MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
ENGENHARIA DE COMPUTAÇÃO - SE/8

Luan Ferreira Cardoso, Ricardo Sollon Zalla, Venicius Gonçalves da Rocha Junior

DevOps: aproximando a área de desenvolvimento da operacional

Rio de Janeiro
7 de maio de 2016

Luan Ferreira Cardoso, Ricardo Sollon Zalla, Venicius Gonçalves da Rocha Junior

DevOps: aproximando a área de desenvolvimento da operacional

Trabalho apresentado ao Curso de Engenharia de Computação do Instituto Militar de Engenharia como Verificação Especial do Projeto de Fim de Curso.

Instituto Militar de Engenharia

Orientador: Clayton Escouper das Chagas

Coorientador: Coorientador ?????

Rio de Janeiro

7 de maio de 2016

Instituto Militar de Engenharia
Praça General Tibúrcio, 80 - Praia Vermelha
Rio de Janeiro - RJ CEP: 22290-270

Cardoso, Luan; Zalla, Ricardo e Gonçalves, Venicius
S586d DevOps: aproximando a área de desenvolvimento da operacional / Luan Ferreira Cardoso, Ricardo Sollon Zalla, Venicius Gonçalves da Rocha Junior. - Rio de Janeiro: Instituto Militar de Engenharia, 2016.

19f. : il., graf., tab. : -cm.

Projeto de Fim de Curso - Instituto Militar de Engenharia
Orientador: Clayton Escouper das Chagas.

1 - DevOps 2 - Desenvolvimento e Operação

CDU ???.???.??

Luan Ferreira Cardoso, Ricardo Sollon Zalla, Venicius Gonçalves da Rocha Junior

DevOps: aproximando a área de desenvolvimento da operacional

Trabalho apresentado ao Curso de Engenharia de Computação do Instituto Militar de Engenharia como Verificação Especial do Projeto de Fim de Curso.

Trabalho aprovado. Rio de Janeiro, 7 de maio de 2016:

Prof. Clayton Escouper das Chagas
Orientador, D. Sc., do IME

Prof. Humberto
Convidado, M. c., do IME

Prof. Chorem
Convidado, D. c., do IME

Rio de Janeiro

7 de maio de 2016

Sumário

1 INTRODUÇÃO 9
1.1 Motivação 9
1.2 Objetivo 9
1.3 Justificativa 10
1.4 Metodologia 10
1.5 Estrutura 10

2 FERRAMENTAS DEVOPS 12
2.1 Bancos de dados 12
2.1.1 Oracle 12
2.1.2 MySQL 12
2.1.3 MSSQL 12
2.1.4 Postgresql 12
2.1.5 MongoDB 12
2.1.6 DB2 12
2.1.7 Cassandra 12
2.2 Integração contínua 12
2.2.1 Jenkins 12
2.2.2 Bamboo 12
2.2.3 Travis CI 12
2.2.4 Codeship 12
2.2.5 Snap CI 12
2.2.6 Circle CI 12
2.2.7 TeamCity 12
2.2.8 Shippable 12
2.2.9 CruiseControl 12
2.2.10 Continum 12
2.2.11 Continua CI 12
2.2.12 Gump 12
2.3 Deployment 12
2.3.1 Ssh 12
2.3.2 Deployment Manager 12
2.3.3 SmartFrog 12
2.3.4 Capistrano 12

2.4 Núvem, IaaS(Infrastructure as a Service), PaaS(Plataform as a Service) . . . 12
2.4.1 Amazon AWS . . . 12
2.4.2 Azure . . . 12
2.4.3 Heroku . . . 12
2.4.4 Rachspace . . . 12
2.5 Monitoramento . . . 12
2.5.1 Kibana . . . 12
2.5.2 New Relic . . . 12
2.5.3 Nagios . . . 12
2.5.4 Ganglia . . . 12
2.6 SMC . . . 12
2.6.1 Git . . . 12
2.6.2 Subversion . . . 12
2.6.3 Github . . . 12
2.6.4 Bitbucket . . . 12
2.7 Gerencia de repositórios . . . 12
2.7.1 Archiva . . . 12
2.7.2 Nexus . . . 12
2.7.3 Artifactory . . . 12
2.7.4 NuGet . . . 12
2.8 Configuração e provisionamento . . . 12
2.8.1 Chef . . . 12
2.8.2 Puppet . . . 12
2.8.3 Ansible . . . 12
2.8.4 Salt . . . 12
2.8.5 BladeLogic . . . 12
2.8.6 Vagrant . . . 12
2.8.7 TerraForm . . . 12
2.8.8 Cobbler . . . 12
2.8.9 Bcfg2 . . . 12
2.8.10 CFEngine . . . 12
2.9 Release Managiment . . . 12
2.9.1 XL Release . . . 12
2.9.2 UrbanCodeRelease . . . 12
2.10 Logging . . . 12
2.11 Build . . . 12
2.12 Testing . . . 12
2.13 Conteinerization . . . 12

2.14 Colaboration **12**
2.15 Security **12**

3 ESTRUTURAS DE DEVOPS COMPLETAS **13**
3.1 Modelo do cardoso!!!!! **13**
3.2 Modelo baseado no livro Caixa de Ferramentas DevOps **13**
3.2.1 Ferramentas básicas: SSH, Git 13
3.2.1.1 SSH 13
3.2.1.2 Git 13
3.2.2 Vagrant e Virtualbox 17
3.2.3 Ansible 17
3.2.4 Instalando Wordpress em uma máquina 17
3.3 Modelo do Zalla!!!!! **17**

4 CONCLUSÕES **18**

Referências **19**

Resumo

Resumo em pt

Palavras-chave: DevOps, desenvolvimento, operação, ambientes.

Abstract

Abstract in English

Keywords: DevOps, development, operation, environment.

1 Introdução

1.1 Motivação

Quando uma organização precisa de servidores e computadores, ou precisa desenvolver um software e liberá-lo para os usuários, ou ainda precisa de mais claboração e cominicação entre as equipes devido a peculiaridades de alguns projetos, surge a necessidade de instalação e configuração de sistemas operacionais, programas e serviços que entrarão em operação ao final do projeto. Essas situação, aparentemente simples do ponto de vista de um usuário comum que instala os programas convencionais de que precisa, se transforma em uma tarefa de configuração complexa e inviável de ser feita para organizações com um número de servidores e computadores muito elevado [1]. Essa demanda por ativos computacionais pode variar muito dependendo do serviço oferecido pela organização, pode crescer dia a dia ou apresentar picos sob uma demanda específica, e para se otimizar a relação entre custo benefício, e para possibilitar uma entrega contínua e confiável [2], se faz necessária a capacidade de automatizar o processo de desenvolvimento e implantação de tais sistemas computacionais quando for necessário.

Assim, o processo de instalação dos sistemas operacionais e dos aplicativos se torna árduo e envolve tarefas trabalhosas e repetitivas para os administradores e desenvolvedores [3]. Nesse cenário, surgiu uma tendência de tentar criar estruturas automatizadas que pudessem facilitar a integração desses processos de desenvolvimento de sistemas [1], englobando todas as fazes do processo de desenvolvimento de softwares e sistemas.

A partir desse momento, os administradores não mas ficaram responsáveis por configurar e instalar sistemas de softwares, e passaram a investir seu tempo no desenvolvimento de ferramentas que automatizem todos os passos do processo. Nesse contexto, uma área chamada DevOps [4], que trata da integração de operação com desenvolvimento de sistemas vem se apresentando e se fortalecendo, a medida em que as demandas por estruturas de sistemas cada vez mais flexíveis e com menor custo vem crescendo.

1.2 Objetivo

Com o constante crescimento da comunidade DevOps, o suporte e o número de ferramentas e alternativas disponíveis estão aumentando constantemente. Assim, já é possível encontrar diversas ferramentas de DevOps, incluindo artigos, scripts e softwares, que po-

dem ser reusadas para automatizar o processo de colocar um software em produção como é possível observar em [5] e [6].

Esse trabalho tem como objetivo o desenvolvimento e a comparação de estruturas de DevOps. Cada estrutura dessa deverá conter uma das ferramentas usadas em cada fase do processo de desenvolvimento e implantação de software: Bancos de dados, Integração contínua, Colocar em produção (deployment), Núvem, IaaS(Infrastructure as a Service), PaaS(Plataform as a Service), BI, Monitoring, SMC, Gerencia de repositórios, Configuração e Provisionamento, Release Managiment, Logging, Build, Testing, Conteinerization, Colaboration, Security.

Assim, serão desenvolvidas estruturas dessas completas e funcionais, e serão feitas comparações com o objetivo de tentar determinar um parâmetro que possa ser útil na determinação de qual dessas estruturas se deve usar.

1.3 Justificativa

Para mostrar a importância desse trabalho, é possível citar alguns casos de sucesso da implementação da metodologia DevOps e analisar as melhorias que essa nova abordagem trouxe para essas organização.

Inicialmente, pode-se citar o grupo empresarial WOTIF GROUP que atua no comércio de viagens com uma plataforma na internet, segundo [7]. Em 2013 e 2014, a organização reorganizou o seu processo de liberação de softwares, reduzindo o tempo médio de liberação de software de semanas para horas, ratificando a importância dessa nova metodologia. Em resumo, uma das principais dificuldades encontradas pela empresa era que seus diversos departamentos de engenharia queriam colaborar nas fases de desenvolvimento de infraestrutura, de teste e de colocar em produção, mas não conseguiam encontrar uma maneira de fazer isso. Assim essa organização conseguiu resolver seus problemas utilizando as técnicas de DevOps e criando uma cadeia de ferramentas que atendeu às suas expectativas.

1.4 Metodologia

1.5 Estrutura

2 Ferramentas DevOps

2.1 Bancos de dados

2.1.1 Oracle

2.1.2 MySQL

2.1.3 MSSQL

2.1.4 Postgresql

2.1.5 MongoDB

2.1.6 DB2

2.1.7 Cassandra

2.2 Integração contínua

2.2.1 Jenkins

2.2.2 Bamboo

2.2.3 Travis CI

2.2.4 Codeship

2.2.5 Snap CI

2.2.6 Circle CI

2.2.7 TeamCity

2.2.8 Shippable

2.2.9 CruiseControl

2.2.10 Continum

2.2.11 Continua CI

2.2.12 Gump

2.3 Deployment

3 Estruturas de devops completas

3.1 Modelo do cardoso!!!!!

3.2 Modelo baseado no livro Caixa de Ferramentas DevOps

3.2.1 Ferramentas básicas: SSH, Git

3.2.1.1 SSH

É o protocolo utilizado para conectar-se de forma segura em servidores. Por meio de criptografia ele cria um canal seguro entre duas máquinas. Também pode ser utilizado para executar comandos remotamente sem entrar no shell da máquina. O Ansible, ferramenta que será usada mais afrente, só precisa de acesso por SSH.

Na maioria dos provedores, será encontrada uma opção para adicionar uma chave SSH pública. Para criar uma chave RSA local segue-se os seguintes passos:

1. Digite o comando a seguir:

```
1          ssh-keygen -t rsa
```

2. Quando for perguntado sobre uma frase digite enter duas vezes.
3. Serão criados dois arquivos:

```
1          id_rsa_devops e id_rsa_devops.pub
```

4. Os arquivos com extenção .pub são os únicos que devem ser copiados para outras máquinas e representam as chaves públicas.

3.2.1.2 Git

O Git é um sistema de controle de versão gratuita. Em termos práticos, sua função é controlar mudanças em repositórios de dados. Essa ferramenta controla repositórios de dados, pois, além de código-fonte de programas, também é possível controlar repositórios para a constução de livros, e qualquer outra atividade que em que várias pessoas trabalham ao paralelamente na sua construção.

Para instalar essa ferramenta, serão seguidos os seguintes passos:

1. Para linux baseado em debian:

```
sudo  apt-get install git-core
```

2. Para linux CentOS/RH

```
sudo  yum     install git
```

3. Para MacOSX com homebrew

```
brew  install git
```

Após instalar o Git, execute os seguintes comandos para configurar um nome de usuário e um email:

```
git config --global user.name "Seu nome"
git config --global user.email "seu@email.com.br"
```

Nesse momento o ambiente está com o Git instalado e com nome e e-mail configurados. Será adotado agora um sistema para controlar os repositórios usados para construir o modelo de DevOps proposto.

O “Github” é um site que fornece um serviço de controle de repositórios remotos de “Git” sem custo, caso o projeto seja aberto, e com custo caso o projeto seja privados. Para começar a utilizalo de forma gratuita, deve-se acessar o site “github.com” e criar uma conta. Neste ponto, será necessária a chave “ssh”, que está em:

```
~/.ssh/id_rsa.pub.
```

Agora, será criado um novo repositório que posteriormente será sincronizado com o repositório local que estará em cada máquina que estiver trabalhando no projeto. Para criar esse repositório, basta seguir os paços do site clicando no botão de novo repositório. Para esse projeto, será dado o nome de projeto-simples. Agora, será criado um diretório local e, posteriormente, será feita a associação desse diretório local com o repositório remoto criado anteriormente no GitHub. Para isso, deve-se seguir os passos listados abaixo:

1. Para criar o diretório local digite na linha de comando:

```
mkdir projeto-simples
```

2. Entre nesse diretório com:

```
cd projeto-simples
```

3. Com um editor de textos favorito, será criado um arquivo de exemplo, com nome de README.md, para poder ser sincronizado com o repositório remoto criado, explicando a função do projeto. Será colocado o seguinte texto no arquivo:

```
# README do meu projeto-simples
```

4. Esse projeto será apenas um shell script que conta itens únicos no diretório etc. Assim, será criaco o seguinte script com o nome de itens_unicos.sh:

```
#!/bin/sh
Echo "Itens unicos"
Ls /etc | cut -d" -f 1 | sort | uniq | wc -l
```

Nesse momento, já foi criado o projeto simples que será sincronizado com o repositório remoto no github. Agora seguiremos os passos a seguir para realizar essa sincronização:

1. Para iniciar o repositório Git no diretório local:

```
git init
```

2. Para adicionar todos os arquivos modificados ao conjunto de modificações que serão enviadas para serem sincronizadas com o repositório remoto:

```
git add .
```

3. Para criar uma mensagem, descrevendo as modificações:

```
git commit -m "mensagem descrevendo a altera  o"
```

4. Para vincular o repositório local ao repositório remoto:

```
git remote add origin git@github.com:veniciusgrjr/
   projeto-simples.git
```

5. Para enviar as alterações:

```
git push -u origin master
```

Após isso, se a página do repositório remoto for atualizada, os arquivos locais estarão lá. Será feito agora um pequeno resumo dos comandos do Git:

- Esse comando Inicia um repositório Git no diretório atual.

```
Git init
```

- Adiciona um ou mais arquivos para serem enviados (commit) ao repositório.

```
Git add.
```

- Confirma as mudanças e cria um commit com uma mensaApós isso, faça reload da página do repositório, e os arquivos locais estarão lá.

```
Git commit -m "mensagem"
```

- Esse comando adiciona um "remote" ao repositório atual, chamado origin. Você poderia trabalhar sem ter um remote, não é mandatório.

```
Git remote ...
```

- Envia (push) as modificações para o repositório remoto. O parametro -u só é necessário na primeira execução.

```
Git push -u origin master
```

- Cria uma cópia o repositório dado pela URL para a máquina local em que foi digitado.

```
Git clone
```

- Mostra o estado atual do repositório e das mudanças.

```
Git status
```

Agora, o diretório local será modificado e sincronizado com o repositório no Github. Para isso, será criado um arquivo com o nome de portas.sh, com o seguinte código:

```
#!/bin/sh
echo "Lista de porta 80 no netstat"
netstat -an | grep 80
```

Para adicionar essa modificação ao repositório no Github:

```
Git add portas.sh
Git commit -m "add portas.sh"
Git push origin master
```

3.2.2 Vagrant e Virtualbox

3.2.3 Ansible

3.2.4 Instalando Wordpress em uma máquina

3.3 Modelo do Zalla!!!!!

4 Conclusões

Texto Conclusão

Referências

1 HUMBLE, J.; MOLESKY, J. Why enterprises must adopt devops to enable continuous delivery. *Cutter IT Journal*, v. 24, n. 8, p. 6, 2011.

2 HUMBLE, J.; FARLEY, D. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Adobe Reader)*. [S.l.]: Pearson Education, 2010.

3 HTTERMANN, M. *DevOps for developers*. [S.l.]: Apress, 2012.

4 LOUKIDES, M. *What is DevOps?* [S.l.]: " O'Reilly Media, Inc.", 2012.

5 NELSON-SMITH, S. *Test-Driven Infrastructure with Chef: Bring Behavior-Driven Development to Infrastructure as Code*. [S.l.]: " O'Reilly Media, Inc.", 2013.

6 SABHARWAL, N.; WADHWA, M. *Automation through Chef Opscode: A Hands-on Approach to Chef*. [S.l.]: Apress, 2014.

7 CALLANAN, M.; SPILLANE, A. Devops: Making it easy to do the right thing. IEEE.