

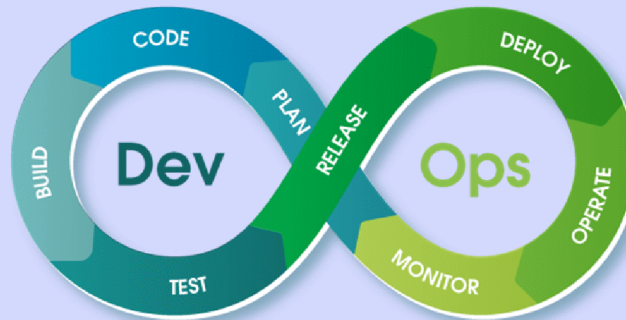
DevOps Professional

► Preparatório para o exame oficial do EXIN

ESTE É UM CURSO OFICIAL

Curso online completo para compreender os conceitos, princípios e práticas do DevOps

Curso atualizado para o exame atual



Todos os direitos de cópia reservados. Não é permitida a distribuição física ou eletrônica deste material sem a permissão expressa do autor.

Versão: 1.1 Liberação: 23/07/18

Módulo 1: Introdução ao DevOps

Aviso de marcas registradas e direitos autorais

- Todos os direitos reservados. Nenhuma parte deste material poderá ser reproduzida ou transmitida em qualquer ou por qualquer meio sem a permissão escrita da TIEXAMES Consultoria e Treinamento Ltda.
- Algumas marcas registradas podem aparecer no decorrer deste curso. O uso delas, bem como de logotipos é apenas para fins editoriais, em benefício exclusivo do dono da marca registrada, sem intenção de infringir as regras de sua utilização.



Tl.exames

Módulo 1



Introdução ao DevOps

Parte 2

Peso no exame: 2,5%

Este módulo 1 cobre:

- Infraestrutura ágil (agile infrastructure).
- Entrega contínua (continuous delivery).
- Débito técnico (technical debt).

TI.exames

© Todos os direitos reservados. Proibida a redistribuição deste material.

Slide 3

Módulo 1: Introdução ao DevOps

Infraestrutura Ágil (Agile Infrastructure)

- Infraestrutura ágil é basicamente aplicar princípios ágeis à infraestrutura.
- Um dos pontos centrais deste movimento é tratar sua infraestrutura com código (infrastructure as code).
- É possível gerenciar e provisionar ambientes automaticamente por meio de código, ao invés de usar um processo manual.

```
- hosts: server
  sudo: yes
  sudo_user: root

  tasks:

    - name: Install mysql-server
      apt: name=mysql-server state=present update_cache=yes

    - name: Install ansible dependencies
      apt: name=python-mysqldb state=present

    - name: Ensure mysql is running
      service: name=mysql state=started

    - name: Create user with the password and all privileges
      mysql_user: login=user-root login_password={{ name }} password={{
        mysql_password }} priv=*.*ALL host=* state=present

    - name: Delete test database
      mysql_db: name=test state=absent

    - name: Create ansible_example database
      mysql_db: name=ansible_example state=present

    - name: Copy mysql back up dump to the remote user
      copy: src=dump.sql.bz2 dest=/tmp

    - name: Restore the dump into ansible_example database
      mysql_db: name=ansible_example state=import target=/tmp/dump.sql.bz2
```

Por exemplo, considere o script Ansible ao lado:

- 1 Instala o mysql-server na máquina remota (servidor).
- 2 Assegura que o mysql está rodando.
- 3 Cria um usuário com a senha.
- 4 Exclui o banco de dados de teste.
- 5 Cria o banco de dados ansible_example.
- 6 Copia um dump sql para a máquina e o restaura em ansible_example base de dados.

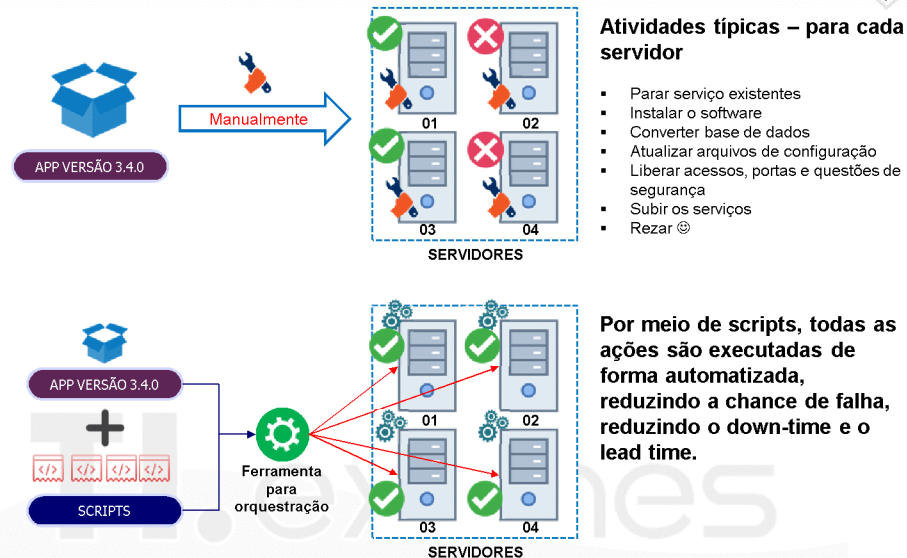
- Todas essas ações são executadas automaticamente.
- Se a tarefa necessária já tiver sido executada na máquina, a tarefa não será executada novamente.
- Isso apenas garante que a máquina esteja no estado requerido.

TI.exames

© Todos os direitos reservados. Proibida a redistribuição deste material.

Slide 4

Infraestrutura Ágil (Agile Infrastructure)



Infraestrutura Ágil (Agile Infrastructure)

- Algumas ferramentas consolidadas que permitem tratar a infraestrutura como código:



<https://www.docker.com>



<https://www.vagrantup.com>



<https://www.chef.io>



<https://www.ansible.com>



<https://puppet.com>



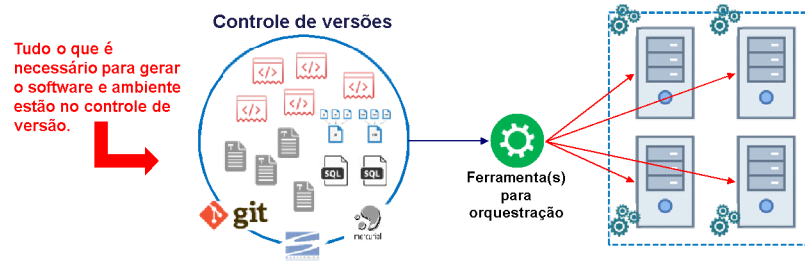
<https://saltstack.com>

- Puppet, Chef, Ansible e SaltStack são ferramentas disponíveis para orquestração de nuvem.
- Orquestração de nuvem** é a automação de ponta a ponta da implementação de serviços em um ambiente de nuvem.

<https://www.intigua.com/blog/puppet-vs.-chef-vs.-ansible-vs.-saltstack>

Infraestrutura como código

- Tratar a infraestrutura como código permite que ambientes sejam criados a partir de um repositório de código-fonte, um backup de dados e, claro, um servidor físico disponível ("bare metal").



- Para isso:
 - Trabalhe com infraestrutura como código.
 - Use técnicas de virtualização (criação virtual de hardware, sistemas operacionais, espaço de armazenamento ou recursos de rede).
 - Use a nuvem (pública ou privada).

Módulo 1



Introdução ao DevOps

Parte 2

Peso no exame: 2,5%

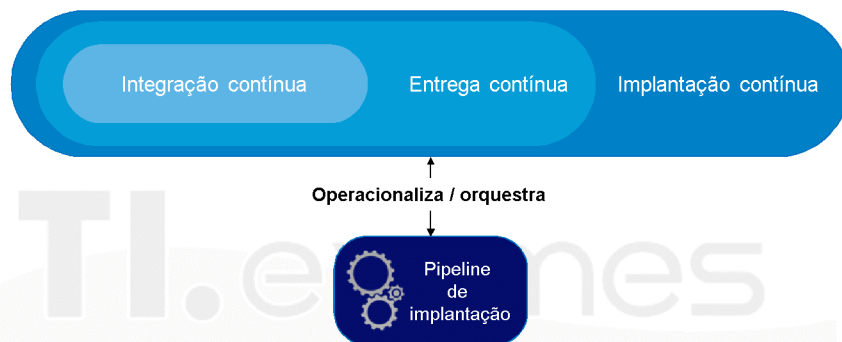
Este módulo 1 cobre:

- Infraestrutura ágil (agile infrastructure).
- Entrega contínua (continuous delivery).
- Débito técnico (technical debt).

Entrega contínua (continuous delivery)

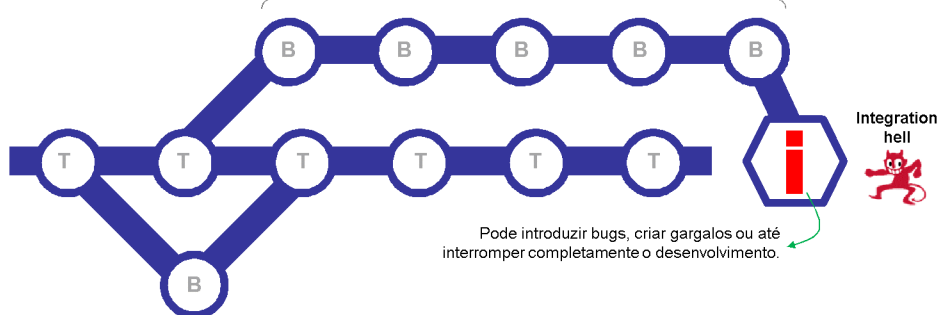
- A entrega contínua existe para que as funcionalidades sejam liberadas continuamente e de forma segura para o cliente.
- A entrega contínua é uma disciplina de desenvolvimento de software na qual você cria software de maneira que ele possa ser liberado para produção a qualquer momento.

**Entrega contínua estende o conceito de integração contínua.
Implantação contínua estende o conceito de entrega contínua.**



O problema da integração e a integração contínua

Quanto maior o tempo sem integrar código, mais problemático se torna.



- A integração contínua, ou CI, é uma estratégia de fluxo de trabalho que ajuda a garantir que as alterações de todos se integrem à versão atual do projeto.
- Isso permite **identificar erros, reduzir conflitos de merge** e ter certeza de que seu software está funcionando.

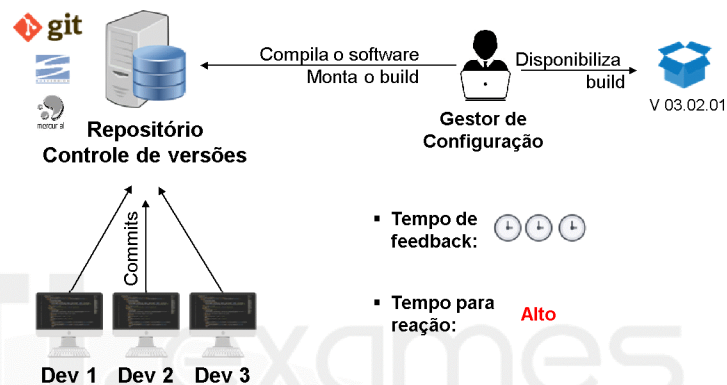
Na maioria dos cenários, uma equipe praticará o CI em conjunto com testes automatizados



Integração e teste de código

- A integração de código produzido por desenvolvedores sempre foi uma parte importante do processo de desenvolvimento. Sem integração contínua, tudo é feito manualmente.

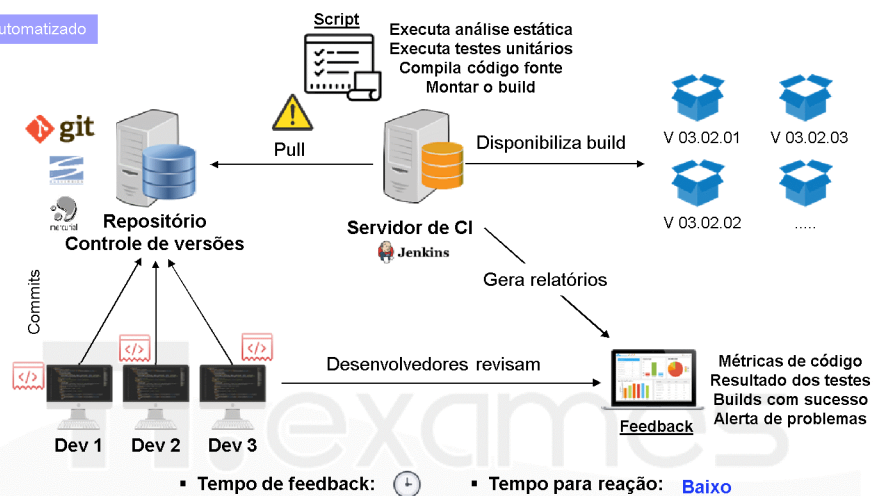
Manual



Integração contínua (Continuous integration)

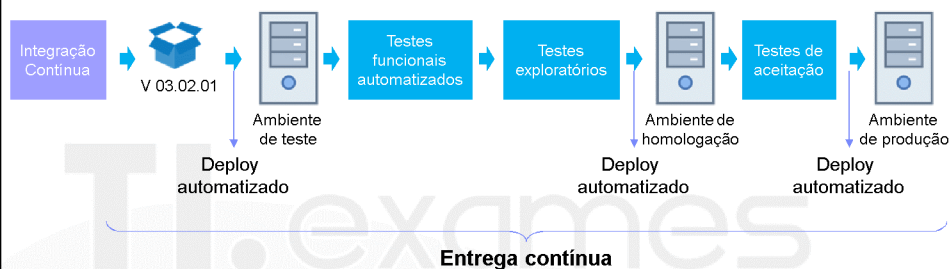
- A integração contínua se refere à integração, construção e teste de código dentro do ambiente de desenvolvimento, permitindo então a realização de testes sistêmicos.

Automatizado



Entrega Contínua (Continuous Delivery)

- A integração contínua concentra-se principalmente nos times desenvolvedores.
- A saída do sistema de CI normalmente forma a entrada para o processo de teste manual e daí para o restante do processo de liberação.
- Este processo manual resulta em um software que não pode ser implantado rapidamente
- A solução é adotar uma abordagem mais holística, de ponta a ponta, para entregar software, ou seja, adotar a **entrega contínua (continuous delivery)**



Entrega Contínua (Continuous Delivery)

- A entrega contínua estende a integração contínua.
- Além de automatizar seus testes, **também é necessário automatizar o processo de implantação**, possibilitando a implantação com um simples clique em um botão.
→ **Infraestrutura como código é portanto um pré-requisito para a entrega contínua.**
- A entrega contínua garante que tanto o **código (software)** como a **infraestrutura ("as code")** estejam em um estado implantável a qualquer tempo no processo de desenvolvimento e implantação.

Você está fazendo entrega contínua quando:

- Seu software é implantável em todo o seu ciclo de vida.
- Sua equipe prioriza manter o software em estado "implantável" em vez de trabalhar em novos recursos.
- Qualquer pessoa pode obter feedback rápido e automatizado sobre a prontidão de produção de seus sistemas sempre que alguém fizer uma alteração.
- Você pode realizar implantações de qualquer versão do software **em qualquer ambiente**, sob demanda, com o simples apertar de um botão.

Entrega Contínua (Continuous Delivery)

- Para conseguir uma entrega contínua, você precisa de:
 - Um relacionamento próximo e colaborativo entre todos os envolvidos na entrega (**desenvolver uma cultura DevOps**).
 - **Automação** extensiva de todas as partes possíveis do processo de entrega, geralmente usando um pipeline de implantação.
- Os principais benefícios da entrega contínua são:

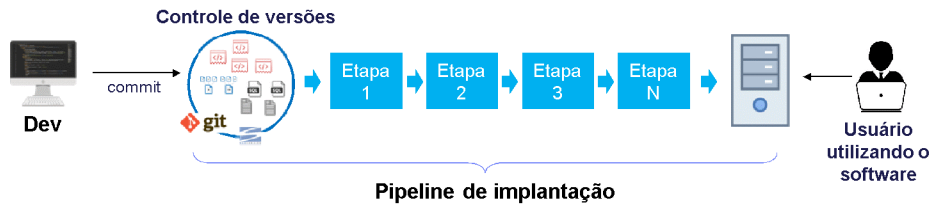


Implantação contínua

- A Entrega Contínua às vezes é confundida com a Implantação Contínua.
- Implantação contínua significa que todas as alterações passam pelo pipeline de implantação (deployment pipeline) e são **automaticamente** colocadas em produção, resultando em muitas implantações de produção todos os dias.
- Entrega Contínua significa apenas que você é capaz de realizar implantações frequentes, mas pode optar por não fazê-lo, geralmente devido a empresas que preferem uma taxa mais lenta de implantação.
- Para fazer a Implantação Contínua, você deve estar fazendo a Entrega Contínua.

Pipeline de Implantação (deployment pipeline)

- Em um nível abstrato, um pipeline de implantação é uma manifestação **automatizada** de seu processo para levar o software do controle de versão para as mãos de seus usuários.



- Normalmente, o primeiro estágio de um pipeline de implantação fará qualquer compilação e fornecerá binários para as etapas posteriores.
- As etapas posteriores podem incluir verificações manuais, como quaisquer testes que não possam ser automatizados.
- Etapas podem ser automáticas, ou exigirem autorização humana para prosseguir, elas podem ser paralelizadas em muitas máquinas para acelerar a execução.
- A implantação na produção é geralmente o estágio final de um pipeline.

Integração, Entrega e Implantação contínua

Integração Contínua (Continuous Integration)



Entrega Contínua (Continuous Delivery)



Implantação Contínua (Continuous Deployment)



Módulo 1



Introdução ao DevOps

Parte 2

Peso no exame: 2,5%

Este módulo 1 cobre:

- Infraestrutura ágil (agile infrastructure).
- Entrega contínua (continuous delivery).
- Débito técnico (technical debt).

Módulo 1: Introdução ao DevOps

Débito técnico (technical debt)

O custo implícito de retrabalho adicional causado pela escolha de uma solução fácil agora, em vez de usar uma abordagem melhor que levaria mais tempo.

O débito técnico descreve como as decisões que tomamos levam a problemas que se tornam cada vez mais difíceis de consertar com o tempo, reduzindo continuamente nossas opções disponíveis no futuro.

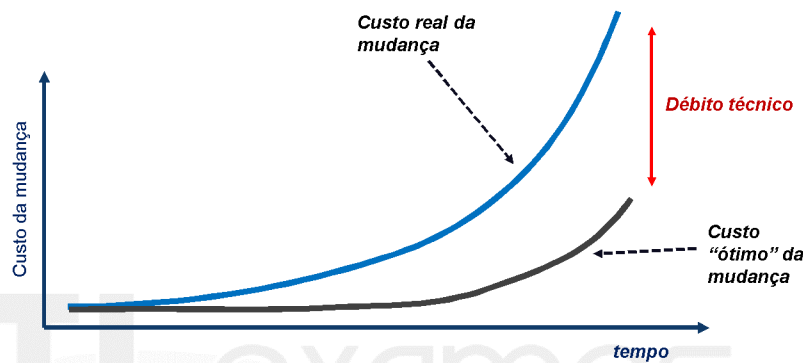


Débito técnico é tudo aquilo que reduz a velocidade do time e que aumenta o TCO.

- Erros no sistema, ausência de testes automatizados, alta complexidade do código, ausência de testes unitários, arquitetura ruim, ausência de padrões de design e código, entre outros.

Débito técnico (technical debt)

- É necessário saber que débito técnico é algo inevitável. Sempre vai existir, mas deve ser em baixo nível. E, se não for pago, o débito tende a aumentar com o tempo.



Débito técnico é causado por:

Definição inicial insuficiente

- Requisitos ainda estão sendo definidos durante o desenvolvimento.
- O desenvolvimento é iniciado antes de qualquer design.
- Isso é feito para economizar tempo, mas muitas vezes tem que ser retrabalhado mais tarde.

Pressões do negócio

- A empresa considera a obtenção de algo liberado antes que todas as mudanças necessárias sejam concluídas.
- Acumula dívida técnica que inclui essas alterações não concluídas.

Ausência de processo ou compreensão

- Empresas são cegas para o conceito de débito (dívida) técnico e tomam decisões sem considerar as implicações.

Componentes fortemente acoplados

- As funções não são modulares, o software não é flexível o suficiente para se adaptar às mudanças nas necessidades de negócios.

Ausência de testes

- A falta de um conjunto de testes, que incentiva testes superficiais e arriscados para corrigir bugs.
- Famoso "dá uma testada e libera".

Ausência de documentação

- O código é criado sem a documentação de suporte necessária.

Ausência de colaboração

- O conhecimento não é compartilhado em toda a organização e a eficiência dos negócios é prejudicada ou os desenvolvedores juniores não são devidamente orientados.

Desenvolvimento paralelo

- O desenvolvimento paralelo em duas ou mais branches acumula dívida técnica devido ao trabalho necessário para mesclar as mudanças em uma única base de origem.

Ausência de refatoração sistemática

- À medida que os requisitos de um projeto evoluem, pode ficar claro que partes do código se tornaram ineficientes ou difíceis de editar e precisam ser refatoradas para suportar requisitos futuros.

Minissimulado final


Recomendamos agora completar o minissimulado deste módulo. Clique no botão abaixo para abrir a página do minissimulado.

CLIQUE PARA REALIZAR O MINISSIMULADO

O botão acima não abriu a página do minissimulado?

Caso o botão acima não abra a página do minissimulado no seu navegador, acesse o link "Realizar quiz", disponível na lista de módulos gravados dentro da página do curso, no ambiente de ensino.



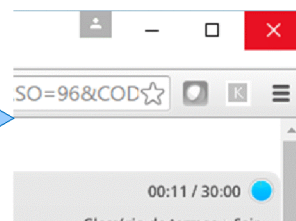
 [Realizar quiz](#)

 [Baixar Pdf](#)

Fim do módulo

Pronto, você finalizou este módulo. Leia as instruções abaixo:

- Recomendamos neste momento fazer uma revisão dos slides para confirmar o entendimento de tudo o que foi apresentado neste módulo.
- Você pode clicar no (X) da janela para fechar este módulo.
- A sua nota obtida no teste do slide anterior será exibida na lista dos módulos. Se isso não ocorrer, é porque você está utilizando um navegador incompatível. Utilize apenas os navegadores Chrome ou Firefox para que o script de captura da nota funcione.



Última nota quiz
60%

 [Ver aula](#)

 [Baixar Pdf](#)