

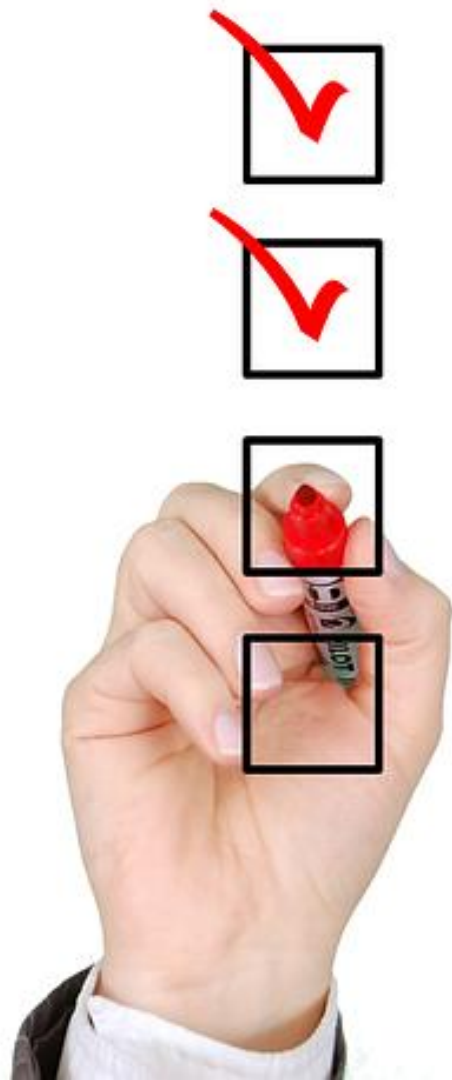
INTEGRAÇÃO E ENTREGA CONTÍNUA

Introdução à Disciplina e Conceitos Fundamentais de CI/CD


Professora:

Lucineide Pimenta

Tópicos da aula



- ❑ Controle de versão com Git e GitHub.
- ❑ Integração Contínua e automação de testes.
- ❑ Entrega Contínua e automação de deploys.
- ❑ Monitoramento e logging em sistemas.
- ❑ Segurança e boas práticas em pipelines de CI/CD.

- ❑  **Referência:** Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley.

Objetivo da Aula

1. Apresentar a disciplina e como ela se relaciona com o projeto ABP.
2. Explicar os conceitos fundamentais de Integração Contínua (CI) e Entrega Contínua (CD).
3. Demonstrar a importância da automação no ciclo de desenvolvimento de software.
4. Apresentar ferramentas essenciais utilizadas no mercado para CI/CD.



O Que é Integração Contínua (CI)?

❑ **Definição:**

- É um processo de automação para integração de código de múltiplos desenvolvedores em um repositório compartilhado.
- Permite a detecção precoce de erros através de execução automática de testes.

❑ **Exemplo real:** Imagine uma equipe de desenvolvedores criando um aplicativo bancário.

Cada um escreve uma parte do código e ao integrar tudo no repositório, um sistema automatizado verifica se nenhuma alteração quebrou a aplicação.

Caso um erro seja detectado, o desenvolvedor é notificado imediatamente para correção.

 **Referência:** Fowler, M. (2006). *Continuous Integration*. Disponível em:
<https://martinfowler.com/articles/continuousIntegration.html>

O Que é Entrega Contínua (CD)?

❑ **Definição:**

- É a prática onde o software está sempre pronto para ser implantado em produção.
- Usa pipelines automatizados para build, testes e implantação.

- ❑ **Exemplo real:** Uma loja virtual frequentemente recebe novas funcionalidades. Com a Entrega Contínua, assim que uma nova função é aprovada e testada, ela é automaticamente enviada para os servidores da loja, ficando disponível para os clientes sem necessidade de interrupção.

 **Referência:** Kim, G. (2016). *The DevOps Handbook*. IT Revolution.

O que é CI/CD e por que é importante?

- ❑ Imagine que você está construindo o aplicativo do INPE e a cada nova funcionalidade precisa garantir que o app não quebre.
- ❑ O **CI/CD** é como um robô que testa, constrói e, se tudo estiver certo, entrega o aplicativo para os usuários - tudo isso automaticamente.

O que é CI/CD e por que é importante?

- ❑ **CI – Integração Contínua (Continuous Integration):**
 - ❑ É o processo de unir o trabalho de todos os desenvolvedores várias vezes ao dia em um repositório compartilhado.
 - ❑ Cada integração dispara um conjunto de verificações automáticas (testes, análise de qualidade, build).

O que é CI/CD e por que é importante?

- ❑ **CD – Entrega Contínua (Continuous Delivery):**
 - ❑ É levar o software sempre pronto para implantação.
 - ❑ Isso significa que a qualquer momento é possível colocar a nova versão em produção com segurança.

O Que é Entrega Contínua (CD)?

❑ **Benefícios:**

- Redução do tempo entre desenvolvimento e produção.
- Implementação de novas funcionalidades de forma mais segura.
- Padronização do processo de deploy.

❑  **Referência:** Kim, G. (2016). *The DevOps Handbook*. IT Revolution.

Benefícios para o projeto do INPE

- ❑ Menos erros chegando aos usuários.
- ❑ Entrega mais rápida de novas funcionalidades (alertas de queimadas, inundações etc.).
- ❑ Equipe trabalhando de forma colaborativa, sem “travar” o código do colega.
- ❑ Histórico de mudanças e possibilidade de voltar atrás se algo der errado.

O Que são Pipelines, Build e Testes Automatizados?

1. Pipeline:

- Um conjunto de processos automatizados que movem o código de uma etapa para outra (exemplo: testes, build, deploy).
- **Exemplo:** No desenvolvimento de um aplicativo, o código passa por uma pipeline que executa testes, verifica a qualidade e depois o implanta em produção.

2. Build:

- Processo de transformar o código-fonte em um programa executável.
- **Exemplo:** Um desenvolvedor escreve código em JavaScript, e o build gera os arquivos necessários para rodar a aplicação no navegador.

3. Testes Automatizados:

- Conjunto de verificações automáticas para garantir que o código funcione como esperado.
- **Exemplo:** Testes unitários verificam se funções individuais retornam os valores esperados.

O Que é Deploy e Como Ele Funciona?

❑ **Definição:**

- O processo de colocar uma aplicação em funcionamento em um ambiente de produção.
- Pode ser manual ou automatizado (Continuous Deployment).

❑ **Exemplo real:** O Instagram implementa melhorias constantes sem que os usuários percebam.

Isso é possível porque os deploys são feitos de maneira automatizada, sem afetar a disponibilidade da plataforma.

 **Referência:** AWS Docs. *Blue/Green Deployments on AWS*. Disponível em:
<https://docs.aws.amazon.com/>

O Que é um Repositório Compartilhado?

- ❑ **Definição:**

Local centralizado onde os desenvolvedores armazenam e gerenciam código. GitHub, GitLab e Bitbucket são exemplos de plataformas populares.

- ❑ **Exemplo real:** Uma equipe de desenvolvimento de um aplicativo de mensagens armazena todo o código no GitHub.

Quando um membro faz uma alteração, o restante da equipe tem acesso instantâneo, garantindo sincronização entre todos.

Como Ocorre a Redução de Erros na Integração?

❑ **Processo:**

1. O código de cada desenvolvedor é testado automaticamente ao ser enviado.
2. Se houver erros, a equipe é notificada para correção antes da implementação final.
3. Testes automatizados garantem que novos recursos não afetem funcionalidades existentes.

- ❑ **Exemplo real:** Um aplicativo de e-commerce detecta que uma nova funcionalidade de pagamento está causando falha no carrinho de compras.
O erro é identificado rapidamente e corrigido antes de afetar os usuários.

Benefícios da Automação no Desenvolvimento de Software

- ❑ **Redução de Erros:** Testes automatizados evitam problemas em produção.
- ❑ **Maior Velocidade:** Menos tempo gasto com tarefas repetitivas.
- ❑ **Consistência:** Builds e deploys são feitos de forma padronizada.
- ❑ **Colaboração Aprimorada:** Todos os desenvolvedores trabalham na mesma base de código, evitando conflitos.

 **Referência:** Bass, L. (2015). *DevOps: A Software Architect's Perspective*. Addison-Wesley.

O Que é um Commit e Como Ele Funciona?

❑ **Definição:**

- Um commit é um ponto de controle no histórico do código-fonte, representando uma versão do projeto.
- Cada commit contém uma mensagem descrevendo as alterações feitas.

Exemplo real:

- Um desenvolvedor altera o design de um site e faz um commit com a mensagem "Melhorias na interface da página inicial".
- Esse commit fica registrado no histórico do repositório e pode ser recuperado caso necessário.



Referência: Chacon, S., & Straub, B. (2014). *Pro Git*. Apress.

Benefícios da Integração e Entrega Contínua (CI/CD)

1. Redução de Erros:

- Erros de integração são detectados rapidamente, evitando problemas em produção.
- Testes automatizados garantem que cada alteração no código não afete funcionalidades existentes.

2. Feedback Rápido para Desenvolvedores:

- Cada commit aciona testes automáticos, fornecendo feedback imediato sobre problemas.
- Equipes conseguem corrigir falhas antes que elas afetem usuários finais.

Benefícios da Integração e Entrega Contínua (CI/CD)

3. Agilidade no Desenvolvimento:

- Implementações frequentes de pequenas mudanças tornam o desenvolvimento mais ágil.
- Diminui o tempo entre escrita de código e disponibilização em produção.

4. Maior Confiança nas Entregas:

- Processos automatizados garantem que apenas código validado chegue à produção.
- Menos necessidade de intervenção manual, reduzindo erros humanos.

Benefícios da Integração e Entrega Contínua (CI/CD)

5. Consistência no Processo de Deploy:

- Cada nova versão do software é implantada da mesma forma, garantindo estabilidade.
- Implementações podem ser desfeitas rapidamente em caso de falha.

 **Referência:** Bass, L. (2015). *DevOps: A Software Architect's Perspective*. Addison-Wesley.

Ferramentas Essenciais para CI/CD

- ❑ **Git & GitHub:** Controle de versão e armazenamento de repositórios.
- ❑ **GitHub Actions:** Automatiza processos de CI/CD.
- ❑ **Jenkins:** Plataforma open-source para automação.
- ❑ **Docker:** Containerização para consistência entre ambientes.
- ❑ **Kubernetes:** Orquestração de containers.
- ❑ **Firebase Hosting:** Hospedagem de aplicações web e mobile.

 **Referência:** OpenAI. *Ferramentas para CI/CD*. Disponível em:
<https://openai.com/tools-ci-cd>

Casos Reais de Uso

- ❑ **Google:** CI/CD para atualizações constantes no Android e Chrome.
- ❑ **Netflix:** Pipelines automatizados para implantação segura de novos recursos.
- ❑ **Facebook:** Entrega contínua de código sem interromper serviços.

 **Referência:** Forsgren, N. (2018). *Accelerate: The Science of Lean Software and DevOps*. IT Revolution.

Como funciona na prática (pipeline básico)

- ❑ O desenvolvedor cria/edita uma funcionalidade e envia para o repositório.
- ❑ O GitHub Actions detecta a mudança e:
 - ❑ Executa testes automatizados.
 - ❑ Faz análise de qualidade do código.
 - ❑ Compila o aplicativo.
 - ❑ Se aprovado, envia para o ambiente de homologação ou produção.

Links de Referência

-  O que é CI/CD – Atlassian
-  Guia Rápido de CI/CD – Red Hat



Integração e Entrega Contínua (IEC)

Exercícios Práticos 01

Bibliografia Básica

- ❑ HUMBLE J; PRIKLANDNICKI R. **Entrega Contínua**: Como Entregar Software de Forma Rápida e Confiável. São Paulo: Bookman, 2013.
- ❑ MUNIZ, A.; et al. **Jornada DevOps**: Unindo Cultura Ágil, Lean e Tecnologia Para Entrega de Software Com Qualidade. São Paulo: Brasport, 2019.
- ❑ SATO D. **DevOps na prática**: entrega de software confiável e automatizada. São Paulo: Casa do Código, 2014.
- ❑ SILVA, R. **Entrega contínua em Android**: Como automatizar a distribuição de apps. São Paulo: Casa do Código, 2016.

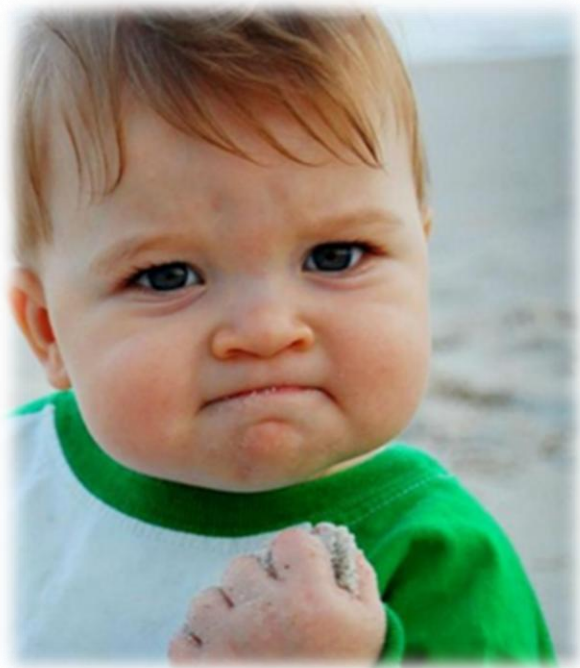
Bibliografia Complementar

- ❑ ARUNDEL, J. DOMINGUS, J. **DevOps nativo de nuvem com Kubernetes**. São Paulo: Novatec, 2019.
- ❑ MORAES, G. **Caixa de Ferramentas DevOps**: Um guia para construção, administração e arquitetura de sistemas modernos. São Paulo: Casa do Código, 2015. PIRES, A.; MILITÃO, J. **Integração Contínua com Jenkins**. São Paulo: Casa do Código, 2019.
- ❑ VITALINO, J. F. N.; CASTRO, M. A. N. **Descomplicando o Docker**. 2 ed. São Paulo: Brasport, 2018.
- ❑ SILVERMAN, R. E. **Git**: guia prático. São Paulo: Novatec, 2019.

Dúvidas?



Considerações Finais



**Professor(a):
Lucineide Pimenta**

Bom semestre à todos!

