DevOps e Jenkins Pipeline: Conectando o Desenvolvimento a Entrega de Software!

Hello! - whoami

Daniel

DevOps @ Rivendel (não é cargo, eu sei)

Sysadmin há 12 anos

LPI-1/LPI-2 e Zabbix Specialist

Bacharel em CCP / Mestre em Eng. da Computação.

Barista Júnior.

Agenda

- Pipelines + DevOPS
- Jenkins
- Ambiente e aplicação
- Cenario 1
- Cenario 2
- Cenario 3
- Por que parar por aqui?
- Perguntas fáceis.

Pipelines + DevOps

Definições:

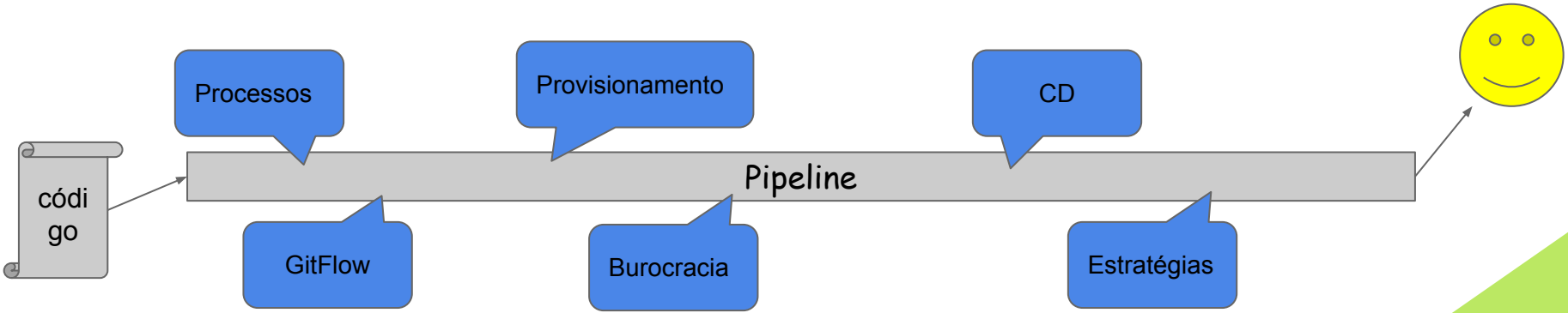
Hippie: “Uma pipeline é onde a manifestação do DevOPS realmente acontece.”

Rosa Weber: “segundo Friedrich Carl Freiherr (1815), uma pipeline, é a união completa dos Altos empresariais , lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation…” (+4 páginas)

Pipelines + DevOps

Objetivo:

- Automatizar o processo de entrega de software em produção de forma rápida e garantindo sua qualidade, estabilidade e resiliência.



Pipelines + DevOps

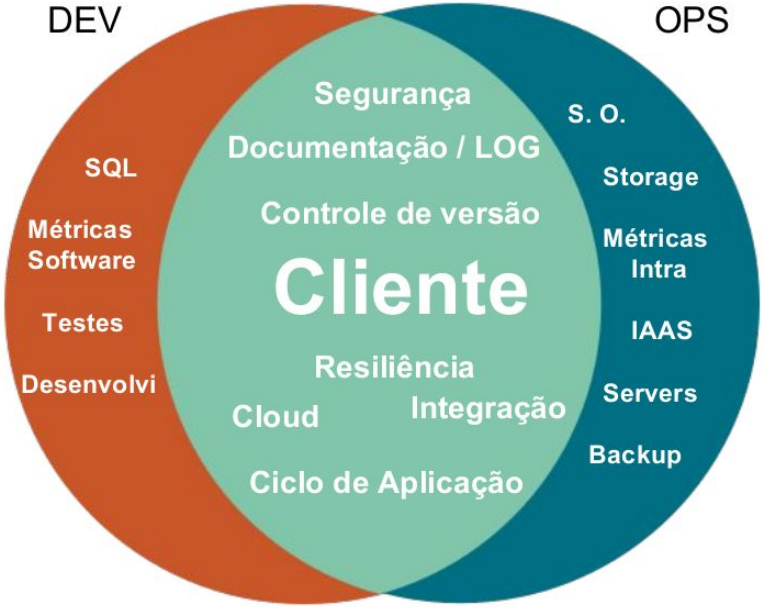
Alinhado com o CAMS.

- C - cultura
- A - automação (*)
- M - métricas
- S - compartilhamento (sharing)

Pipelines + DevOps

DevOPS:

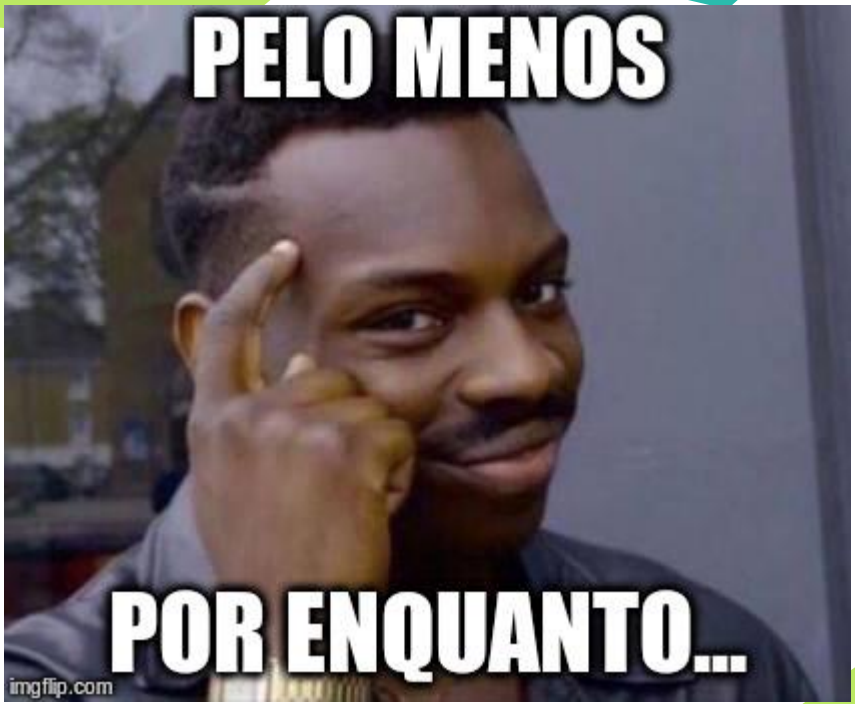
Pipelines + DevOps



Pipelines + DevOps

DevOPS:

- DEV não vira OPS
- OPS não vira DEV



Pipelines + DevOps

Importante!

- Pipeline é reflexo dos seus processos
  - Processo “torto” gera pipeline “torta”

- Software sem teste
  - Parabéns, você conseguiu automatizar seu Rollback

Jenkins

Por que?

- Mais antigo no "mercado"
- Orientado a plugins (extensível)
- Muito conhecido e utilizado
- Muito flexivel
- Farta documentação e material (livros, cursos, posts)
- Oferece versão com suporte (para quem interessar)

Jenkins

Considerações

- Pipeline: Declarativa X Script

■ Declarativa	■ Script
• Mais recente	• Mais flexível
• Não precisa aprender Groovy	• Groovy
• Em processo de adoção	• Mais poderosa
• Documentação mais escassa	• Muito utilizada

Jenkins

- Pipeline: Declarativa

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                sh './gradlew build'
            }
        }
        stage('Test') {
            steps {
                sh './gradlew check'
            }
        }
    }

    post {
        always {
            archiveArtifacts artifacts: 'build/libs/**/*.jar', fingerprint: true
            junit 'build/reports/**/*.xml'
        }
    }
}
```

Jenkins

Considerações

- Plugins
  - Extendem as funcionalidades básicas do Jenkins
  - Ferramental padronizado para pipelines
  - Abaixo do “guarda-chuva” do Java/Jenkins
  - Ajuda a evitar o uso do bom e velho “sh”

Ex:

- Amazon
- Docker
- Slack

Ambiente e Aplicação

Jenkins:

- Instalado na AWS em ec2 t2.micro
- Debian 9
- Versão 2.1XYZ
- Plugins (Gitlab, Slack, BlueOcean)
- Docker
- Utilitários do SO (curl, jq, etc…)

Gitlab:

- Na nuvem, com projetos públicos + webhooks

Ambiente e Aplicação

API Rest em Flask (python):

- MoviePoll
  - getmovies
  - getresult
  - vote
  - gettotalvotes (*)

Algum desenvolvedor Front JS?

Ambiente e Aplicação

Ambiente e Aplicação

ATENÇÃO

Cenario 1

- Desenvolvedor único
- tudo na master (sem branches)
- Sem testes
- Execução direta (CD) - Docker
- FreeStyle vs Coded
- Notificações

Cenario 1

- app.py
- Dockerfile
- Jenkinsfile

Fluxo:

- Exec FreeStyle Job: Clone-> Build-> Run.
- Dev -> commit master -> webhook p/ Jenkins -> Clone-> Build-> Run

Cenario 1

- Gitlab Webhook env VARs

gitlabMergeRequestDescription
gitlabMergeRequestTargetProjectId
gitlabSourceRepoURL
gitlabTargetRepoHttpUrl
gitlabUserEmail
gitlabMergeRequestTitle
gitlabTargetBranch
gitlabTargetRepoSshUrl
gitlabMergeRequestLastCommit
gitlabTargetNamespace
gitlabTargetRepoName
gitlabSourceNamespace
gitlabSourceRepoHttpUrl

gitlabMergeRequestState
gitlabUserName
gitlabSourceRepoName
gitlabActionType
gitlabSourceBranch
gitlabSourceRepoHomepage
gitlabMergedByUser
gitlabMergeRequestIid
gitlabBranch
gitlabMergeRequestId
gitlabSourceRepoSshUrl
...

Cenario 1

DEMO-1

Cenario 2

- + 1 Dev no time
- Gitflow mínimo (FB)
- Testes unitários e de integração
- Ambiente isolado para testes
- Notificações + completas

Cenario 2

- Jenkinsfile
- unit.sh
- Int.sh

Fluxo:

- Branch -> Push -> Clone -> Build -> Unit tests -> (if master) Int tests -> NEW Env.

Cenario 2

DEMO-2

Cenario 3

- Testes de Acc (em job separado) - Job Parametrizado
- Gitflow com MR
- Ambiente isolado para testes
- Notificações + completas

Cenario 3

- Jenkinsfile
- unit.sh
- Int.sh
- Nova Pipeline (Acc tests)

Fluxos:

- Branch -> Push -> Clone -> Build -> Unit tests -> Int tests -> **Acc tests** -> NEW Env.
- **MR p/ master** -> Clone -> Build -> Unit tests -> Int tests -> **Acc tests** -> PROD

Cenario 3

DEMO-3

Por que parar por aqui?

- Criação de repositório e Webhook
- Criação de Pipeline
- Setup de sistemas auxiliares
- Escolha de tecnologia (PaaS)
- Subida da Infra
  - Terraform
  - Ansible

Por que parar por aqui?

Referências

- https://gitlab.com/requena1/flask-app
- https://gitlab.com/my_pipelines/flask-app
- https://gitlab.com/my_pipelines/acctests
- https://jenkins.io/doc/book/pipeline/
- https://opendevops.com.br/

Obrigado!

Perguntas?

You can find me at @Daniel_Requena (twitter)

www.linkedin.com/in/danielrequena/

daniel.requena@rivendel.com.br

(11) 3447-1144