Integração Contínua: pipelines e testes automatizados com GitHub Actions

Recapitulando - Apresentação

0:00 / 3:19

Olá! Boas-vindas à Alura. Meu nome é **Vinicius Dias** e serei seu instrutor neste curso de **Introdução à Integração Contínua com GitHub Actions**.

> ***Audiodescrição:*** *Vinicius se autodeclara como uma pessoa branca. Ele tem cabelo escuro e curto, além de bigode e cavanhaque bem definidos e rentes ao rosto. Veste uma camiseta azul com detalhes em branco. À sua frente, um microfone apoiado num braço articulado. Ao fundo, um ambiente com iluminação roxa e rosa.*

Pré-requisitos

Se você trabalha na área de tecnologia ou tem interesse nela - sendo de qualquer área, como desenvolvimento, SysAdmin, DevOps e até QA - este curso será bastante útil para você!

Neste curso, vamos colocar na prática conceitos de integração contínua utilizando GitHub Actions como ferramenta. Então, para que você tire proveito deste curso, é necessário que você tenha familiaridade com **Git e Docker**, porque vamos utilizar essas ferramentas durante o curso.

Além disso, é necessário que você já conheça os **conceitos de integração contínua**. Vamos recapitular esse assunto de forma bem resumida, mas é necessário que você já saiba do que se trata integração contínua. Inclusive, já existem cursos mais teóricos sobre o assunto na plataforma da Alura para que você possa chegar aqui com tudo pronto para praticar!

O que vamos aprender?

Este curso será bastante **prático**: vamos pegar os conceitos e os aplicar. Primeiro, vamos recapitular os conceitos teoricamente, depois vamos colocar na prática uma espécie de esteira com alguns passos de execução, para podermos implementar a nossa integração contínua de forma segura e confiável.

Definidos os passos da automatização de processos para a integração contínua, vamos para o uso da ferramenta deste curso: o **GitHub Actions**. Vamos explorar essa ferramenta, aprender a configurá-la e integrá-la no nosso processo.

Após configurar a nossa automação com GitHub Actions, vamos um pouco mais além para configurar um **repositório no GitHub** para ter uma integração contínua mais **segura**. Com isso, vamos impedir commits diretos na branch `main`, por exemplo, exigindo *pull request*, e vamos garantir que o *pull request* só possa ser mergeado se as nossas actions estiverem passando.

Ao final, teremos uma conversa - bem resumida e introdutória - sobre **entrega contínua e deploy automatizado**. Mas não vamos nos aprofundar tanto nessa parte, porque, mais

uma vez, o foco deste curso é integração contínua em si.

Dicas de estudo!

Se você ficar com alguma dúvida ao longo das aulas, não hesite em abrir um tópico no **fórum do curso**. Com certeza alguém vai conseguir te ajudar, porque temos uma ótima comunidade de estudantes, pessoas moderadoras e instrutoras.

Convidamos você para, além de responder perguntas nesse mesmo fórum, participando de outras formas além de apenas perguntar, também fazer parte do nosso servidor do **Discord**. Lá a interação é um pouco mais dinâmica e instantânea, e você com certeza pode tirar bastante proveito dela.

Esperamos você no próximo vídeo, para recapitularmos de forma bem resumida o que é integração contínua, antes de começarmos a colocar a mão na massa!

Recapitulando - O que é integração contínua

0:00 / 3:56

Antes de começarmos a aplicar os conceitos de integração contínua, vamos recapitular brevemente o que é a integração contínua em si.

> *Se você acabou de concluir os cursos teóricos e não julgar necessário recapitular, sinta-se à vontade para pular este vídeo, se quiser.*

O que é a integração contínua?

A integração contínua é um conceito que se aplica principalmente em projetos onde trabalhamos em **equipe**, ou seja, na grande maioria dos projetos no mundo profissional. Nesse contexto, você terá um repositório local, por exemplo, utilizando o Git na sua máquina, e de tempos em tempos, fará o ***push*** (envio) dos seus *commits* (confirmações) com as alterações feitas para algum **repositório remoto**.

A integração contínua se refere à frequência com a qual unimos essas modificações feitas com a *branch* principal no nosso repositório. Ou seja, com que **frequência integramos as mudanças com o código existente** em um sistema em que estamos trabalhando.

A integração contínua prega que isso deva ser feito de forma, como o nome diz, **contínua**, de forma muito frequente - pelo menos uma vez por dia, ou várias vezes ao dia, suas alterações devem ser integradas à *branch* principal (`main`) do seu repositório.

Isso é feito para evitar problemas, por exemplo, de uma funcionalidade do projeto estar funcionando perfeitamente na sua máquina local, ou até no servidor num *branch* separado, mas quando se integra com outras mudanças de outras equipes, quebra a aplicação. Portanto, se você integra de forma contínua, detecta esse tipo de problema e consegue agir sobre ele muito mais rápido.

Processos da integração contínua

Para atingir esse objetivo de integrar nossas mudanças com o código existente de forma contínua, constante e frequente, precisamos de alguns processos. Precisamos, por exemplo, de **testes** e **controle de qualidade** do nosso software, e precisamos que tudo isso seja automatizado.

Neste curso, vamos nos aprofundar na parte prática da implementação desse processo, utilizando uma das ferramentas possíveis para automatização de processos visando a integração contínua.

É muito importante entender que alguns conceitos, normalmente, são deturpados conforme o tempo avança. A integração contínua é amplamente utilizada em praticamente todas as empresas; no entanto, muitas pessoas chamam de "integração contínua" apenas a parte de executar os testes, ou apenas uma pequena parte de todo o processo de integração contínua.

Reforçarmos que integração contínua é todo o processo - é a **cultura** de integrar modificações no código ao código já existente de forma frequente. As automatizações, as ferramentas de qualidade e as ferramentas que vamos conhecer neste curso são apenas uma parte da integração contínua. Em outras palavras, são ferramentas que nos possibilitam **atingir** a integração contínua.

Próximos passos

Agora que já recapitulamos a parte teórica, vamos colocar um pequeno projeto no ar. Vamos utilizar o projeto de outro curso da Alura, de uma linguagem de programação que o instrutor não conhece, como exemplo.

Ele fará isso para demostrar que, para configurar uma *pipeline* (que você vai entender o que é em breve) ou automatizar um processo a ponto de atingir a integração contínua, você não precisa ser uma pessoa desenvolvedora ou conhecer aquela linguagem a fundo. Você precisa conhecer os **princípios** necessários para atingir a integração contínua.

No próximo vídeo, vamos colocar um projeto no ar na nossa máquina local e entender o que vamos fazer durante o curso.

Recapitulando - Subindo o projeto

0:00 / 3:48

Conforme mencionado no vídeo anterior, vamos utilizar uma linguagem de programação com a qual o instrutor não tem familiaridade, para que estejamos mais ou menos no

mesmo nível de domínio. Caso você conheça essa linguagem, sem problemas, será ainda melhor!

Subindo um projeto para o servidor

O primeiro passo será **clonar o repositório** disponível para acesso nas atividades do curso, do Projeto Go Alura. Para isso, abrimos o terminal no VS Code e digitamos o comando `git clone` seguido do **endereço** do repositório. Com isso, você terá um projeto para praticar.

Este projeto foi escrito em ***Go*** (Golang). Novamente, não é necessário saber *Go* para utilizá-lo. Este é um projeto de aplicação que **gerencia estudantes**.

Para executar esse projeto, vamos utilizar ***Docker***. Nele, temos um arquivo chamado `docker-compose.yml` com dois serviços: um **banco de dados *Postgres***, já com um volume montado para ser criado na pasta do projeto, e um serviço para a **aplicação** em si, que vai utilizar *Golang* e ser executado via comando. Ou seja, o projeto está pronto para ser executado. O projeto depende do banco de dados *Postgres*, ou seja, ele já se conecta ao banco de dados.

docker-compose.yml

```
services:
  postgres:
    image: "postgres"
    environment:
      - POSTGRES_USER=root
      - POSTGRES_PASSWORD=root
      - POSTGRES_DB=root
    volumes:
      - ./postgres-data:/var/lib/postgresql/data

  app:
  image: golang:1.22
  command:
    - go
    - run
    - main.go
  volumes:
    - ./:/app
```

```
    working_dir: /app
    ports:
      - 8080:8080
    depends_on:
      - postgres
```

Após clonar o projeto, tudo o que você vai precisar fazer é rodar o seguinte comando no terminal:

```
docker compose up
```

Se você der "***Enter***" nesse comando, o seu terminal vai travar. No nosso caso, é isso mesmo que queremos: ficar com o terminal travado para não digitar mais nada nele. Caso você queira utilizar seu terminal para outras coisas, você vai rodar um comando alternativo, com `-d`:

```
docker compose up -d
```

Esse comando vai subir o projeto e não vai travar o terminal - conceitos que você já deve ter aprendido nos cursos de *Docker*.

Subimos o projeto. Caso queiramos verificar o que está acontecendo com a aplicação, rodamos `docker compose` com os comandos `logs` e `-f`, para continuar **seguindo** os logs da nossa aplicação, `app`.

```
docker compose logs -f app
```

No retorno, confirmamos que a aplciação já está ouvindo e servindo o HTTP na porta 8080.

Então, no navegador, vamos acessar o seguinte endereço pela barra de busca:

```
localhost:8080/alunos
```

Isso vai devolver uma **lista de alunos**, que está vazia porque ainda não temos nada no banco de dados.

Se completarmos o endereço com algum nome, como `/vinicius/`, a aplicação vai fazer uma **saudação**:

```
localhost:8080/alunos/vinicius
```

Resposta:

{"API diz":"E aí Vinícius, tudo beleza?"}

Basicamente, essa é a API que vamos utilizar. Não tem muito segredo, não precisamos conhecer o código. Temos um projeto pronto para ser utilizado.

Fazendo o fork do projeto

E como a ferramenta que vamos utilizar tem "*GitHub*" no nome, faz todo sentido utilizar o próprio **GitHub**, certo?

No repositório que você acessou para clonar, clique no botão "***Fork***" no menu de tarefas à direita do nome do projeto. Isso vai criar uma **cópia desse repositório** na sua conta do GitHub. Com isso, você vai ter o seu próprio repositório para começar a trabalhar e fazer todas as alterações que precisamos fazer, assim como as configurações do próprio *GitHub*.

No repositório clonado na sua conta, você vai clicar em "***Code***", copiar o endereço do campo "SSH" (por exemplo, `git@github.com:CViniciusSDias/projeto_go_alura.git`), e fazer um `git clone` no VS Code seguido do endereço que você copiou. Feito isso, rode o comando `docker compose up` para subir para o servidor. Com isso, você vai ter o **projeto associado à sua conta**, pronto para testar ou implementar qualquer processo necessário para atingir a integração contínua.

Próximos passos

Na próxima aula, vamos preparar a aplicação e entender o que vamos automatizar em um processo de integração contínua. Para isso, vamos executar alguns comandos interessantes.

Até lá!

Sobre o curso Integração Contínua: pipelines e testes automatizados com GitHub Actions

O curso Integração Contínua: pipelines e testes automatizados com GitHub Actions possui **95 minutos de vídeos**, em um total de **38 atividades**. Gostou? Conheça nossos outros **cursos de Builds** em **DevOps**, ou leia nossos artigos de DevOps.

Matricule-se e comece a estudar com a gente hoje! Conheça outros tópicos abordados durante o curso:

- Recapitulando
- Preparando o terreno
- GitHub Actions
- Protegendo os segredos
- Geração de artefato

Certificado

Mentorias com especialistas

Comunidade exclusiva

Acesso ao conteúdo das Imersões

App Android e iOS para estudar onde quiser

O mais escolhido

PRO

Acelere o seu aprendizado com a IA da Alura e prepare-se para o mercado internacional.

R$149 /mês
1 ano de Alura

Valor total R$1.788 em até 12x

MATRICULE-SE

Todos os benefícios do PLUS e mais vantagens exclusivas:

Luri , a inteligência artificial da Alura

Alura Língua - Inglês e Espanhol

ULTRA

Transforme a sua jornada com benefícios exclusivos e evolua ainda mais na sua carreira.

R$209 /mês

1 ano de Alura

Valor total R$2.508 em até 12x

MATRICULE-SE

Todos os benefícios do PRO e mais vantagens exclusivas:

Luri , com mensagens ILIMITADAS

Luri Vision , a IA que enxerga suas dúvidas

6 Ebooks da Casa do Código

Conheça os Planos para Empresas

Nossas redes e apps

Institucional

Sobre nós

Carreiras Alura

Para Empresas

Para Sua Escola

A Alura

Como Funciona

Formações

Plataforma

Depoimentos

Cursos de Front-end
HTML, CSS | React | Angular | JavaScript | jQuery

Cursos de Data Science
Ciência de dados | BI | SQL e Banco de Dados | Excel | Machine Learning | NoSQL | Estatística

Cursos de Inteligência Artificial
IA para Programação | IA para Dados

Cursos de DevOps
AWS | Azure | Docker | Segurança | IaC | Linux

Cursos de UX & Design
Usabilidade e UX | Vídeo e Motion | 3D

Cursos de Mobile
Flutter | iOS e Swift | Android, Kotlin | Jogos

Cursos de Inovação & Gestão
Métodos Ágeis | Softskills | Liderança e Gestão | Startups | Vendas

CURSOS UNIVERSITÁRIOS FIAP

Graduação | Pós-graduação | MBA