BAB II
LANDASAN TEORI

Pada bab ini, peneliti akan menguraikan teori-teori yang mendukung dalam implementasi *Automation Deployment*. Berikut akan dijelaskan teori-teori tersebut dengan lebih lengkap.

2.1. DevOps (*Development and Operations*)

2.1.1 Pengertian DevOps

DevOps merupakan pendekatan kolaboratif antara bagian pengembangan aplikasi (*dev*) dan bagian operasi aplikasi (*ops*). DevOps juga merupakan sebuah *culture* yang ditujukan untuk pembangunan perusahaan yang berkelanjutan [2]. DevOps adalah serangkaian praktik yang mengautomasikan proses antara pengembangan aplikasi dan tim pengembang agar mereka dapat melakukan proses *build*, *test* dan *release* perangkat lunak lebih cepat dan lebih handal.

2.1.2 Nilai inti dari DevOps

Nilai inti DevOps biasanya dijelaskan dengan singkatan CAMS. Merupakan singkatan dari *Culture*, *Automation*, *Measurement*, dan *Sharing* [3].

DevOps memecah hambatan antara tim, menempatkan fokus pada orang dan interaksi mereka di atas proses dan alat. Kejujuran, keterbukaan, dan ketulusan adalah bagian paling berharga dari tim DevOps. Disarankan untuk mengajukan pertanyaan dan berbagi pengetahuan, pelajaran, dan penemuan. Perilaku yang membantu mengoptimalkan proses dan memahami mengapa proses yang ada gagal, didukung, dan dihargai. Mempromosikan lingkungan yang aman untuk inovasi dan produktivitas adalah tantangan utama bagi manajemen perusahaan [4].

Automasi membawa sistem dalam urutan. Ini membantu untuk secara cerdas menggambarkan perubahan yang dibuat terhadap lingkungan. Untuk memahami apa yang harus diubah dalam produk, tim menggunakan log, statistik, dan *feedback* yang dikumpulkan secara konstan. Analisis proses kerja terus dilakukan, setiap perubahan ditinjau untuk memahami apakah aplikasi menjadi lebih baik dan apakah arah yang dipilih sudah benar [4].

2.1.3 Manfaat DevOps

Perusahaan yang mengadopsi praktik DevOps mendapatkan banyak keuntungan teknis dan organisasi. Beberapa diantaranya adalah sebagai berikut:

- **Meningkatkan stabilitas dan kualitas**. Ketika semua anggota tim memiliki lingkungan pengembangan yang sama, integrasi pengujian berkelanjutan dapat dipercepat. Pengujian berkelanjutan ini memungkinkan siklus rilis lebih cepat dan lebih sering dan untuk masalah dan kegagalan yang lebih mudah diidentifikasi [5].

- **Meningkatkan efektivitas organisasi**. Lebih banyak waktu dihabiskan untuk memingkatkan nilai dan kualitas produk [6].

- **Meningkatkan pengalaman pelanggan**. Kemampuan untuk menerima *feedback* terus menerus dan lebih cepat memperkenalkannya ke dalam pengembangan proyek mengarah pada peningkatan pendapatan dan peningkatan kepuasan pelanggan [7].

- **Reaksi cepat terhadap perubahan pasar dan permintaan pelanggan**. Kemampuan untuk mempertahankan tingkat penyebaran yang tinggi ditransformasikan menjadi nilai bisnis dalam dua cara utama: seberapa cepat suatu organisasi dapat berpindah dari ide ke sesuatu yang dapat ditransfer ke pelanggan dan berapa banyak percobaan yang dapat dilakukan organisasi secara bersamaan. Tingkat penyebaran yang tinggi

memungkinkan untuk melakukan percobaan dengan cepat dan terus menerus [8].

- **Automation**. Dengan mengganti prosedur manual dengan automasi membuat manajemen infrastruktur lebih efisien. Hal ini meningkatkan tingkat penggunaan dan kemudahan anda mengelola semua sumber daya, baik *on premises*, pada *cloud*, atau di lingkungan *hybrid*. Respon cepat terhadap perubahan kebutuhan pada bisnis sangat penting. Automasi memungkinkan anda meningkatkan atau menurunkan respon terhadap permintaan. Automasi memastikan konfigurasi konsisten di seluruh jaringan anda, sehingga setiap pengembang memiliki lingkungan yang sama dan pengembang baru dapat dengan mudah menggunakan lingkungan kerja dan mulai bekerja. Konsistensi memberi anda lebih banyak kontrol, dan kontrol mengurangi resiko. Manfaat nyatanya adalah anda memiliki proses standar untuk menyediakan *server* [9].

- **Tingkat komunikasi yang tinggi antara departemen dan anggota tim** [10].

- **Mengurangi risiko perubahan**. DevOps mengurangi risiko perubahan dengan membuat banyak perubahan kecil dan *incremental* alih-alih lebih sedikit. Perubahan kecil lebih mudah untuk ditinjau dan diuji. Karena cakupan setiap perubahan kecil dan umumnya terisolasi, maka jauh lebih mudah untuk memperbaiki kesalahan yang mungkin terjadi [11].

2.1.4 Praktek pada DevOps

- *Continuous Integration* (CI)

  CI adalah praktik pengembangan yang mengharuskan pengembang untuk mengintegrasikan kode ke dalam *mainline* sesering mungkin, dan setiap *check-in* kemudian diverifikasi oleh *automated build* yang mengkompilasi kode dan menjalankan rangkaian uji otomatis terhadapnya, memungkinkan tim mendeteksi masalah sejak dini.

- *Continuous Delivery* (CD)

  CD adalah praktek *software development* dimana para pengembang yang melakukan perubahan pada code, sudah melakukan *build* & *test* yang dijalankan otomatis oleh CI dan siap untuk *deploy* ke *environment production*.

- *Continuous Deployment*

  Continuous Deployment merupakan salah satu rangkaian setelah CI dan CD selesai dijalankan. Umumnya organisasi/ perusahaan memiliki e*nvironment test/ development*, dan disinilah fungsi utama *Continuous Deployment*, yaitu ketika hasil dari *Continuous Integration* sudah dinyatakan baik, tim pengembang dapat segera melihat perubahan pada *environment test/ development/ production*.

- *Configuration Management*

  Configuration Management adalah praktek dalam proses *System Engineering* yang memiliki tujuan untuk me-*maintain* konfigurasi sebuah produk, dan memastikan konsitensinya dalam seluruh *environment*. Dengan menggunakan *Configuration Management*, proses konfigurasi produk dapat diotomatisasi, distandardisasi dan mengurangi proses konfigurasi yang manual. Tahap selanjutnya, *Configuration Management* akan mempermudah dalam konfigurasi banyak *server* dan dapat meminimalisir kesalahan, karena konfigurasi ditulis dalam *code*, tidak lagi menjalankan perintah manual.

- *Infrastructure as a Code* (IaaC)

  Infrastructure as a Code adalah sebuah praktek dalam *System Architecture* yang mana infrastruktur sebuah produk didefinisikan dalam *code* yang dapat diprogram, distandardiasikan dan mudah untuk diduplikasi. Produk skala menengah, mungkin membutuhkan lebih dari satu mesin. Dengan IaaC, tim pengembang dapat dengan mudah menambah mesin melalui satu baris kode.

- *Monitoring*

  Sebuah produk haruslah di-*monitoring* untuk mengetahui bagaimana produk digunakan oleh pengguna. Dalam praktek DevOps, *monitoring* merupakan hal yang sangat penting. Tim pengembang harus mengetahui bagaimana perubahan kodenya berdampak pada produk juga penggunanya melalui *monitoring tools*.

- *Logging*

  Log aplikasi adalah salah satu cara untuk mengetahui apakah produk kita berjalan dengan baik atau tidak. Namun seiring dengan tingkat kompleksitas sebuah produk, ada banyak *log* komponen yang harus diterima dan dianalisis. Dan *log* tersebut haruslah terpusat, tidak terpisah-pisah.

- *Communication* & *Collaboration*

  Salah satu aspek utama dalam praktek DevOps yaitu meningkatnya komunikasi dan kolaborasi dalam sebuah organisasi/ perusahaan, baik dalam bentuk fisik maupun non fisik. Praktek DevOps yang berjalan dengan baik, akan meningkatkan aspek komunikasi dan kolaborasi tidak hanya pada tim pengembang, namun juga tim *marketing*, *sales*, *operations*, dan tim lain yang ada didalam organisasi/ perusahaan

Masalah utama yang perlu diselesaikan DevOps adalah lambatnya proses *delivery*. tujuan utama DevOps adalah mengurangi biaya proses pengembangan dengan mengautomasi dan mengintegrasikan seluruh sistem yang memungkinkan *developer* lebih fokus ke proses *development* [12]

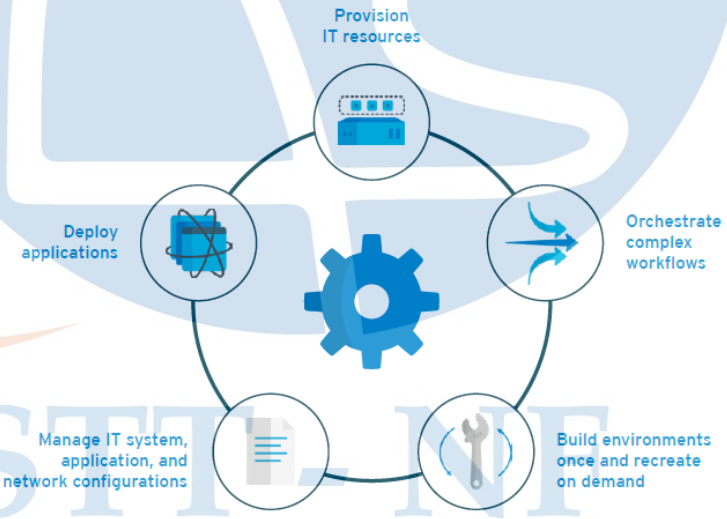
2.2. Pengertian *Automation Deployment*

Tahap Penyebaran (*Deployment*) adalah tahap dimana sistem dibuat tersedia bagi komunitas pengguna. Proses penyebaran harus direncanakan dengan baik sehingga meminimalkan *downtime* dan dampak untuk mengakhiri produktivitas

pengguna. Hal ini tidak hanya mencakup perangkat keras dan perangkat lunak tetapi pengguna akhir.

Automation Deployment memungkinkan aplikasi untuk digunakan di berbagai lingkungan yang digunakan dalam proses pengembangan, serta lingkungan produksi. Teknik ini menghasilkan penerapan yang lebih efisien, andal, dan dapat diprediksi. Proses *Automation Deployment* meningkatkan produktifitas tim Dev dan Ops dan memungkinkan mereka untuk mendeploy lebih cepat dan membangun *software* yang lebih baik untuk *end-user* [13]. *Tools* yang dapat digunakan untuk menerapkan *Automation* salah satunya adalah Ansible.

Automasi adalah satu-satunya cara untuk dapat berhasilnya mengelola lingkungan IT dalam jangka panjang. Menciptakan sebuah pendekatan *enterprise-wide* memungkinkan kita mengautoamsi tidak hanya proses IT, tetapi juga seluruh teknologi dan organisasi, pengembangan proses DevOps dapat dilihat pada gambar 2.1 di bawah ini.



Gambar 2.1: *Automate IT Process*

2.3. Ansible

Ansible adalah mesin autoamsi *open source* yang mengautomasi penyediaan perangkat lunak, Manajemen Konfigurasi, dan pemasangan aplikasi [1]. Ansible disertakan sebagai sebuah *provisioning tool* yang dikembangkan oleh RedHat.