JL. /pgats

Integração Contínua e Testes Automatizados

Capacitar os alunos a implementar e otimizar práticas de Integração Contínua e Testes Automatizados em desenvolvimento de software, abrangendo desde a introdução até a aplicação de boas práticas, configuração de pipelines, análise de resultados e distribuição.

Prof. Samuel Lucas

Ementa e Horários

Integração Contínua e Testes Automatizados

01 Introdução a Integração Contínua
02 Ferramentas de Integração Contínua
03 Configuração de Pipelines de CI
04 Testes Automatizados na Integração Contínua
05 Boas Práticas e Padrões na Integração Contínua
06 Instalando e configurando uma ferramenta para pipeline de Integração Contínua
07 Jobs e Gatilhos de execução da pipeline
08 Adicionando testes automatizados para execução na pipeline
09 Gerando relatórios de execução dos testes
10 Distribuindo os relatórios de execução dos testes
11 Análise de resultados de execução
12 Configurações especiais da ferramenta
13 Plugins
14 Boas práticas no uso da ferramenta

Datas:
23 e 24 de Maio
06 e 07 de Junho

Horários:
Sextas
19h00 - 20h30; 20h45 - 22h00

Sábados
09h00 - 10h30; 10h45 - 12h00;
13h30 - 15h00; 15h15 - 16h30;
16h45 - 18h00

Para você, o que é Integração Contínua?

(abra o microfone ou mande no chat)

Introdução a Integração Contínua

Uma analogia

Introdução a Integração Contínua

Motivação

Introdução a Integração Contínua

1. Desenvolvimento paralelo (branches)
2. Conflitos de código e esforço de mesclagem
3. Silos de conhecimento

Conceito

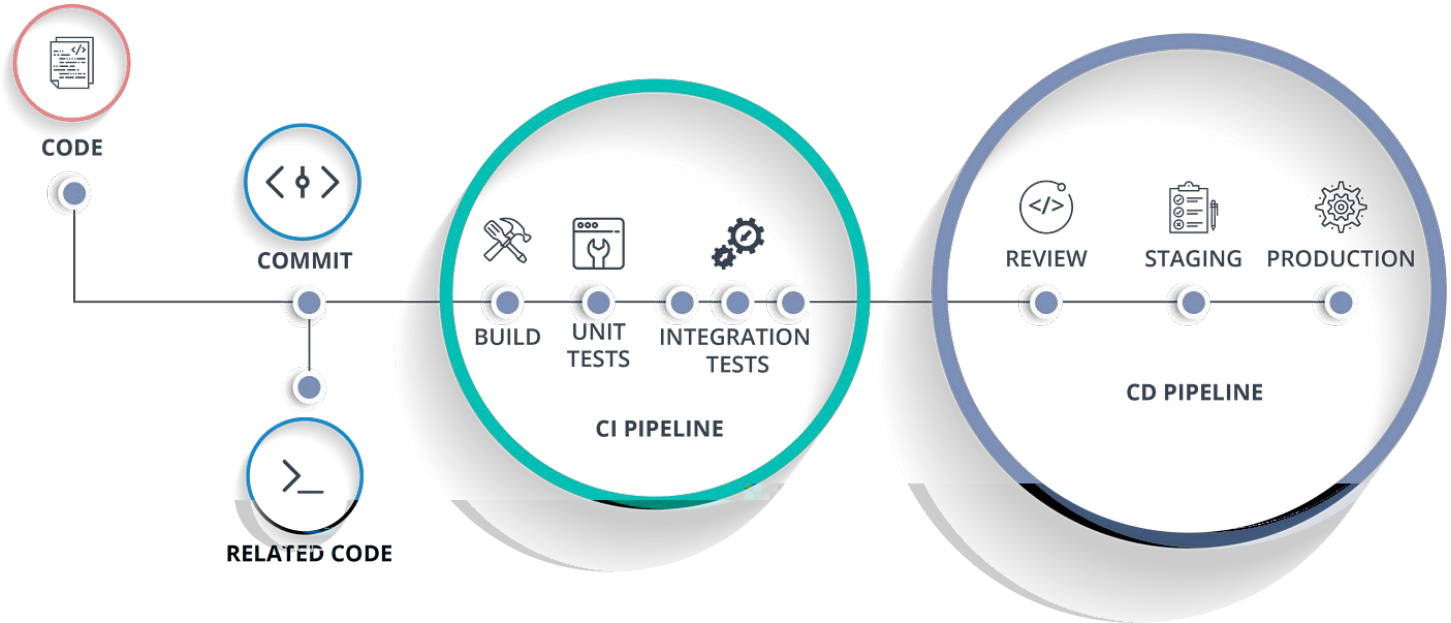
Introdução a Integração Contínua

Integração Contínua é uma prática de desenvolvimento de software em que os desenvolvedores, com frequência, juntam suas alterações de código em um repositório central. Depois disso, **inspeções** e **testes** são executados.

O principal objetivo da Integração Contínua é encontrar problemas mais rápido e reduzir o tempo necessário para identificá-los e corrigi-los.

Conceito

Introdução a Integração Contínua



Fundamentos - Etapas macro do Processo de CI

Introdução a Integração Contínua

1. Compilação
2. Testes
3. Inspeção
4. Implantação

Fundamentos - Etapas macro do Processo de CI

Introdução a Integração Contínua

Compilação é o primeiro passo do processo de build, onde o código-fonte escrito em uma linguagem de programação de alto nível é convertido em um formato executável, como código de máquina ou bytecode. Este processo é crucial, pois permite que o código seja interpretado e executado pelo sistema operacional ou pela máquina virtual, garantindo que não haja erros de sintaxe e que todas as dependências sejam resolvidas.

Fundamentos - Etapas macro do Processo de CI

Introdução a Integração Contínua

Após a compilação, o próximo passo é a **execução de testes**, que validam a funcionalidade do software e garantem que ele atende aos requisitos especificados. Isso inclui testes em diferentes camadas, como unidade, integração e aceitação, permitindo a detecção precoce de falhas e garantindo que as novas alterações no código não quebrem funcionalidades existentes.

Fundamentos - Etapas macro do Processo de CI

Introdução a Integração Contínua

A **inspeção** envolve a análise do código e dos resultados dos testes para identificar problemas de qualidade como vulnerabilidades de segurança ou violações de padrões de codificação. Isso pode incluir a realização de revisões de código, a análise estática do código-fonte e o uso de ferramentas de linting. O objetivo da inspeção é garantir que o código não apenas funcione corretamente, mas também mantenha um padrão elevado de legibilidade e manutenibilidade.

Fundamentos - Etapas macro do Processo de CI

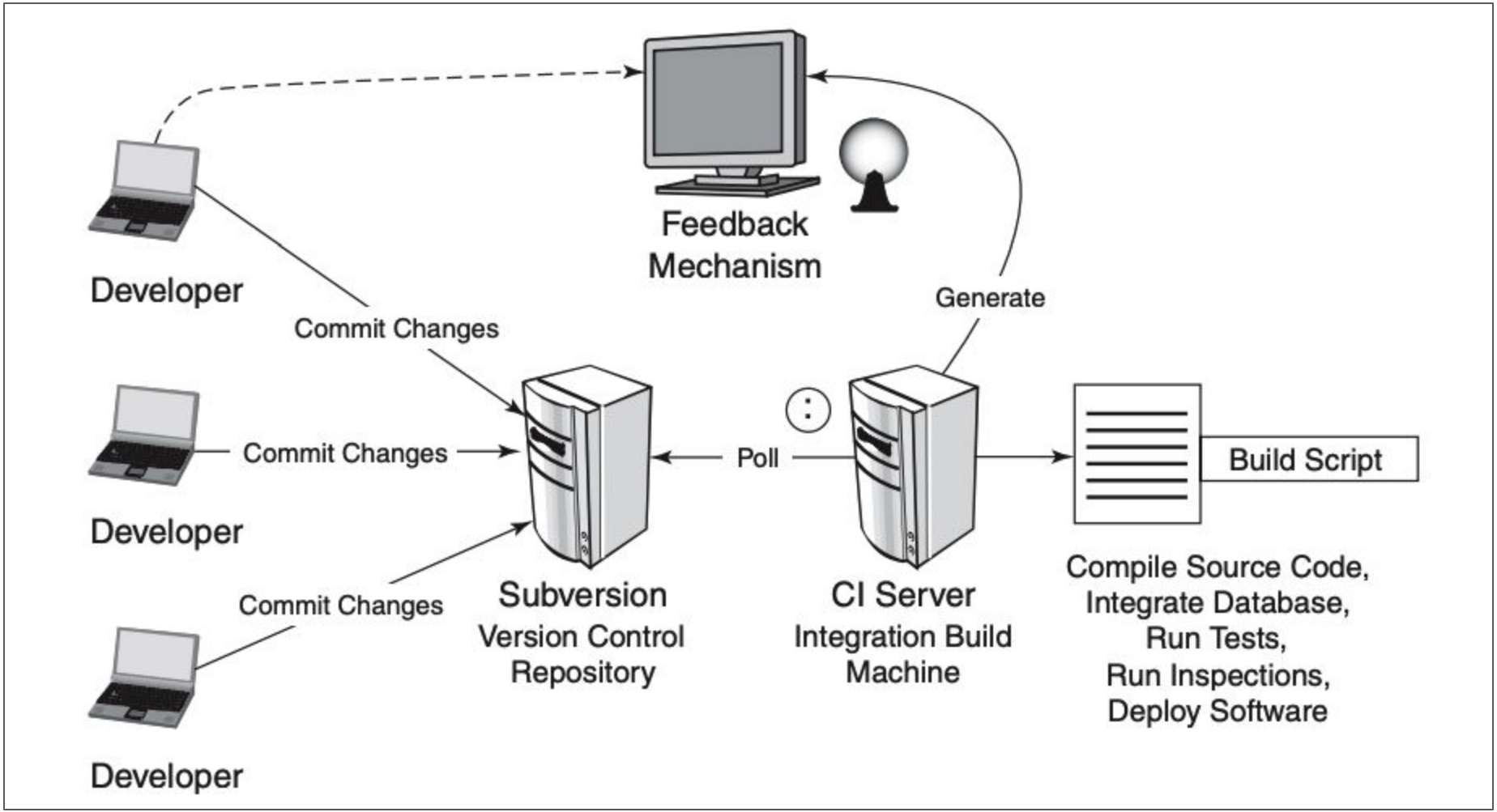
Introdução a Integração Contínua

A **implantação** é a etapa final do processo de build, onde a versão compilada e testada do software é disponibilizada para uso em um ambiente de produção. Isso pode envolver a transferência de arquivos para servidores, a configuração de ambientes de execução e a realização de testes finais para garantir que tudo funcione conforme esperado.

Fundamentos - Componentes do Processo de CI

Introdução a Integração Contínua

- Desenvolvedor
- Controle de Versão
- Servidor de Automação
- Script de Build
- Mecanismo de Feedback
- Máquina para Execução



Máquinas de Build do SoundCloud (2015)

Fundamentos - Vocabulário

Introdução a Integração Contínua

1. Ramo (branch)
2. Tronco (trunk ou branch principal)
3. Pull Request (pr)
4. Construção (build)
5. Artefato
6. Implantação (deploy)
7. Reverter Implantação (rollback)
8. Lançamento Candidato (release candidate)
9. Lançamento (release)
10. Pacote (package)
11. Rótulo (tag)
12. Pipeline

Ferramentas de Integração Contínua

Ferramentas de Integração Contínua

Introdução a Integração Contínua

- *Controle de Versão*
  - *Git*
  - *SVN*
  - *TFVC*

Ferramentas de Integração Contínua

Introdução a Integração Contínua

- *Build*
  - *Node - npm, Yarn, pnpm*
  - *Java - Ant, Maven, Gradle*
  - *Swift - XCode*
  - *React Native - react-native build command line*
  - *Flutter - flutter build command line*
  - *Go - go build command line*
  - *Outras - Make, Buck2, Bazel*

Ferramentas de Integração Contínua

Introdução a Integração Contínua

- *Servidores de Automação*
  - *On-premise ou Cloud*
  - *Cloud-hosted agents vs Self-hosted agents*
  - *Declarative, Scripted, Visual*

Ferramentas de Integração Contínua

Introdução a Integração Contínua

- *Servidores de Automação*
  - *Jenkins*
  - *Github Actions*
  - *Azure DevOps Pipelines*
  - *Bitbucket Pipelines*
  - *Gitlab CI*
  - *Circle CI*

Configuração de Pipelines de CI

The Continuous Delivery Maturity Model

	Base	Beginner	Intermediate	Advanced	Expert
Culture & Organization	• Prioritized work • Defined and documented process • Frequent commits	• One backlog per team • Share the pain • Stable teams • Adopt basic Agile methods • Remove boundary dev & test	• Extended team collaboration • Component ownership • Act on metrics • Remove boundary dev & ops • Common process for all changes • Decentralize decissions	• Dedicated tools team • Team responsible all the way to prod • Deploy disconnected from Release • Continuous improvement (Kaizen)	• Cross functional teams • No rollbacks (always roll forward)
Design & Architecture	• Consolidated platform & technology	• Organize system into modules • API management • Library management • Version control DB changes	• No (or minimal) branching • Branch by abstraction • Configuration as code • Feature hiding • Making components out of modules	• Full component based architecture • Push business metrics	• Infrastructure as code
Build & Deploy	• Versioned code base • Scripted builds • Basic scheduled builds (CI) • Dedicated build server • Documented manual deploy • Some deployment scripts exsists	• Polling builds • Builds are stored • Manual tag & versioning • First step towards standardized deploys	• Auto triggered build (commit hooks) • Automated tag & versioning • Build once deploy anywhere • Automated bulk of DB changes • Basic pipeline with deploy to prod • Scripted config changes (e.g. app server) • Standard process for all environments	• Zero downtime deploys • Multiple build machines • Full automatic DB deploys	• Build bakery • Zero touch continuous deployments
Test & Verification	• Automatic unit tests • Separate test environment	• Automatic integration tests	• Automatic component tests (isolated) • Some automatic acceptance tests	• Full automatic acceptance tests • Automatic performance tests • Automatic security tests • Risk based manual testing	• Verify expected business value
Information & Reporting	• Baseline process metrics • Manual reporting	• Measure the process • Static code analysis • Scheduled quality reports	• Common information model • Traceability built into pipeline • Report history is available	• Graphing as a service • Dynamic test coverage analysis • Report trend analysis	• Dynamic graphing and dashboards • Cross silo analysis

Maturidade do Processo de Testes no CI

Introdução a Integração Contínua

1. Disparar o Pipeline de testes de forma manual a partir de 1 clique
2. Disparar o Pipeline de testes de forma automática em períodos definidos
3. Disparar o Pipeline de testes sempre que for concluído Deploy no ambiente
4. Disparar o Pipeline de testes como uma fase dentro do Pipeline de desenvolvimento, sendo um *Quality Gate* para publicar no ambiente

Desafios:

- ❏ Quem analisa os resultados?
- ❏ Se houver falhas, quem reporta?
- ❏ Se os testes estiverem desatualizados, quem atualiza?
- ❏ Se houver "alarmes falsos" no *pipeline*, o que fazer?
- ❏ Se os testes estiverem atrasando o *pipeline*, o que fazer?

Testes Automatizados no CI

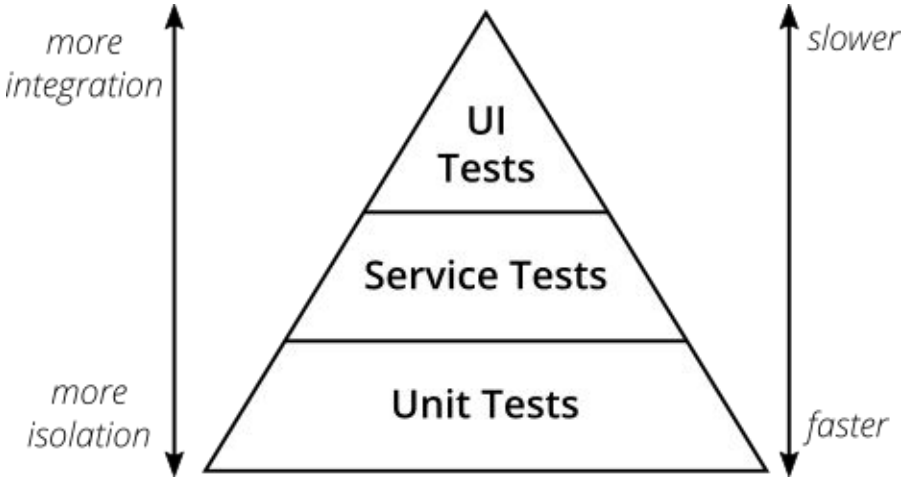
Testes Automatizados no CI

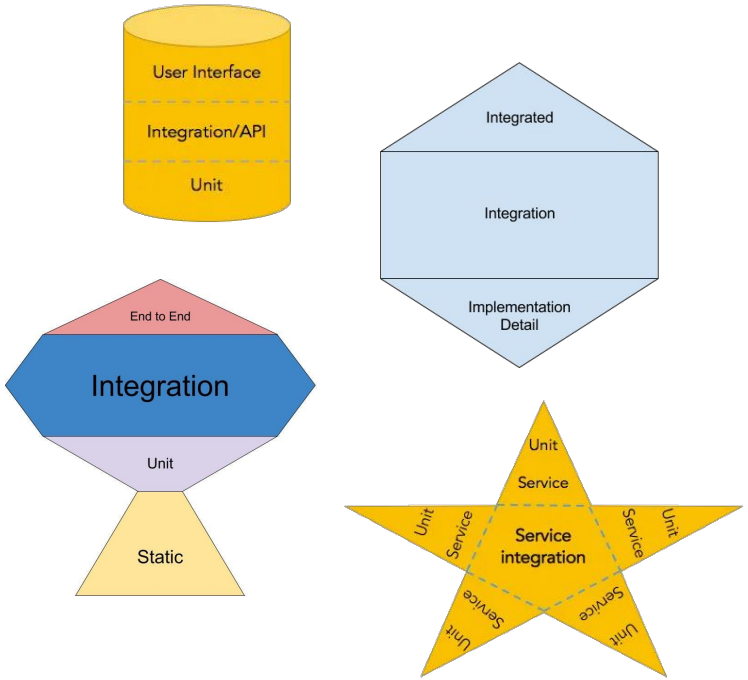
Introdução a Integração Contínua

- *Build fast, fail fast*
- *Feedback rápido e contínuo*
- *Padronização e Assertividade*

Testes Automatizados no CI

Introdução a Integração Contínua





Testes Automatizados no CI

Introdução a Integração Contínua

Há diferentes convenções sobre quanto de testes e quais camadas devem ser automatizadas, assim como quando devem ser executados no CI.

Testes no CI são como testes fora do CI - contextuais.

Pipelines na Prática

Exercício Individual

Introdução a Aplicações Web

Hands-on

Use os conceitos e exemplos praticados em aula e aplique em outra ferramenta de Integração Contínua. Sugestões:

- Azure DevOps
- CircleCI
- Gitlab CI
- Jenkins

Exercício Individual

Introdução a Aplicações Web

Hands-on

Explore os *plugins* disponíveis no Marketplace e escolha um que pode agregar ao fluxo de trabalho como: relatórios, notificações, IA, etc.

https://github.com/marketplace?type=actions

Exercício Individual

Introdução a Aplicações Web

Hands-on

Leia sobre self-hosted runners/agents. Tente executar os pipelines criados usando um.

Avalie: Quando faz sentido usar esse recurso? Outras plataformas oferecem recursos similares?

Revisão

Perguntas e Respostas

Até a próxima aula