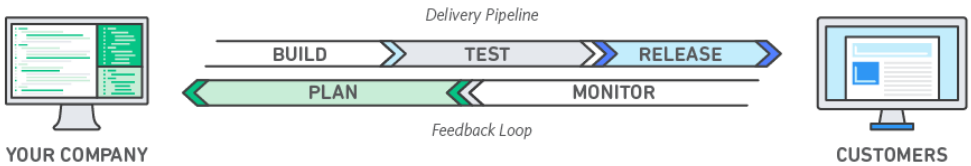
sebelumnya. Dan memberikan solusi yang berhasil untuk masalah pelanggan lebih penting daripada dokumentasi yang sangat detail.

Tim *agile* bersatu di bawah visi bersama, lalu mewujudkannya dengan cara yang mereka tahu adalah yang terbaik. Setiap tim menetapkan standar mereka sendiri untuk kualitas, kegunaan, dan kelengkapan. "Definisi selesai" mereka kemudian menginformasikan seberapa cepat mereka akan menyelesaikan pekerjaan tersebut. Meskipun pada awalnya menakutkan, para pemimpin perusahaan menemukan bahwa ketika mereka menaruh kepercayaan mereka pada tim *agile*, tim tersebut merasakan rasa kepemilikan yang lebih besar dan bangkit untuk memenuhi (atau melampaui) ekspektasi manajemen.

Pelajari topik tentang *agile* secara lengkap di https://www.atlassian.com/agile

C. DevOps

DevOps adalah kombinasi dari filosofi budaya, praktik, dan alat yang meningkatkan kemampuan organisasi untuk memberikan aplikasi dan layanan dengan kecepatan tinggi, mengembangkan dan meningkatkan produk dengan kecepatan yang lebih cepat daripada organisasi yang menggunakan pengembangan perangkat lunak tradisional dan proses manajemen infrastruktur. Kecepatan ini memungkinkan organisasi untuk melayani pelanggan mereka dengan lebih baik dan bersaing dengan lebih efektif di pasar.



Gambar 3 Ilustrasi DevOps

Berikut adalah manfaat menggunakan DevOps:

1. *Speed*

   Bergerak dengan kecepatan tinggi sehingga Anda dapat berinovasi untuk pelanggan lebih cepat, beradaptasi dengan perubahan pasar yang lebih baik, dan tumbuh lebih efisien dalam mendorong hasil bisnis. Model DevOps memungkinkan pengembang dan tim operasi Anda untuk mencapai hasil ini.

2. *Rapid Delivery*

   Tingkatkan frekuensi dan kecepatan rilis sehingga Anda dapat berinovasi dan meningkatkan produk Anda lebih cepat. Semakin cepat Anda merilis fitur baru dan memperbaiki bug, semakin cepat Anda dapat merespons kebutuhan pelanggan dan membangun keunggulan kompetitif. Integrasi berkelanjutan dan pengiriman berkelanjutan adalah praktik yang mengotomatiskan proses rilis perangkat lunak, mulai dari pembuatan hingga penerapan

3. Reliability

   Pastikan kualitas pembaruan aplikasi dan perubahan infrastruktur sehingga Anda dapat mengirimkan dengan lebih cepat dengan tetap menjaga pengalaman positif bagi pengguna. Gunakan praktik seperti integrasi berkelanjutan dan pengiriman berkelanjutan untuk menguji bahwa setiap perubahan berfungsi dengan aman. Praktik pemantauan dan pencatatan membantu Anda tetap mendapatkan informasi tentang kinerja secara waktu nyata.

4. Scale

   Operasikan dan kelola infrastruktur proses pengembangan Anda dalam skala besar. Otomatisasi dan konsistensi membantu Anda mengelola sistem yang kompleks atau berubah secara efisien dan dengan risiko yang lebih rendah.

5. Improved Collaboration

   Bangun tim yang lebih efektif di bawah model budaya DevOps, yang menekankan nilai-nilai seperti kepemilikan dan akuntabilitas. Pengembang dan tim operasi berkolaborasi secara erat, berbagi banyak tanggung jawab, dan menggabungkan alur kerja mereka. Hal ini mengurangi inefisiensi dan menghemat waktu.

Selengkapnya: https://aws.amazon.com/devops/what-is-devops/

2. DevOps: Version Control

Version control adalah sebuah sistem yang merekam perubahan-perubahan dari sebuah berkas atau sekumpulan berkas dari waktu ke waktu sehingga Anda dapat menilik kembali versi khusus suatu saat nanti (Scott & Ben, 2014). Terdapat 3 bentuk version control:

1. Sistem Version Control Lokal

   Sebelumnya metode version control yang banyak dipilih oleh orang-orang adalah dengan menyalin berkas-berkas ke direktori lain (mungkin direktori yang diberi catatan waktu, jika mereka cerdas). Pendekatan ini sangat umum karena ini sangat sederhana, namun ini juga sangat rentan terkena *error*. Mudah sekali untuk lupa pada direktori mana Anda sedang berada dan menulis ke berkas.

   Untuk menghadapi hal ini, dahulu para programmer mengembangkan VCS lokal yang memiliki database sederhana yang menyimpan semua perubahan pada berkas pada revision control.

2. Sistem Version Control Terpusat

   Masalah besar selanjutnya yang dihadapi programmer adalah bahwa mereka butuh bekerja bersama dengan para pengembang pada sistem lain. Untuk menangani masalah ini, Centralized Version Control System (CVCS) dikembangkan. Sistem-sistem ini, seperti CVS, Subversion, dan Perforce, memiliki sebuah server tunggal yang berisi semua berkas-berkas yang telah diberi versi, dan beberapa klien yang melakukan check out pada berkas-berkas dari pusat tersebut. Selama bertahun-tahun, hal ini telah menjadi standar untuk version control.

   Akan tetapi, pengaturan ini juga memiliki beberapa kekurangan. Yang paling jelas adalah satu titik kegagalan yang diwakili oleh server terpusat. Jika server tersebut sedang *down* selama satu jam, maka selama itu tidak ada orang yang dapat bekerja bersama atau menyimpan perubahan yang telah diberi versi terhadap apapun yang sedang mereka kerjakan. Jika hard disk dari database pusat menjadi *corrupted*, dan cadangan yang memadai belum tersimpan, Anda akan kehilangan seluruh riwayat dari proyek kecuali setiap *snapshot* yang dimiliki oleh orang-orang pada mesin lokal mereka.

3. Sistem Version Control Tersebar

   Di sinilah *Distributed Version Control System* (DVCS) masuk. Pada DVCS (seperti Git), para klien tidak hanya melakukan check out pada snapshot terakhir dari berkas: mereka mencerminkan sepenuhnya repository tersebut. Dan juga, jika ada salah satu server yang mati, dan sistem-sistem ini bekerja bersama melalui server itu, setiap repository milik klien dapat disalin kembali ke server untuk memulihkannya. Setiap check out benar-benar cadangan penuh dari semua data. Sistem ini mampu menangani beberapa remote repository yang dapat mereka kerjakan dengan baik, sehingga Anda dapat bekerja bersama dengan beberapa kelompok orang yang berbeda dengan cara yang berbeda secara bersamaan dalam proyek yang sama.

3. DevOps: CI/CD

Continuous integration (CI) adalah praktik rekayasa perangkat lunak di mana anggota tim mengintegrasikan pekerjaan mereka dengan frekuensi yang semakin meningkat. Sesuai dengan praktik CI, tim berusaha untuk mengintegrasikan setidaknya setiap hari dan bahkan setiap jam, mendekati integrasi yang terjadi secara terus menerus.

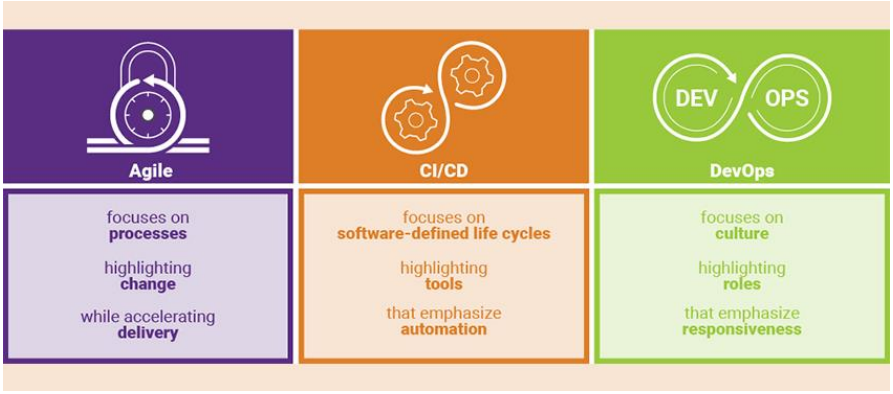
Secara historis, integrasi telah menjadi aktivitas rekayasa yang mahal. Jadi, untuk menghindari kekacauan, CI menekankan alat otomasi yang mendorong pembuatan dan pengujian, yang pada akhirnya berfokus pada pencapaian siklus hidup yang ditentukan perangkat lunak. Jika CI berhasil, upaya membangun dan integrasi berhenti, dan tim dapat mendeteksi kesalahan integrasi secepat mungkin.

Continuous delivery (CD) adalah upaya untuk mengemas dan menyebarkan CI yang akan dibangun dan diuji. Tim yang mempraktikkan CD dapat membangun, mengonfigurasi, dan mengemas perangkat lunak serta mengatur penerapannya sedemikian rupa sehingga dapat dirilis ke produksi dengan cara yang ditentukan perangkat lunak (berbiaya rendah, otomatisasi tinggi) kapan saja.

Praktik CI/CD yang berfungsi tinggi secara langsung memfasilitasi pengembangan agile karena perubahan perangkat lebih sering. Hasilnya, pelanggan memiliki lebih banyak kesempatan untuk mengalami dan memberikan umpan balik tentang perubahan.

Untuk mempermudah, berikut adalah cara cepat dan mudah untuk melihat perbedaan antara CI/CD, Agile dan DevOps:

- Agile berfokus pada proses yang menyoroti perubahan sambil mempercepat pengiriman.
- CI/CD berfokus pada siklus hidup perangkat dalam menyoroti alat yang menekankan otomatisasi.
- DevOps berfokus pada peran budaya yang menekankan daya tanggap.

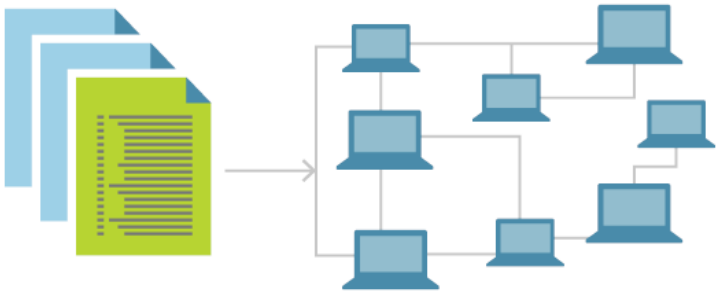


Gambar 4 Perbedaan Agile, CI/CD dan DevOps

Selengkapnya:https://www.synopsys.com/blogs/software-security/agile-cicd-devops-difference/

4. DevOps: Infrastructure as Codes

Infrastructure as Code (IaC) adalah pengelolaan infrastruktur (jaringan, mesin virtual, penyeimbang beban, dan topologi koneksi) dalam model deskriptif, menggunakan versi yang sama seperti yang digunakan tim DevOps untuk *soyrce code*. Seperti prinsip bahwa *source code* yang sama menghasilkan biner yang sama, model IaC menghasilkan lingkungan yang sama setiap kali diterapkan. IaC adalah praktik DevOps utama dan digunakan bersama dengan *continuous delivery.*



Gambar 5 Ilustrasi pemanfaatan IaC

IaC berevolusi untuk memecahkan masalah penyimpangan *environtment* dalam *pipeline.* Tanpa IaC, tim harus memelihara pengaturan *environment* secara individu. Seiring waktu, setiap lingkungan menjadi *snowfake*, yaitu konfigurasi unik yang tidak dapat direproduksi secara otomatis. Inkonsistensi antar lingkungan menyebabkan masalah selama penerapan. Dengan *snowfake*, administrasi dan pemeliharaan infrastruktur melibatkan proses manual yang sulit dilacak dan berkontribusi pada kesalahan.

Idempotence adalah prinsip IaC. *Idempotence* adalah properti di mana perintah penerapan selalu menetapkan *environment* target ke dalam konfigurasi yang sama, terlepas dari status awal *environment*. *Idempotence* dicapai baik dengan mengonfigurasi target yang ada secara otomatis atau dengan membuang target yang ada dan menciptakan kembali *environment* baru.

Karenanya, dengan IaC, tim membuat perubahan pada deskripsi *environtment* dan versi model konfigurasi, yang biasanya dalam format kode yang terdokumentasi dengan baik seperti JSON. Pipeline menjalankan model untuk mengonfigurasi lingkungan target. Jika tim perlu melakukan perubahan, mereka mengedit sumbernya, bukan targetnya.

IaC memungkinkan tim DevOps menguji aplikasi di *environtment* seperti produksi di awal siklus pengembangan. Tim ini berharap untuk menyediakan beberapa *environtment* pengujian yang andal dan sesuai permintaan. IaC juga dapat divalidasi dan diuji untuk mencegah masalah *deployment*. Pada saat yang sama, cloud secara dinamis menyediakan dan menghapus lingkungan berdasarkan definisi IaC.

Tim yang menerapkan IaC dapat memberikan lingkungan yang stabil dengan cepat dan dalam skala besar. Tim menghindari konfigurasi *environment* manual dan menegakkan konsistensi dengan merepresentasikan status *environment* yang diinginkan melalui *source code*. Penerapan infrastruktur dengan IaC dapat diulang dan mencegah masalah waktu proses yang disebabkan oleh penyimpangan konfigurasi atau ketergantungan yang hilang. Tim DevOps dapat bekerja sama dengan serangkaian praktik dan alat terpadu untuk mengirimkan aplikasi dan infrastruktur pendukungnya dengan cepat, andal, dan dalam skala besar.

Selengkapnya:https://docs.microsoft.com/en-us/azure/devops/learn/what-is-infrastructure-as-code

DAFTAR PUSTAKA

Atlassian. (t.thn.). *The Agile Coach*. Diambil kembali dari atlassian.com: https://www.atlassian.com/agile

Aws Amazon. (t.thn.). *What is DevOps?* Diambil kembali dari aws.amazon.com: https://aws.amazon.com/devops/what-is-devops/

Guckenheimer, S. (2017, April 4). *What is Infrastructure as Code?* Diambil kembali dari docs.microsoft.com: https://docs.microsoft.com/en-us/azure/devops/learn/what-is-infrastructure-as-code

Kanbanize. (t.thn.). *What is Lean Management? Definition & Benefits*. Diambil kembali dari kanbanize.com: https://kanbanize.com/lean-management/what-is-lean-management

Scott Chacon, B. S. (2014). *Pro Git Second Edition.* Apress.

Steven, J. (2018, Maret 19). *What's the difference between agile, CI/CD, and DevOps?* Diambil kembali dari Synopsys: https://www.synopsys.com/blogs/software-security/agile-cicd-devops-difference/