**CI/CD PIPELINE**

Continuous Integration (CI) adalah sebuah praktik pengembangan perangkat lunak yang melibatkan integrasi reguler dan otomatisasi proses build dan test setiap kali ada perubahan kode dalam repositori bersama. Praktik ini bertujuan untuk memastikan bahwa perubahan-perubahan tersebut terintegrasi dengan baik dan tidak menyebabkan masalah dalam pengembangan perangkat lunak.

Secara khusus, CI melibatkan langkah-langkah berikut:

1. Integrasi Kode: Setiap kali seorang pengembang melakukan perubahan pada kode mereka, perubahan tersebut diintegrasikan dengan kode yang ada di repositori bersama. Ini memastikan bahwa semua perubahan dikombinasikan dengan baik dan tidak ada konflik yang terjadi.
2. Build Otomatis: Setelah perubahan kode diintegrasikan, proses build otomatis dijalankan. Proses build ini menghasilkan hasil yang dapat dieksekusi, seperti file biner atau paket perangkat lunak yang siap untuk diimplementasikan.
3. Testing Otomatis: Setelah proses build selesai, langkah selanjutnya adalah menjalankan serangkaian tes otomatis untuk memverifikasi bahwa perubahan-perubahan tersebut tidak mempengaruhi fungsionalitas yang ada. Ini mencakup pengujian unit, pengujian integrasi, dan pengujian fungsional lainnya.

Praktik Continuous Integration berfokus pada kecepatan dan otomatisasi. Dengan mengintegrasikan dan membangun kode secara teratur, tim pengembang dapat mendeteksi masalah lebih cepat dan mengatasi konflik integrasi segera. Hal ini memungkinkan perubahan-perubahan kecil dapat segera diverifikasi dan diuji, sehingga mengurangi risiko kesalahan dan mempercepat waktu respons dalam pengembangan perangkat lunak. Terbaik dicapai melalui integrasi dengan kontrol versi

Continuous Delivery (CD) adalah sebuah praktik pengembangan perangkat lunak yang melibatkan kemampuan untuk melakukan rilis perangkat lunak ke produksi secara fleksibel dan kapan saja. Dalam Continuous Delivery, perangkat lunak yang telah melalui proses CI dapat disiapkan dan dideploy ke sistem staging dengan otomatisasi yang kuat.

Proses Continuous Delivery melibatkan langkah-langkah berikut:

1. Integrasi dan Testing: Setelah perubahan kode diintegrasikan dan dibangun dengan sukses, perangkat lunak tersebut menjalani serangkaian pengujian otomatis untuk memastikan bahwa fungsionalitas yang ada tidak terpengaruh oleh perubahan tersebut. Tes meliputi pengujian unit, pengujian integrasi, pengujian fungsional, dan pengujian lain yang diperlukan.
2. Deployment ke Staging: Setelah perangkat lunak lulus semua tes, ia dapat dideploy ke sistem staging. Sistem staging adalah sebuah lingkungan yang menyerupai produksi, tetapi tidak digunakan oleh pengguna akhir. Tujuan dari staging adalah untuk menguji perangkat lunak dalam kondisi yang mendekati produksi sebelum dilakukan rilis ke lingkungan yang sesungguhnya.
3. Verifikasi di Staging: Setelah perangkat lunak dideploy ke sistem staging, pengujian tambahan dapat dilakukan untuk memverifikasi kinerja, keandalan, dan interaksi dengan komponen lain dalam lingkungan produksi yang telah disimulasikan. Jika semua pengujian berjalan dengan sukses, perangkat lunak dianggap siap untuk dideploy ke produksi.

Continuous Deployment (CD) adalah praktik yang lebih maju dari Continuous Delivery. Dalam Continuous Deployment, perangkat lunak yang telah melewati semua tahap pengujian di staging secara otomatis dideploy ke lingkungan produksi secara langsung dan tanpa intervensi manusia. Proses ini melibatkan otomatisasi yang kuat untuk menjaga kualitas dan keandalan perangkat lunak saat melakukan rilis langsung ke pengguna akhir.

Continuous Deployment memungkinkan organisasi untuk merespons cepat terhadap perubahan pasar, mempercepat waktu rilis, dan meningkatkan umpan balik dari pengguna. Namun, perlu diperhatikan bahwa Continuous Deployment memerlukan tingkat kepercayaan yang tinggi pada otomatisasi dan pengujian yang kuat untuk memastikan bahwa perubahan yang diimplementasikan tidak menyebabkan masalah atau kerugian bagi pengguna atau organisasi itu sendiri.

Apa yang membuat CD Bagus ?

● Desain dengan mempertimbangkan sistem ○ Mencakup sebanyak mungkin bagian penerapan, seperti: Aplikasi, Infrastruktur, Konfigurasi, Data

● Pipa ○ Terus tingkatkan kepercayaan diri saat Anda bergerak menuju produksi

● Versi Unik Secara Global ○ Ketahui status sistem kapan saja ○ Mampu menunjukkan perbedaan antara keadaan saat ini dan masa depan

Apa yang membuat CI Bagus

● Tahapan Terpisah - Setiap langkah dalam CI harus melakukan satu tugas terfokus.

● Dapat diulang - Otomatis dengan cara yang dapat diulang secara konsisten - Tooling juga dapat digunakan untuk developer lokal

● Gagal Cepat - Gagal pada tanda pertama masalah

Proses CI/CD (Continuous Integration/Continuous Deployment) umumnya melibatkan empat tahap utama: commit, build, test, dan deploy. Berikut adalah penjelasan singkat tentang setiap tahap tersebut:

1. Commit:
   * Tahap ini dimulai ketika seorang pengembang melakukan perubahan pada kode sumber dan melakukan commit ke repositori bersama.
   * Setiap commit mewakili satu perubahan terhadap kode sumber yang akan diintegrasikan ke dalam alur kerja CI/CD.
2. Build:
   * Setelah commit dilakukan, tahap build dimulai.
   * Pada tahap ini, sistem akan mengambil kode sumber dari repositori dan membangun atau mengkompilasi aplikasi atau perangkat lunak yang sedang dikembangkan.
   * Build dapat melibatkan penggabungan (merge) perubahan kode dengan kode yang ada, memeriksa ketergantungan eksternal, dan mempersiapkan artefak yang akan digunakan selanjutnya.
3. Test:
   * Tahap ini melibatkan menjalankan serangkaian pengujian pada aplikasi atau perangkat lunak yang telah dibangun.
   * Pengujian dapat mencakup tes unit, tes integrasi, tes fungsional, tes performa, atau tes keamanan, tergantung pada kebutuhan dan kompleksitas proyek.
   * Tujuan dari tahap ini adalah untuk memverifikasi bahwa perubahan kode bekerja dengan baik dan tidak menyebabkan kerusakan pada aplikasi atau sistem yang ada.
4. Deploy:
   * Setelah perubahan kode telah melewati tahap pengujian, tahap terakhir adalah deploy atau penerapan perubahan ke lingkungan produksi atau staging.
   * Pada tahap ini, perubahan kode secara otomatis diterapkan ke server atau infrastruktur yang relevan.
   * Proses deploy dapat melibatkan konfigurasi sistem, pemindahan file ke server, dan aktivasi perubahan kode agar dapat diakses oleh pengguna.

Penerapan Continuous Integration (CI) dalam pengembangan perangkat lunak membawa beberapa manfaat, antara lain:

1. Deteksi Bug Awal: CI melibatkan menjalankan tes otomatis dan melakukan analisis kode pada setiap komit kode. Ini membantu dalam mendeteksi bug dan masalah pada tahap awal pengembangan, memungkinkan pengembang untuk segera mengatasinya. Dengan menangkap bug lebih awal, biaya dan upaya untuk memperbaikinya berkurang secara signifikan.
2. Mengurangi Jumlah Bug: Dengan CI, pengembang diharuskan untuk sering mengintegrasikan perubahan mereka dan memastikan bahwa basis kode tetap dalam keadaan berfungsi. Praktik ini membantu dalam mengidentifikasi masalah dan konflik integrasi sejak dini, menghasilkan lebih sedikit bug dan konflik pada produk akhir. Putaran umpan balik berkelanjutan yang disediakan oleh CI membantu menjaga kualitas kode.
3. Proses Pengembangan yang Lebih Transparan: CI mendorong transparansi dalam proses pengembangan. Pengembang dapat melihat hasil pengujian otomatis dan analisis kode, termasuk cakupan pengujian dan metrik kualitas kode. Transparansi ini mendorong akuntabilitas dan kolaborasi dalam tim pengembangan, karena setiap masalah atau kesalahan dapat dengan mudah diidentifikasi dan diselesaikan.
4. Alur Kerja Pengembangan yang Efisien: CI mengotomatiskan berbagai tugas seperti pembuatan, pengujian, dan penerapan. Otomatisasi ini menghemat waktu dan tenaga bagi pengembang, memungkinkan mereka untuk lebih fokus pada pengkodean dan pengiriman fitur. Dengan mengotomatiskan tugas berulang, alur kerja pengembangan menjadi lebih efisien dan efisien.

Manfaat penerapan CD

1. Reduce The Risiko, Painles
2. Deploymen
3. Mengurangi Biaya
4. Otomatis Dan Proses Transparan
5. Melepaskan Lebih Banyak Sering

CI/CD Tools

1. Jenkins

Jenkins adalah alat otomasi sumber terbuka yang ditulis dalam Java dengan plugin

dibangun untuk integrasi berkelanjutan.

1. Bamboo

Atlassian Bamboo adalah server build integrasi berkelanjutan yang melakukan:

pembuatan, pengujian, dan rilis otomatis di satu tempat

1. Circle CI

Circle CI is a flexible CI tool that runs in any environment like cross-platform mobile

app, Python API server or Docker cluster. This tool reduces bugs and improves the

quality of the application

1. Azure devops

Azure Devops adalah CI/CD tool yang dapat diandalkan untuk perencanaan,

menguji, menyebarkan atau mendistribusikan. Alat ini dibuat oleh Microsoft

1. CI CD
2. Travis CI

Travis CI adalah CI pertama sebagai alat Layanan. Ini memperkenalkan pendekatan baru untuk membangun kode di awan. Alat CI ini memungkinkan pengguna untuk mendaftar, menautkan repositori mereka, membangun, sebagai serta menguji aplikasi mereka

1. Github action

GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform

that allows you to automate your build, test, and deployment pipeline

1. AWS Codebuild

AWS Codebuild adalah alat CI/CD yang memungkinkan pengembang membangun dan menguji kode pada skala berkelanjutan

WORKFLOW : Alur kerja adalah proses otomatis yang dapat dikonfigurasi yang akan menjalankan satu atau lebih pekerjaan. Alur kerja ditentukan oleh file YAML yang didaftarkan ke repositori Anda dan akan berjalan saat dipicu oleh peristiwa di repositori Anda, atau bisa juga dipicu secara manual, atau pada jadwal yang ditentukan

Acara : Peristiwa adalah aktivitas spesifik dalam repositori yang memicu alur kerja berjalan. Misalnya, aktivitas dapat berasal dari GitHub saat seseorang membuat a tarik permintaan, membuka masalah, atau mendorong komit ke repositori

Runner adalah server yang menjalankan alur kerja Anda saat dipicu. Setiap pelari dapat menjalankan satu pekerjaan pada satu waktu. GitHub menyediakan Ubuntu Linux, Microsoft Windows, dan pelari macOS untuk menjalankan alur kerja Anda

Pekerjaan : Pekerjaan adalah serangkaian langkah dalam alur kerja yang dijalankan pada pelari yang sama. Setiap langkah adalah baik skrip shell yang akan dijalankan, atau tindakan yang akan dijalankan. Langkah-langkahnya adalah dijalankan secara berurutan dan saling bergantung satu sama lain. Karena setiap langkah dijalankan pada pelari yang sama, Anda dapat berbagi data dari satu langkah ke langkah lainnya

Actions : An action is a custom application for the GitHub Actions platform that performs a complex but frequently repeated task. Use an action to help reduce the amount of repetitive code that you write in your workflow files