Centro Paula Souza

Fatec Jacareí
Prof. Francisco de Moura

INTEGRAÇÃO E ENTREGA CONTÍNUA

Automação com GitHub Actions

Professora: Lucineide Pimenta

Tópicos da aula

Fatec
Jacareí
Prof. Francisco de Moura

- Apresentar o conceito de automação no desenvolvimento de software.
- Apresentar o funcionamento do GitHub Actions e sua importância em pipelines de CI/CD.
- Demonstrar como criar workflows automatizados para integração contínua.
- Implementar um pipeline básico usando GitHub Actions na prática.

Introdução à Automação no Desenvolvimento de Software

Fatec Jacareí
Prof. Francisco de Moura

- **Por que automatizar processos de desenvolvimento?**
  - Reduz o tempo gasto com tarefas repetitivas.
  - Minimiza erros humanos.
  - Aumenta a eficiência e confiabilidade do software.
  - Possibilita feedback rápido sobre mudanças no código.
- **Exemplo real:** Imagine um time desenvolvendo um aplicativo de e-commerce. A cada nova funcionalidade, é necessário rodar testes, validar código e fazer deploy. Sem automação, esses processos são manuais e propensos a erros. Com GitHub Actions, cada push pode automaticamente rodar testes e preparar o ambiente de produção.

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

O Que é GitHub Actions?

Fatec
Jacareí
Prof. Francisco de Moura

- **Definição:**
  - GitHub Actions é uma ferramenta de automação nativa do GitHub para CI/CD.
  - Permite a execução de scripts automatizados dentro do próprio repositório GitHub.
  - Baseia-se em workflows definidos por arquivos YAML.
- **Principais Benefícios:**
  - Integração direta com repositórios GitHub.
  - Execução de tarefas automatizadas a cada commit, push ou pull request.
  - Facilidade na criação de fluxos de trabalho personalizados.

🔗 **Referência:** https://docs.github.com/en/actions

Conceito de Workflows no GitHub Actions

Fatec
Jacareí
Prof. Francisco de Moura

- **O que é um workflow?**
  - Um conjunto de ações automatizadas que são executadas em resposta a eventos.
  - Configurado através de arquivos YAML dentro do repositório.
  - Pode ser acionado manualmente ou automaticamente por eventos do GitHub.

Conceito de Workflows no GitHub Actions

Fatec Jacareí
Prof. Francisco de Moura

- **Exemplo:**

```yaml
name: CI Pipeline
on: [push]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout do código
        uses: actions/checkout@v2
      - name: Instalar dependências
        run: npm install
      - name: Rodar testes
        run: npm test
```

🔗 **Referência:** https://docs.github.com/en/actions/learn-github-actions

INTEGRAÇÃO E ENTREGA CONTÍNUA

Configuração de um Workflow Básico

Configuração de um Workflow Básico

Fatec Jacareí
Prof. Francisco de Moura

- **Passo a passo:**
  1. Criar a pasta *.github/workflows* no repositório.
  2. Criar um arquivo YAML (*ci-pipeline.yml*).
  3. Definir eventos que acionam o workflow (push, pull request, etc.).
  4. Criar etapas de execução (checkout, build, testes, deploy).
  5. Testar o workflow e validar o funcionamento.

🔗 **Referência:** https://docs.github.com/en/actions/quickstart

Configuração de um Workflow Básico - Passo a Passo

Fatec Jacareí
Prof. Francisco de Moura

1. **Criar um Repositório no GitHub**
   - Acesse o GitHub e crie um novo repositório.
   - Clone o repositório para sua máquina local.
2. **Criar a Pasta do Workflow**
   - Dentro do repositório, crie a pasta *.github/workflows/*
   - Essa pasta armazenará todos os arquivos de automação do GitHub Actions.
3. **Criar um Arquivo YAML**
   - Dentro da pasta *workflows/*, crie um arquivo YAML, por exemplo: *ci-pipeline.yml*.

Configuração de um Workflow Básico - Passo a Passo

Fatec
Jacareí
Prof. Francisco de Moura

4. Definir um Evento Disparador

- No arquivo YAML, defina quais eventos devem acionar o workflow. Exemplo:

```
on:
    push:
        branches:
            - main
```

Configuração de um Workflow Básico - Passo a Passo

Fatec Jacareí
Prof. Francisco de Moura

5. Criar um Job e Definir o Ambiente

- Dentro do YAML, configure um job para rodar a automação:

```
jobs:
    build:
    runs-on: ubuntu-latest
    steps:
```

Configuração de um Workflow Básico - Passo a Passo

Fatec Jacareí
Prof. Francisco de Moura

6. Adicionar Etapas ao Workflow

- As etapas podem incluir checkout do código, instalação de dependências e execução de testes:

```
steps:
    - name: Checkout do código
    uses: actions/checkout@v2
    - name: Instalar dependências
    run: npm install
    - name: Rodar testes
    run: npm test
```

Configuração de um Workflow Básico - Passo a Passo

Fatec Jacareí
Prof. Francisco de Moura

7. Salvar e Comitar o Arquivo

- Após finalizar o arquivo, salve e execute os comandos:

```
git add .
git commit -m "Adicionando workflow de CI“
git push origin main
```

Configuração de um Workflow Básico - Passo a Passo

Fatec
Jacareí
Prof. Francisco de Moura

8. Verificar Execução no GitHub

- Acesse o repositório no GitHub.
- Clique na aba "Actions" e veja se o workflow foi executado corretamente.

🔗 **Referência:** https://docs.github.com/en/actions/quickstart

INTEGRAÇÃO E ENTREGA CONTÍNUA

Boas Práticas no Uso de GitHub Actions

Boas Práticas no Uso de GitHub Actions

Fatec Jacareí
Prof. Francisco de Moura

1. Manter workflows simples e modularizados.
2. Usar ações reutilizáveis para evitar redundância.
3. Evitar execuções desnecessárias para reduzir custos.
4. Armazenar segredos de ambiente com GitHub Secrets.
5. Monitorar logs de execução para identificar falhas rapidamente.

🔗 **Referência:** https://github.blog/

Boas Práticas no Uso de GitHub Actions

Fatec Jacareí
Prof. Francisco de Moura

1. Manter Workflows Simples e Modularizados

- **Por que?**
  - Workflows complexos podem ser difíceis de manter e depurar.
  - Modularizar permite reutilização e facilita ajustes.
- **Exemplo real:** Uma equipe de desenvolvimento percebeu que seu workflow continha mais de 300 linhas de código e levava muito tempo para rodar. Ao dividir o workflow em jobs separados para build, testes e deploy, a execução ficou mais rápida e fácil de monitorar.
- **Como aplicar?**
  - Divida o pipeline em **múltiplos jobs**.
  - Use **ações reutilizáveis** para evitar código duplicado.
  - Evite **scripts longos dentro do YAML** e prefira scripts externos.

🔗 **Referência:** https://docs.github.com/en/actions/learn-github-actions/reusing-workflows

Boas Práticas no Uso de GitHub Actions

Fatec Jacareí
Prof. Francisco de Moura

2. Usar Ações Reutilizáveis para Evitar Redundância

- **Por que?**
  - Reduz código repetitivo entre múltiplos repositórios.
  - Facilita atualizações centralizadas.
- **Exemplo real:** Uma empresa de software gerenciava 10 repositórios diferentes, todos com workflows semelhantes. Ao criar uma **ação personalizada** centralizada, eliminaram a duplicação e facilitaram futuras alterações.
- **Como aplicar?**
  - Crie **ações personalizadas** e armazene em repositórios públicos ou privados.
  - Utilize **marketplace de ações** para evitar reinventar soluções.
  - Exemplo de um workflow usando ação reutilizável:

```
uses: my-org/my-custom-action@v1.2
with:
  parameter: value
```

- 🔗 **Referência:** https://docs.github.com/en/actions/creating-actions

Boas Práticas no Uso de GitHub Actions

Fatec Jacareí
Prof. Francisco de Moura

3. Evitar Execuções Desnecessárias para Reduzir Custos

- **Por que?**
  - Cada execução consome recursos da conta GitHub, o que pode gerar custos.
  - Evitar execuções desnecessárias melhora o desempenho da pipeline.
- **Exemplo real:** Uma startup enfrentava altos custos no GitHub Actions porque cada commit acionava testes completos, mesmo para pequenas alterações de documentação. Ao restringir a execução de workflows apenas para alterações no código, reduziram 40% dos custos.
- **Como aplicar?**
  - Defina filtros para rodar workflows apenas quando necessário:

```yaml
on:
  push:
    paths:
      - 'src/**'
      - '!docs/**'
```

  - Utilize **caches** para evitar recompilar dependências já existentes.
  - Execute testes completos **somente na branch principal** e use testes mais leves em branches secundárias.

🔗 **Referência:** https://docs.github.com/en/actions/using-jobs/using-conditions-to-control-job-execution

Boas Práticas no Uso de GitHub Actions

Fatec Jacareí
Prof. Francisco de Moura

4. Armazenar Segredos de Ambiente com GitHub Secrets

- **Por que?**
  - Nunca armazene **senhas, tokens ou chaves de API** diretamente no código.
  - O GitHub Secrets mantém esses dados protegidos e acessíveis apenas no ambiente de execução.
- **Exemplo real:** Uma equipe desenvolvendo um aplicativo usava uma API de pagamentos. Inicialmente, armazenaram a chave da API no código-fonte. Um desenvolvedor acidentalmente compartilhou o repositório e expôs os dados. Após esse erro, passaram a usar **GitHub Secrets**.
- **Como aplicar?**

1. Acesse Settings no repositório GitHub.
2. Vá até **Secrets and variables** > **Actions**.
3. Adicione um novo segredo (exemplo: *API_KEY*).
4. No workflow, acesse o segredo:

```
env:
API_KEY: ${{ secrets.API_KEY }}
```

🔗 **Referência:** https://docs.github.com/en/actions/security-guides/encrypted-secrets

Boas Práticas no Uso de GitHub Actions

Fatec Jacareí
Prof. Francisco de Moura

5. Monitorar Logs de Execução para Identificar Falhas Rapidamente

- **Por que?**
  - Logs detalhados permitem diagnosticar falhas e melhorar a eficiência da automação.
  - Facilita a auditoria e resolução de problemas.
- **Exemplo real:** Um desenvolvedor notou que sua pipeline de deploy falhava esporadicamente. Ao verificar os logs no GitHub Actions, descobriu que a falha ocorria devido a **timeouts na instalação de pacotes**. A solução foi adicionar **retries** nas dependências.
- **Como aplicar?**
  - ❑ Acesse Actions no GitHub e clique no workflow desejado.
  - ❑ Verifique logs de cada etapa para entender o erro.
  - ❑ Use comandos para logs detalhados dentro do workflow:

    run: echo "Debugging Step - Variáveis: $GITHUB_ENV“
  - ❑ Se necessário, **habilite debugging avançado**:

    env:
    ACTIONS_RUNNER_DEBUG: true

🔗 **Referência:** https://docs.github.com/en/actions/monitoring-and-troubleshooting-workflows

INTEGRAÇÃO E ENTREGA CONTÍNUA

Atividade – Aula03

Bibliografia Básica

Fatec
Jacareí
Prof. Francisco de Moura

- 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

Fatec
Jacareí
Prof. Francisco de Moura

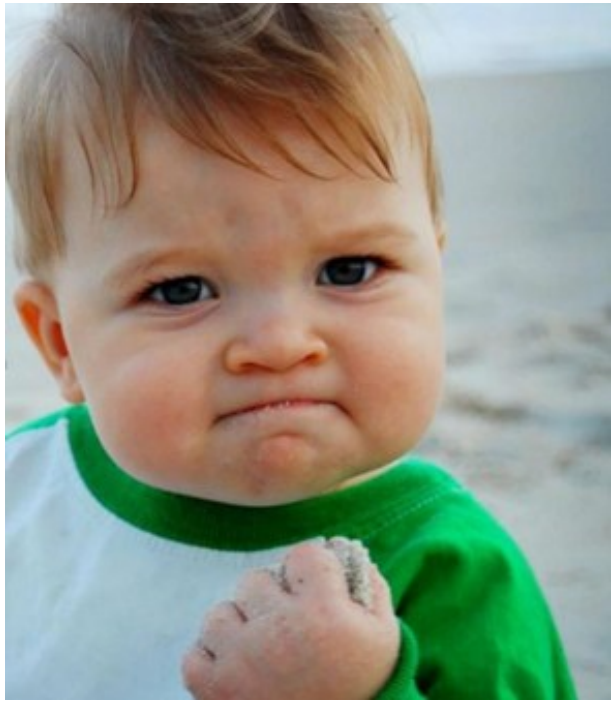
- ARUNDEL, J. DOMINGUS, J. **DevOps nativo de nuvem com Kubernetes**. São Paulo: Novatec, 2019.
- MORAES, G. **Caixa de Ferramentas DevOps:** Um guia para construção, administração e arquitetura de sistemas modernos. São Paulo: Casa do Código, 2015. PIRES, A.; MILITÃO, J. **Integração Contínua com Jenkins**. 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?

Fatec
Jacareí
Prof. Francisco de Moura

Considerações Finais

Fatec
Jacareí
Prof. Francisco de Moura



Professora:

Lucineide Pimenta

Bom descanso à todos!