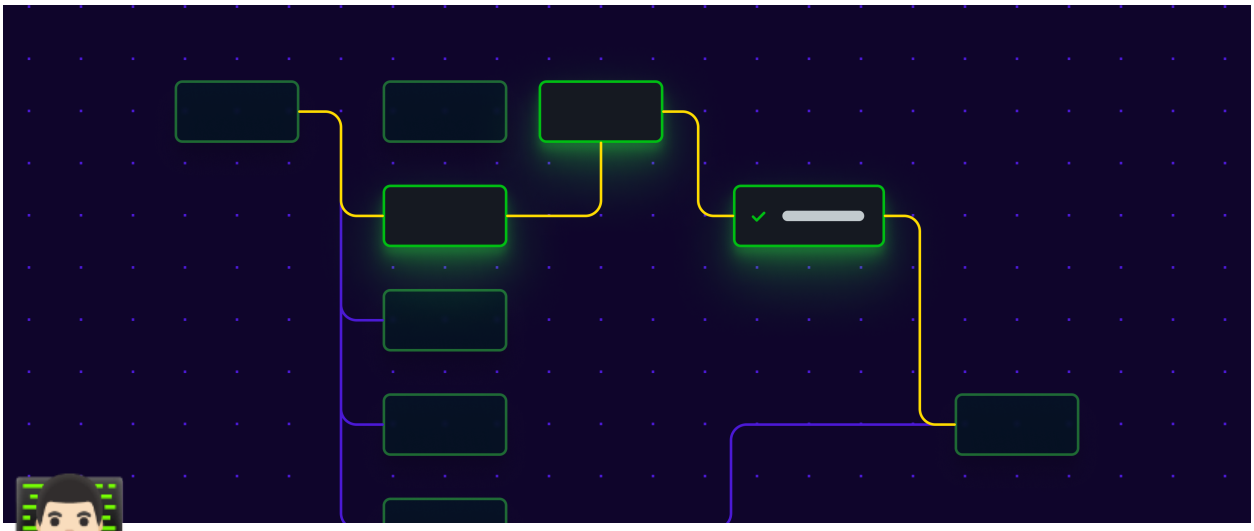


Estudos DevOps

GitLab CI e CD/ Pipelines

ℹ️*GitLab Pipelines é uma ferramenta de integração e entrega contínua (CI/CD) fornecida pelo GitLab, As pipelines do GitLab permitem que os desenvolvedores definam uma série de etapas automatizadas para construir, testar e implantar o código em ambientes de desenvolvimento, teste e produção.*

As pipelines do GitLab podem ser configuradas para serem executadas automaticamente quando um novo código é enviado para um repositório, ou manualmente pelo desenvolvedor. As etapas da pipeline podem ser definidas em arquivos YAML que ficam armazenados no repositório. O GitLab também oferece uma interface gráfica para visualizar o status das pipelines e dos testes, permitindo que os desenvolvedores identifiquem facilmente as falhas e corrijam problemas no código.

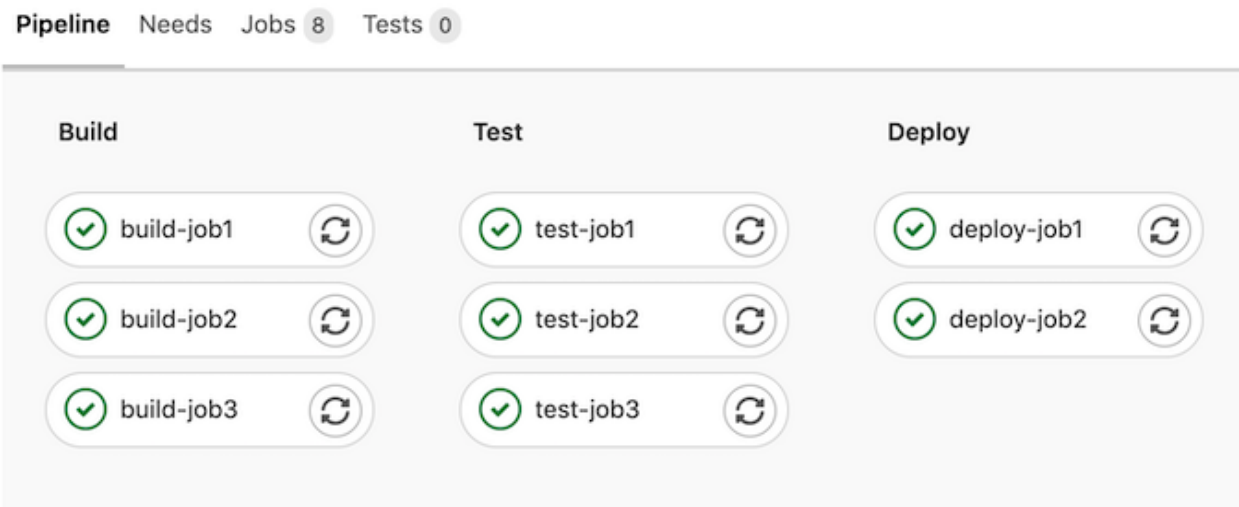
As pipelines do GitLab suportam a integração com diversas ferramentas de terceiros, como Kubernetes, Docker, Ansible, Selenium, entre outras, permitindo que os desenvolvedores construam pipelines personalizadas que se ajustem às suas necessidades. Além disso, as pipelines do GitLab também suportam a execução de

testes de segurança e monitoramento de performance, para garantir que o código seja seguro e eficiente.

.gitlab-ci.yml

O arquivo ".gitlab-ci.yml" é o arquivo de configuração de pipeline do GitLab, onde os desenvolvedores podem definir as etapas e configurações necessárias para construir, testar e implantar o código em um projeto GitLab. Esse arquivo é escrito em YAML e fica armazenado no repositório do GitLab.

Exemplo de Pipelene (T-Systems)



Nesse caso temos 3 jobs de build, 3 de teste e dois jobs de deploy de uma aplicação.

Abaixo um exemplo de um arquivo de esteira que seria somente o caso de um build em uma aplicação .NET Core:

```
variables:
  SolutionName: 'DCX.ITLC.WebAPI.sln'
```

```
stages:
  - build

Application-Build:
  stage: build
  tags:
    - MeuRunnerDeTeste
  script:
    - echo "Compilando a Solution..."
    - 'dotnet build "%SolutionName%"'
```

Na seção dos **Jobs** vamos entender como cada parte desse esteira de exemplo funciona.

Jobs

Um job é uma tarefa que deve ser executada. Por exemplo a compilação de uma aplicação ou o deploy.

Ele sempre vai estar dentro de um **estágio**. O exemplo a seguir mostra o build de uma aplicação em .NET Core:

```
variables:
  SolutionName: 'DCX.ITLC.WebAPI.sln'

stages:
  - build

Application-Build:
  stage: build
  tags:
    - MeuRunnerDeTeste
  script:
    - echo "Compilando a Solution..."
    - 'dotnet build' '%SolutionName%'
```

No caso do trecho acima, estamos declarando uma variável para o nome da *Solution* para a necessidade de reaproveitar em outros trechos da esteira. Logo após, temos o estágio, onde determinamos todos os estágios que devem existir na esteira. Em nosso caso há somente o *build* da aplicação. Por fim, o **Job** que colocamos o nome de

Application-Build. A primeira declaração que normalmente fazemos dentro do **Job** é a qual estágio ele pertence:

```
stage: build
```

Na sequência determinamos qual `Runner` queremos que execute esse **Job** para nós:

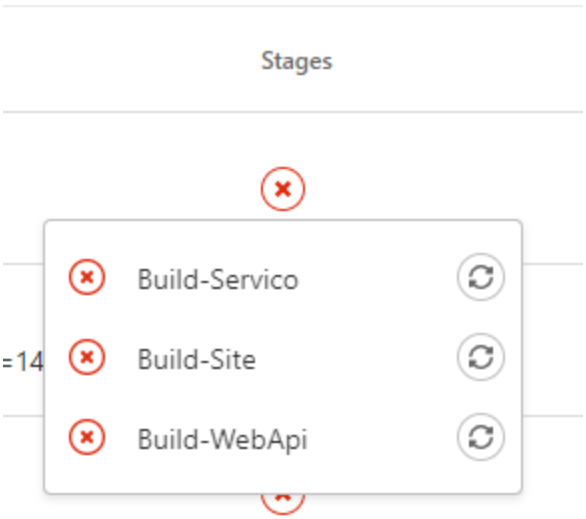
```
tags:
    - MeuRunnerDeTeste
```

E por fim, o que o **Job** vai executar:

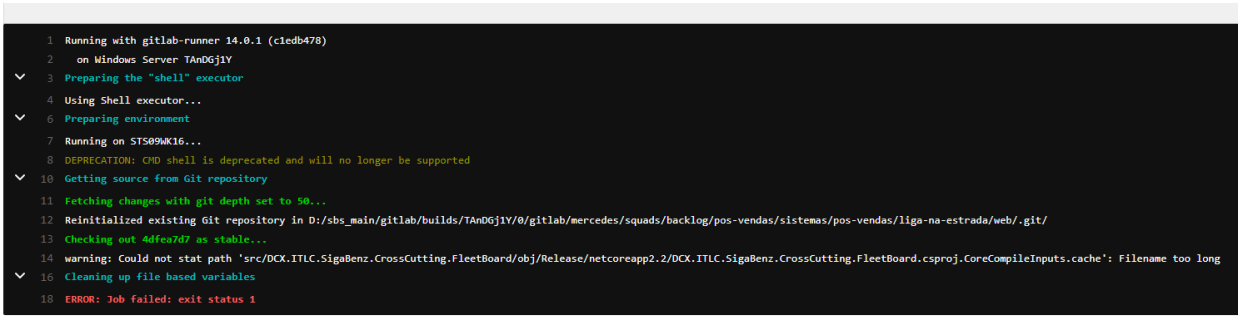
```
script:
    - echo "Compilando a Solution..."
    - 'dotnet build' '%SolutionName%'
```

🚨 Identificando problemas na Pipeline

Quando criamos as esteiras é comum durante a criação ficarmos no processo de tentativa e erro até que ela fique pronta. Por conta disso é importante sempre se atentar aos erros que ocorrem nos **Jobs**. Para visualizar os erros basta clicar em qual **Job** ocorreu o problema. Dentro do projeto no GitLab, basta ir no menu CI/CD ao lado esquerdo e selecionar a opção **Pipelines**. Nessa tela você vai ver em tempo real a **Pipeline** executando e quando houver o erro basta clicar no ícone do **Job** que falhou:



Selecionando o **Job** você vai ver a descrição do que aconteceu:



Nesse caso o **Job** falhou devido o caminho com o nome do projeto estarem muito grande.

Outra forma de identificar problemas na pipeline é através do **Editor de Pipelines**. Em qualquer projeto que você estiver, vá ao menu CI/CD -> Editor. Nessa tela você conseguirá ter uma visão completa da pipeline. Nela são apresentadas as falhas, onde o arquivo está com problema, a visão de todos os jobs existentes, um merge da pipeline caso os arquivos estejam separados e o LINT para verificar se a sintaxe da pipeline está correta.

fonte: https://sbs.t-systems.com.br/gitlab/mercedes/collab/planos-de-desenvolvimento/desenvolvimento-de-software/plano-de-formacao-inicial-backend/-/blob/stable/semanas/8.md