

# 4Linux também é consultoria, suporte e desenvolvimento.

- Você já deve nos conhecer pelos melhores cursos de linux do Brasil mas a 4Linux também é **consultoria, suporte e desenvolvimento** e executou alguns dos mais famosos projetos do Brasil utilizando tecnologias “open software”.
- Você sabia que quando um cidadão faz uma aposta nas loterias, saca dinheiro em um ATM (caixa eletrônico), recebe um SMS com o saldo de seu FGTS ou simula o valor de um financiamento imobiliário no “feirão” da casa própria, ele está usando uma infraestrutura baseada em softwares livres com consultoria e suporte da 4Linux?

## → Serviços

### Consultoria:

Definição de Arquitetura, tuning em banco de dados , práticas DEVOPS, metodologias de ensino on-line, soluções open source ( e-mail, monitoramento, servidor JEE), alocação de especialistas em momentos de crise e auditoria de segurança.

### Suporte Linux e Open Source:

Contratos com regimes de atendimento – preventivo e/ou corretivo - em horário comercial (8x5) ou de permanente sobre-aviso (24x7) para ambientes de missão crítica. Suporte emergencial para ambientes construídos por terceiros.

### Desenvolvimento de Software:

Customização visual e funcional de softwares Open Source, consultoria para a construção de ambiente ágeis de Integração Contínua (Java e PHP) e mentoria para uso de bibliotecas, plataformas e ambientes Open Source. Parceiro Oficial Zend.

## → Parceiros Estratégicos



## Saiba mais

(11) 2125-4769  
[www.4linux.com.br/suporte](http://www.4linux.com.br/suporte)  
[contato@4linux.com.br](mailto:contato@4linux.com.br)



**Curso: 4525**

**Infraestrutura ágil com práticas  
DEVOPS usando Docker, Git,  
Jenkins, Puppet e Ansible**

Versão: 4.2



## Sumário:

1. Infraestrutura Ágil - Infraestrutura como Código	2
2. Automação com <b>Ansible</b>	36
3. Administrando e Operando o <b>Docker</b>	84
4. Gerência de Configuração com <b>Puppet</b>	131
5. Versionsamento com <b>GIT</b>	175
6. Gerenciando Repositórios <b>Gitlab</b>	201
7. Continuous Delivery com <b>Rundeck</b>	230
8. Continuous Integration com <b>Jenkins</b>	270
9. Criando Alta Disponibilidade com <b>Nginx</b>	332
10. Criando Playbooks de Deploy <b>Blue Green</b>	342
11. Automação do Site da Dexter	350
12. Infraestrutura Cloud <b>AWS</b>	360

## Infraestrutura como Código

### Objetivos da Aula

- Compreender a Cultura DevOps;
- Entender a Metodologia de Desenvolvimento Ágil;
- Entender a Infraestrutura como Código;
- Explorar a Gerência de Configuração;
- Compreender Versionamento de Configuração;
- Trabalhar com Timeline;
- Trabalhar com Ambientes Transitórios;
- Trabalhar com Delivery Automático com Pipeline.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

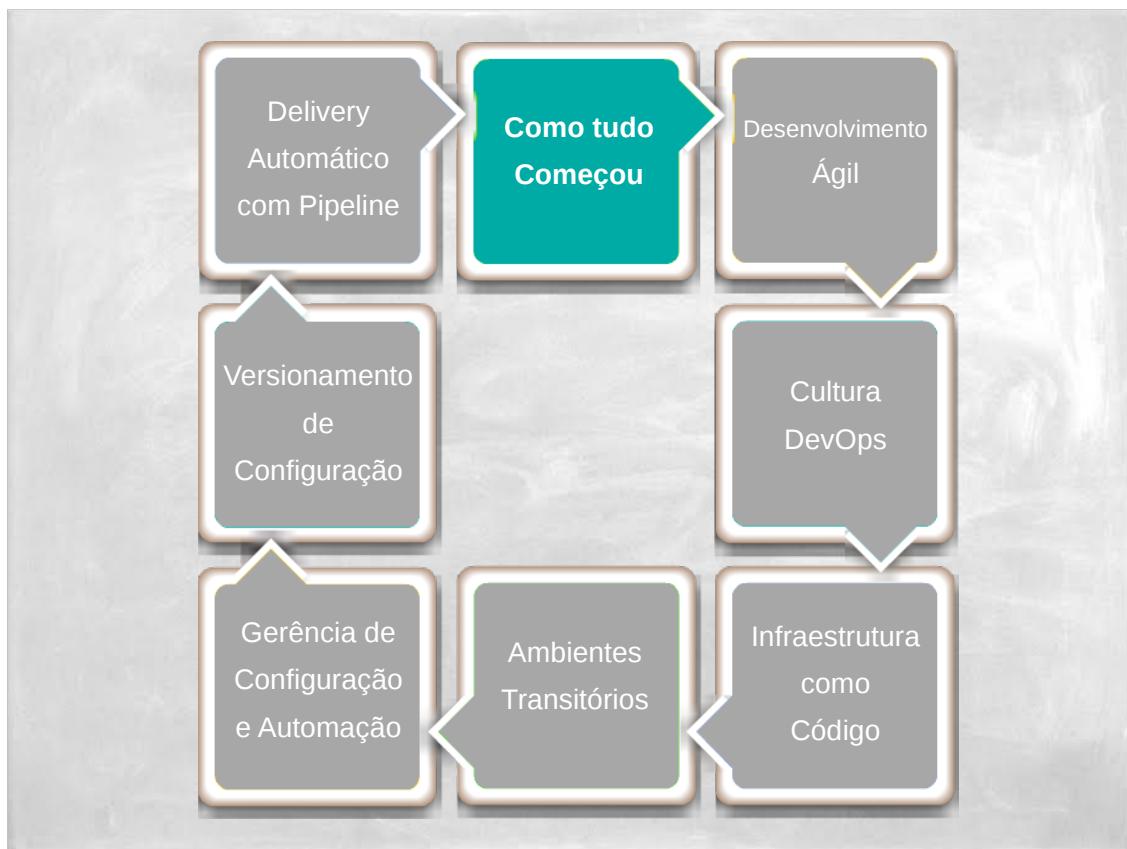
---

---

---

---

---



## Anotações:



Em fevereiro de 2001, um grupo de desenvolvedores se uniu para discutir sobre os padrões e metodologias utilizados no desenvolvimento de software. Durante o encontro foram abordadas questões que caracterizavam o sucesso dos projetos de software com os quais aqueles profissionais estiveram relacionados durante sua experiência profissional. Foi nesse encontro que teve o surgimento do Manifesto para **Desenvolvimento Ágil de Software**, ou, como ficou conhecido, **Manifesto Ágil**. Nesse documento foi apresentados os conceitos e fundamentos das metodologias de desenvolvimento Ágil.

O Manifesto Ágil afirma que melhores resultados no desenvolvimento de software podem ser obtidos através da valorização de:

- **Indivíduos e interações** mais que processos e ferramentas;
- **Software em funcionamento** mais que documentação abrangente;
- **Colaboração com o cliente** mais que negociação de contratos;
- **Responder a mudanças** mais que seguir um plano.



Em 2005 foi lançado o software de gerenciamento de configurações Puppet, desenvolvido por Luke Kanies e mantido atualmente pela empresa Puppet. Atualmente a ferramenta possui compatibilidade com diversas plataformas e é distribuído em duas edições: a versão Open Source e o Enterprise.

No mesmo ano, o relacionamento entre a comunidade que desenvolve o kernel Linux e a BitMover, empresa que desenvolve o sistema de versionamento BitKeeper, se desfez e o status de “isento de pagamento” da ferramenta foi revogado. Isso levou a comunidade (em particular Linus Torvalds, o criador do Linux) a desenvolver sua própria ferramenta baseada no BitKeeper, que posteriormente foi lançada com o nome Git. Alguns dos objetivos do novo sistema eram:

- Velocidade;
- Design simples;
- Suporte robusto a desenvolvimento não linear (milhares de branches paralelos) totalmente distribuído;
- Capacidade de lidar eficientemente com grandes projetos como o kernel do Linux (velocidade e volume de dados).

## DevOps - Linha do Tempo

# DEVOPSDAYS



O termo DevOps surgiu em 2009, porém a ideia nasceu anos antes durante a Agile Conference de 2008. Patrick Debois, um dos fundadores do termo, acabou conhecendo Andrew Shafer, um dos palestrantes do dia que iria apresentar a **Agile Infrastructure**. Pela conversa que ambos tiveram decidiram fundar um grupo aonde pudessem discutir mais sobre o assunto de Infraestrutura Ágil, assim nascendo o grupo **Agile System Administration** do Google Groups.

No ano seguinte, ocorreu a famosa palestra do Flickr, intitulada **10 Deploys a Day: Dev and Ops Cooperation at Flickr**. Debois comentou no Twitter que lamentava não estar presente na palestra, recebendo como resposta o tweet “Porque não organizar sua própria conferência de Velocity na Bélgica?”.

Por conta dessa mensagem, Devops decidiu criar sua própria conferência, assim surgindo o **DevOpsDays**, um evento que ocorreria por dois dias aonde poderiam discutir assuntos relacionados a desenvolvimento e infraestrutura, e a integração entre ambos. Para lembrar o dia, foi criado uma hashtag no twitter chamada **#DevOps**, uma abreviação do nome do evento, que acabou sendo usado como o nome da cultura aonde Desenvolvimento e Operações trabalham em conjunto.

## DevOps - Linha do Tempo



Em 2010, a empresa Zynga possuia um problema de deploy de aplicação, aonde a parte do desenvolvimento estava completamente automatizada, porém operação era necessário ainda executar certos passos de forma manual, ou desenvolver scripts que não eram reutilizáveis entre os projetos.

A solução encontrada foi desenvolver uma ferramenta que pudesse automatizar os deploys da área de operação, semelhante ao que o Jenkins (na época Hudson) fazia para a equipe de desenvolvimento. Assim surgiu o piloto do Rundeck, uma ferramenta Open Source para automação de serviços e agendamento de tarefas.



Em 2004 foi lançado o Hudson, uma ferramenta de integração contínua desenvolvida pela Sun Microsystems, e, posteriormente, adquirido pela Oracle. Em 2010, um problema surgiu sobre a infraestrutura utilizada e questões sobre a gestão de controle da Oracle. Embora houvesse vários pontos de concordância entre eles, uma ponto chave de discussão era o nome da marca.

Após a Oracle reivindicar o direito ao nome e aplicá-lo como uma marca registrada, em Janeiro de 2011, uma chamada de votos foi feita para o troca do nome de Hudson para o nome atual do projeto Jenkins. A Oracle divulgou que iria continuar o desenvolvimento do Hudson e considerou o Jenkins com um fork do projeto e não simplesmente uma troca de nome.

Nesse mesmo ano foi lançado a primeira versão do GitLab, uma ferramenta Open Source e gratuita de gerenciamento de repositórios git. Atualmente o projeto é utilizado por diversas companias que usam o git como versionador, como a IBM, Sony, SpaceX, CERN, e a NASA.



Em fevereiro de 2012, Michael DeHaan começou o desenvolvimento do Ansible. Rapidamente ele foi adotado por vários Desenvolvedores e Administradores de Sistema.

O time de infraestrutura do Fedora foi o primeiro a adotar o Ansible para gerenciar a configuração de seus servidores, também servindo como ambiente de teste para o crescimento da ferramenta.



No ano de 2015 o Docker, de certa forma, “explodiu” no mercado, tendo uma grande visibilidade entre as empresas ao redor do mundo pela facilidade, velocidade e utilização de recursos, além de tratar de containers.

Empresas como The New Work Times, Grupon, PayPal e Spotify utilizam o Docker na infraestrutura de seus produtos.



## Anotações:

## Desenvolvimento Ágil

**Desenvolvimento Ágil** de software, ou **Método Ágil**, é um conjunto de metodologias de desenvolvimento de software.



Alguns princípios da metodologia ágil são:

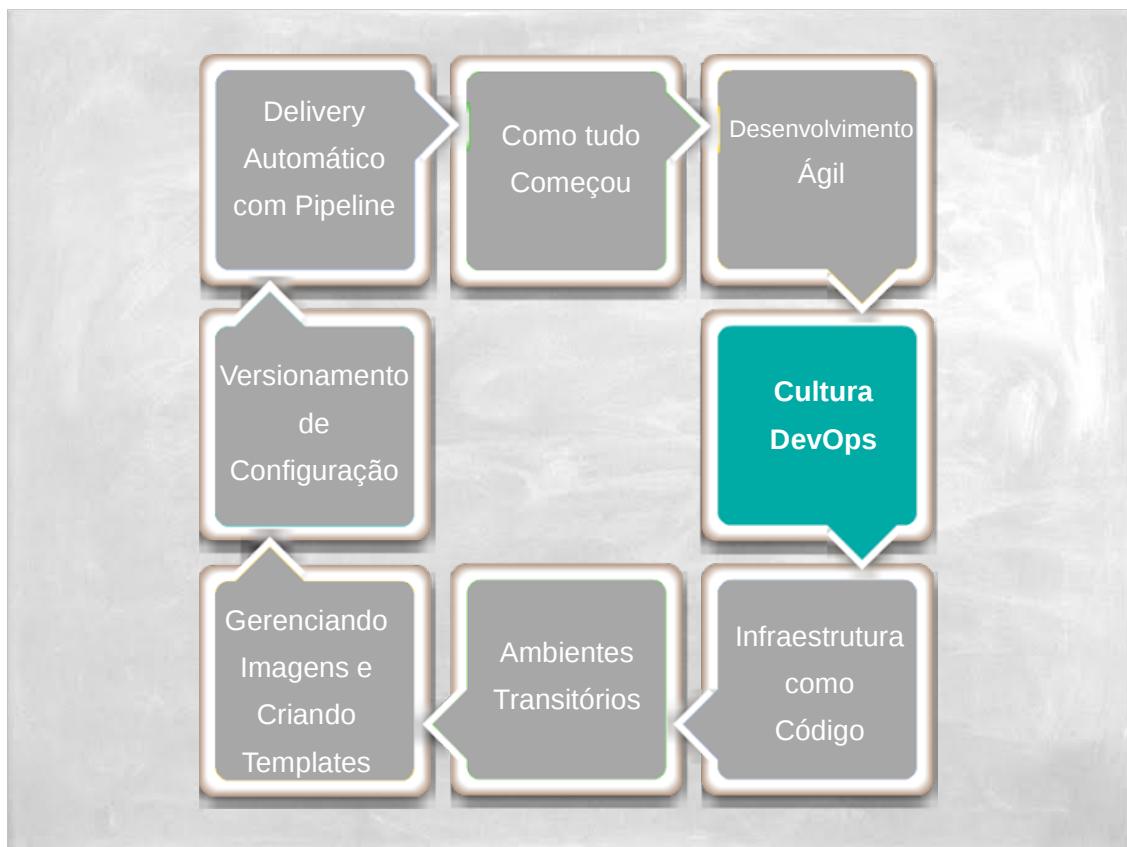
- Garantir a satisfação do consumidor entregando rapidamente e continuamente os softwares funcionais.
- Rápida adaptação a mudanças.
- Simplicidade.



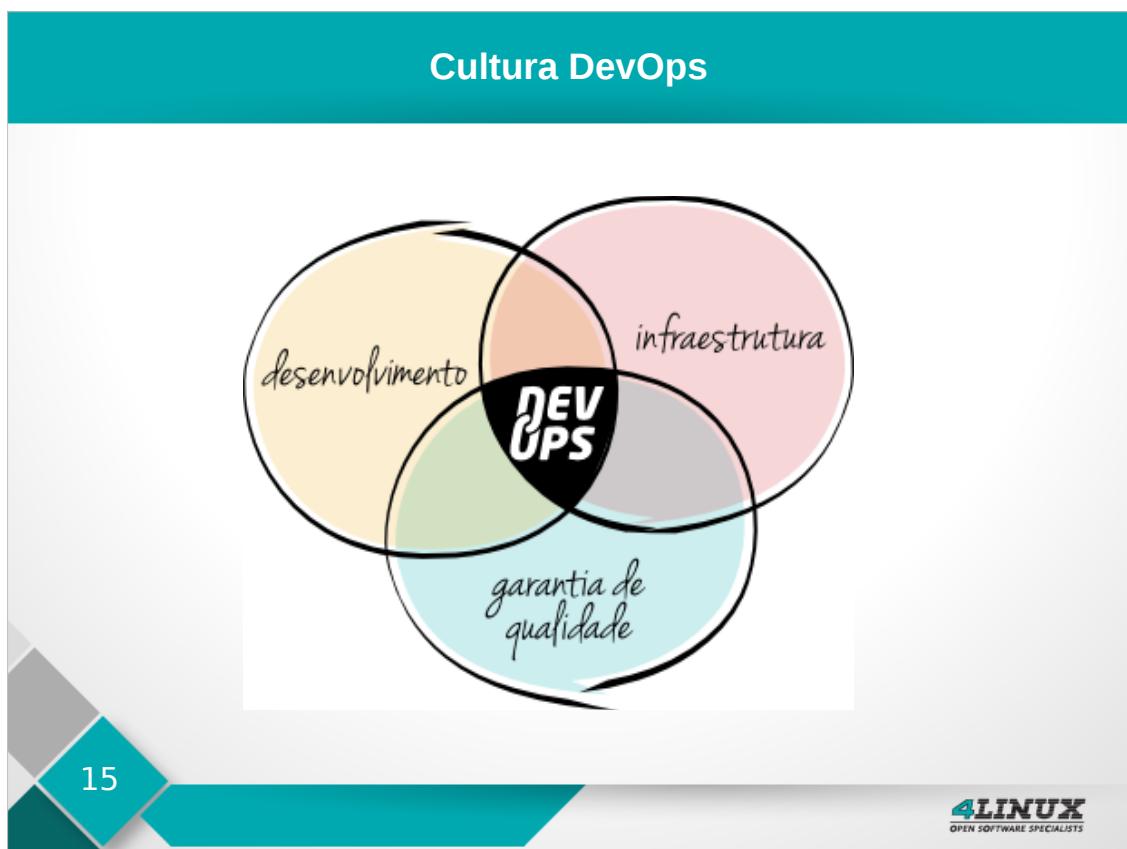
O desenvolvimento ágil, tal como qualquer metodologia de software, providencia uma estrutura conceitual para reger projetos de engenharia de software.

Outros princípios da metodologia ágil são:

- Software funcionais são a principal medida de progresso do projeto.
- Até mesmo mudanças tardias de escopo no projeto são bem vindas.
- Cooperação constante entre pessoas que entendem do negócio e desenvolvedores.
- O projeto surge através de indivíduos motivados, entre os quais existe relação de confiança.
- O design do software deve prezar pela excelência técnica.
- Simplicidade.



## Anotações:



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Cultura DevOps

**DevOps** (Development + Operation) é um processo de desenvolvimento e entrega de software que enfatiza a comunicação entre os profissionais de desenvolvimento e os do operacional (infraestrutura, rede, suporte).



Automatizar a entrega de software e mudanças na infraestrutura são práticas do mundo **DevOps**.



O Termo DevOps tem tomado uma grande importância no mundo todo. Empresas com Amazon, Netflix, Facebook, Walmart, Spotify entre tantas outras empresas, então investindo grandemente em funcionários, ferramentas e ambientes que sigam as práticas DevOps, para melhorar a entrega e qualidade de seus produtos.

Hoje a palavra DevOps também está sendo utilizada para classificar funcionários, departamentos e times dentro do ambiente empresarial. Funcionários que possuem o perfil DevOps têm qualidade para trabalhar tanto na área de Desenvolvimento de Software como na área de Administração de Sistema.

## Cultura DevOps

Objetivos de um ambiente com práticas **DevOps**:

- 1 Melhorar a frequência de Deploys;
- 2 Automatizar processos;
- 3 Diminuir a ocorrência de erros em novas versões;



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Cultura DevOps

Objetivos de um ambiente com práticas **DevOps**:

- 4 Curtos períodos de tempo para mudanças e melhorias;
- 5 Recuperação rápida em caso de falhas no ambiente;
- 6 Padronização nos processos de configuração e servidores;



18



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

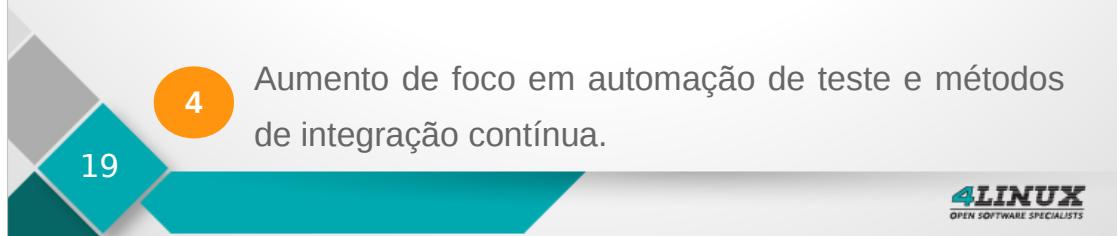
---

---

## Cultura DevOps

Fatores que estão impulsionando o uso de práticas **DevOps**:

- 1 Usar processos e metodologias ágeis.
- 2 Disponibilidade de tecnologias como Virtualização e Infraestrutura em Cloud.
- 3 Aumento do uso de ferramentas de automação e gerência de configuração.
- 4 Aumento de foco em automação de teste e métodos de integração contínua.



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

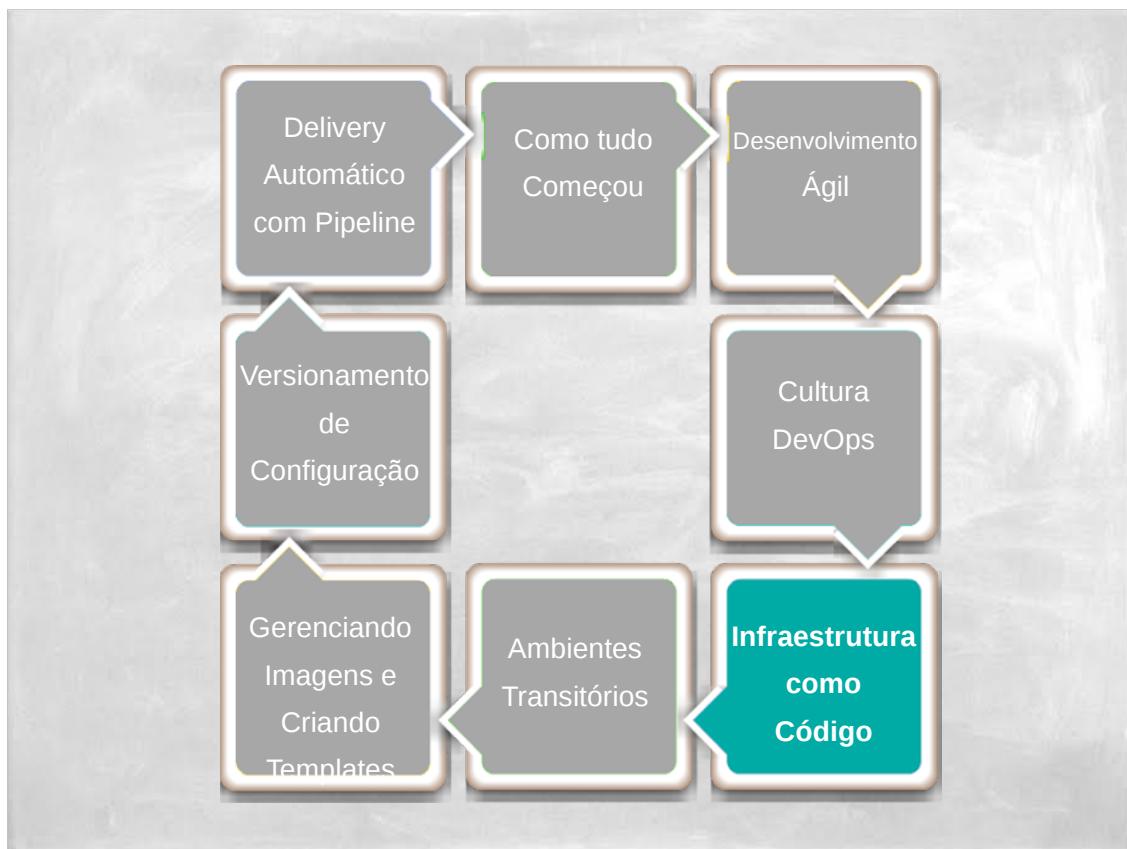
---

---

---

---

---



## Anotações:

## Infraestrutura como Código



Já imaginou sua infraestrutura sendo versionada e automatizada da mesma forma que acontece com o desenvolvimento de software?



**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

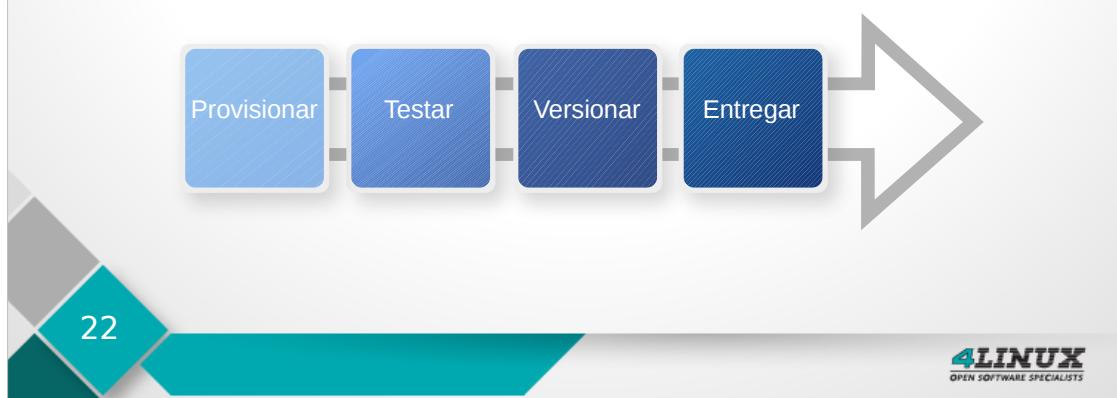
---

---

## Infraestrutura como Código

**Infraestrutura como código** (Infrastructure as Code) é uma forma de construir e gerenciar uma infraestrutura da mesma maneira que ocorre no desenvolvimento de software.

**Automatizar:**



**Infraestrutura como código** (Infrastructure as Code - IaC) é um processo de gerenciamento e provisionamento de infraestruturas, aonde todas as atividades e passos para sua criação e configuração são registrados em scripts utilizados por ferramentas de automação (como Ansible e Puppet), e passam pelo mesmo processo de versionamento que um software.

Com esse processo, torna-se muito mais fácil a gestão de ambientes, pois garante que novos serviços que forem incluídos possuirão as mesmas configurações e versões de pacotes. Outro ponto positivo é que o profissional de operação para de se preocupar em configurar cada servidor, e passa a se preocupar em codificar o ambiente, passando a atividade de implantação para a ferramenta.

## Infraestrutura como Código

Objetivos de uma **Infraestrutura como Código**:

- 1 Permitir mudanças, não sendo um obstáculo ou restrição.
- 2 Aproveitar o tempo com execuções valiosas, ao invés de desperdiçá-lo em tarefas rotineiras e repetitivas.
- 3 Recuperar rapidamente o ambiente em caso de falha ou perda total.
- 4 Possibilitar mudanças na infraestrutura de forma fácil e automatizada.

23



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Infraestrutura como Código

Objetivos de uma **Infraestrutura como Código**:

- 5 Realizar melhorias de forma contínua (Continuous Delivery).
- 6 Testar as mudanças antes que entrem em produção.
- 7 Garantir ambientes idênticos baseados na mesma configuração.



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

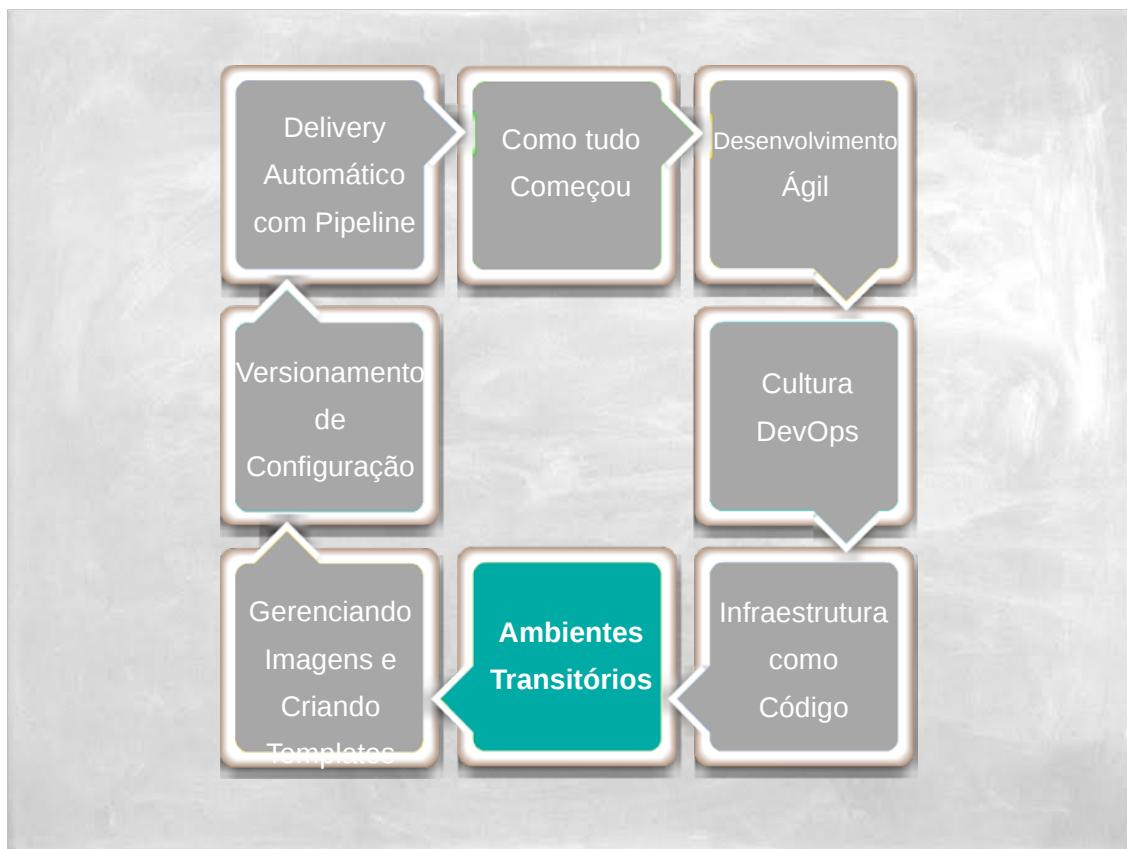
---

---

---

---

---



## Anotações:

## Ambientes Transitórios

A **Virtualização e Computação em Nuvem** (Cloud Computing), também disponibilizam a possibilidade de construir ambientes transitórios.

**Ambientes transitórios** são ambientes de curta duração que são terminados com uma certa frequência.



**Atenção:** Todos os ambientes são transitórios, exceto o ambiente de Produção.



26

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Ambientes Transitórios

### Recursos que Compõem os Ambientes Transitórios

Ambientes Baseados em Scripts	Ambiente de Autoatendimento	Término Automático
São completamente baseados em scripts, atribuídos em versões e testados.	Qualquer pessoa autorizada pode ativar um novo ambiente.	Os ambientes são automaticamente finalizados, de acordo com a política determinada na empresa.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Ambientes Transitórios

### Vantagens na Utilização de Ambientes Transitórios

#### Melhor Utilização de Recurso

Após a finalização de um ambiente, serão liberados os recursos para os demais ambientes.

#### Reducir a dependência do ambiente

Reduz a dependência de ambientes específicos fornecendo o recurso para ativá-los e terminá-los de acordo com a necessidade.

28



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



## Anotações:

## Gerência de Configuração e Automação

Uma **Infraestrutura Ágil** requer automação e gerenciamento de configuração do seu ambiente.



Utilizando o Puppet, Ansible ou Chef, podemos provisionar e gerenciar um parque de máquinas de forma automatizada e escalável.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Gerência de Configuração e Automação

### Vantagens da Gerência de Configuração

- 1 Padronização da Infraestrutura.
- 2 Automatização de tarefas.
- 3 Maior controle e integridade.
- 4 Velocidade nas mudanças e alterações.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

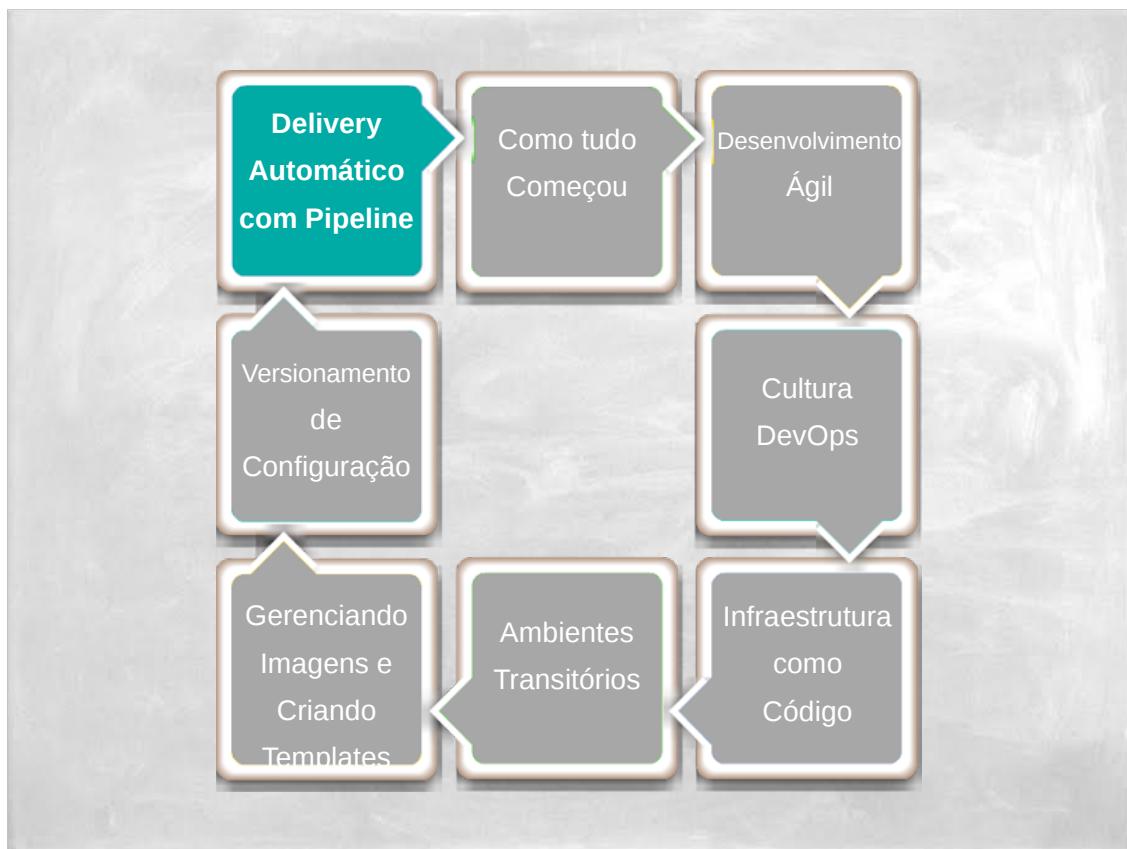
---

---

---

---

---



## Anotações:

## Versionamento de Configuração

Se tivermos uma infraestrutura toda baseada em código e scripts podemos versioná-la da mesma forma que versionamos os softwares:

1 Rastreamento de Mudanças.

2 Marcações e Resgate de Versões Estáveis.

3 Controle de Histórico.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Versionamento de Configuração

No nosso laboratório iremos versionar os módulos do **Puppet**, sendo que cada módulo será responsável por um serviço dentro da infra da Dexter. O **Puppet** será responsável por criar e garantir a infraestrutura dos serviços utilizados de forma automatizada.



34

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

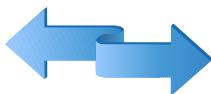
---

---

---

## Delivery Automático com Pipeline

Delivery  
Automático  
com Pipeline



Entregar o serviços da infraestrutura baseando-se em Pipeline de Entregas com Deploy e teste de configuração automatizados.

No nosso laboratório iremos utilizar o **Ansible** para automatizar o deploy das configurações e o **Jenkins** para integrar homologação e produção.



**4LINUX**  
OPEN SOFTWARE SPECIALISTS

35

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Automação com Ansible

### Objetivos da Aula

- Fazer uma introdução ao Ansible;
- Preparar o Ambiente;
- Instalar o Ansible;
- Fazer o inventário do Ansible;
- Trabalhar com Formato YAML;
- Compreender Módulos do Ansible;
- Criar Playbooks;
- Criar Roles.



37



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

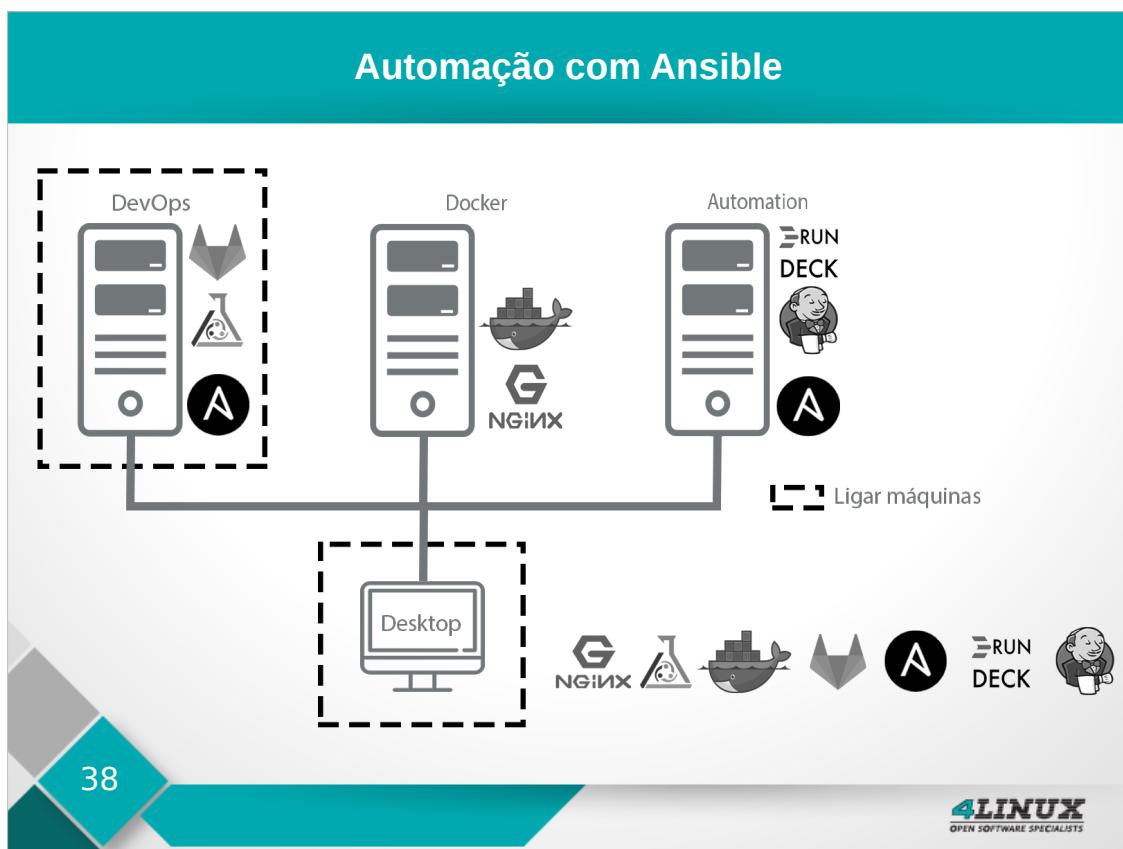
---

---

---

---

---



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

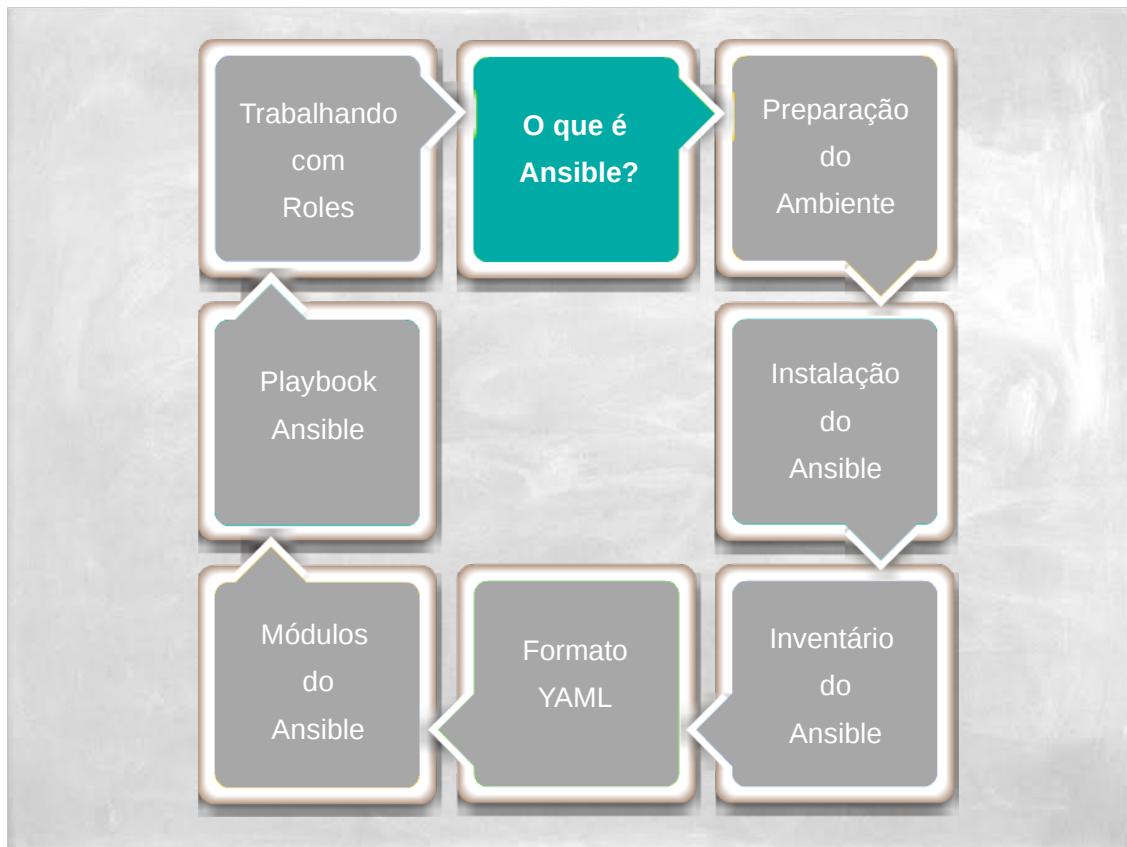
---

---

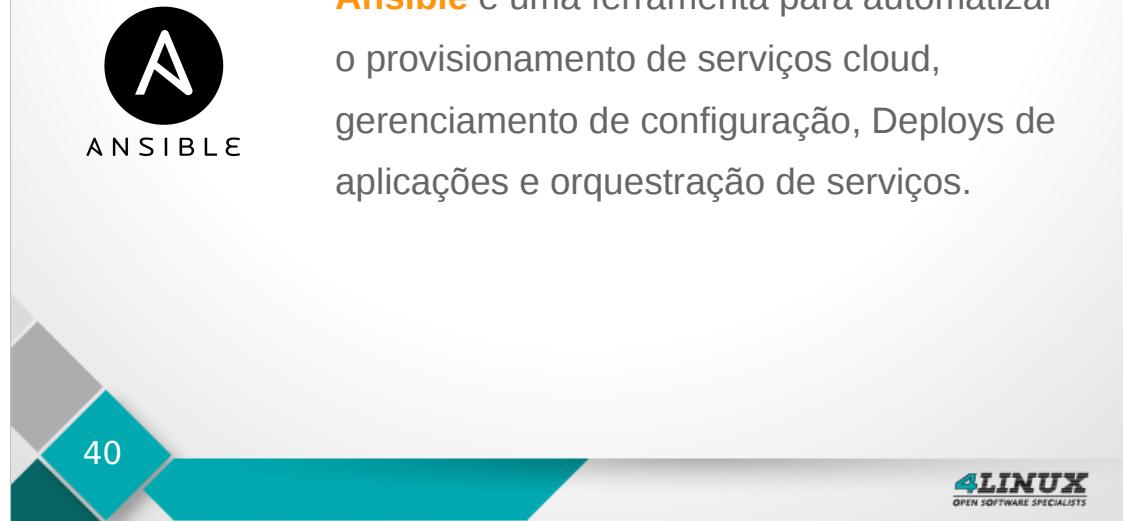
---

---

---



## Introdução ao Ansible



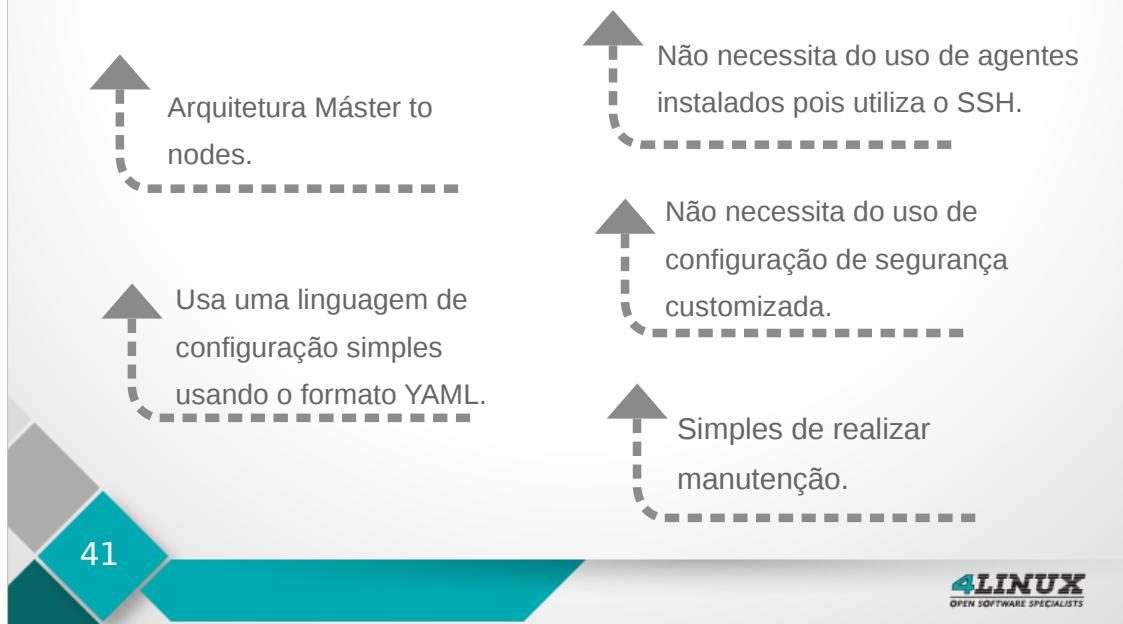
O Ansible foi desenvolvido pensando na simplicidade e facilidade de uso. Este é um dos fatores que está levando os Administradores de Sistema a escolher o Ansible para gerenciar o parque de máquinas. Outro foco do Ansible é a segurança e confiabilidade, utilizando o OpenSSH para transporte e comunicação dos nodes.

A simplicidade ajuda em ambientes de qualquer tamanho, sejam eles ambientes de empresas pequenas ou ambientes de empresas multinacionais, se tornando fácil o gerenciamento das máquinas e tornando possível a qualquer funcionário de TI ler as tarefas do Ansible de forma simples.

O Ansible gera as máquinas de uma forma sem agente (agente-less) pelo openssh. O OpenSSH está presente em todas as distribuições linux. O Ansible é descentralizado e depende somente das credenciais OS existentes para controlar o ambiente. Também podemos utilizar Kerberos ou LDAP para gerenciamento de autenticações centralizadas.

## Introdução ao Ansible

### Características do Ansible



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

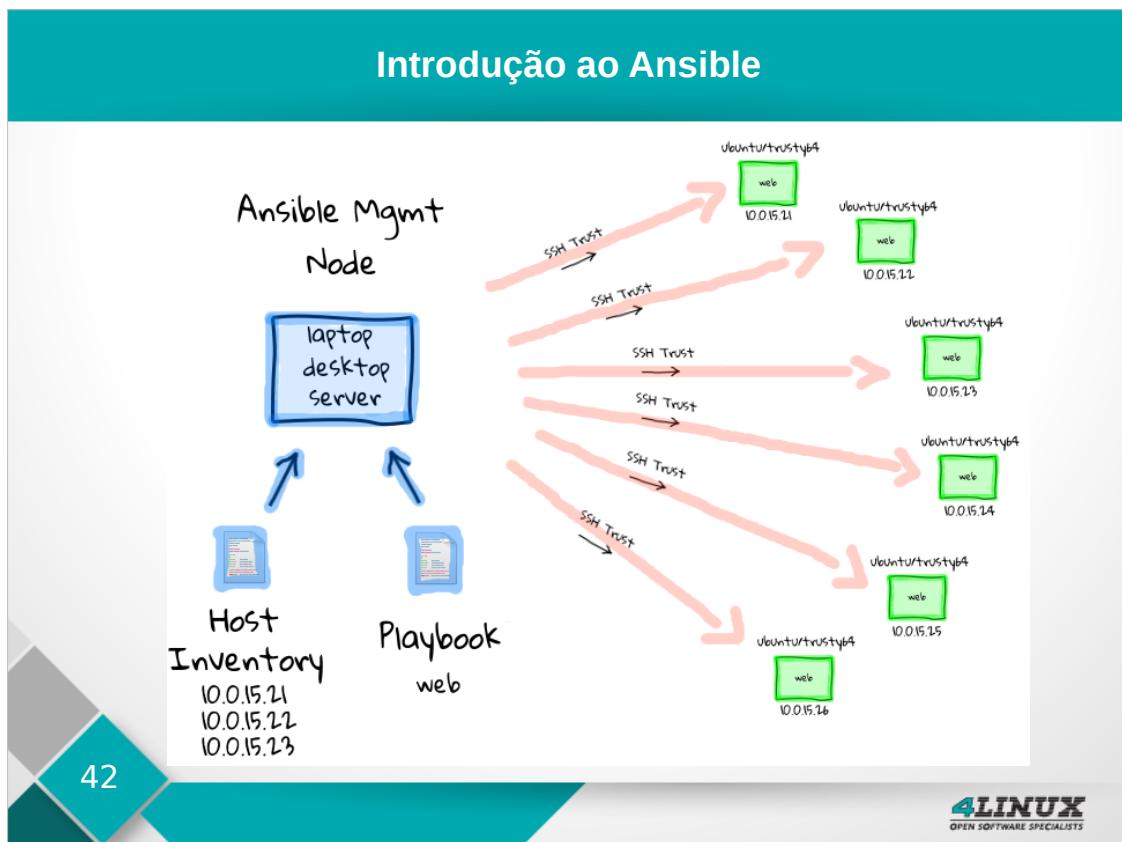
---

---

---

---

---



Ansible se conecta no nodes especificados e envia pequenos programas, que são chamados de Módulos do Ansible. Esse programas são escritos para controlar os recursos e estados do sistema. Ansible executa os programas no servidor utilizando SSH por padrão e remove o módulo enviado após o término da execução.

A bibliotecas de módulos pode ser armazenada em qualquer máquina, e não precisa de nenhum servidor, damons ou banco de dados.



## Anotações:

## Preparação do Ambiente

Máquina: Devops

Crie o diretório **/etc/keys**.

Vamos criar o diretório, onde serão armazenadas as chaves utilizadas pelo ansible para acesso aos nodes sem senha.

1

```
1# mkdir /etc/keys
```

44



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Preparação do Ambiente

Máquina: Devops

2

Vamos gerar o par de chaves no diretório criado, definindo a chave privada como "key.pem".

```
1# ssh-keygen
Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id_rsa): /etc/keys/key.pem
Enter passphrase (empty for no passphrase): <enter>
Enter same passphrase again: <enter>
```

45



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Preparação do Ambiente

Máquina: Devops

3

Utilizando o comando “ssh-copy-id”, vamos adicionar nossa chave pública nos servidores que serão acessados pelo ansible.

```
1# ssh-copy-id -i /etc/keys/key.pem.pub root@192.168.200.10  
2# ssh-copy-id -i /etc/keys/key.pem.pub root@192.168.200.50  
3# ssh-copy-id -i /etc/keys/key.pem.pub root@192.168.200.100
```

4

Vamos testar o acesso SSH entre as máquinas.

```
4# ssh -i /etc/keys/key.pem root@192.168.200.10  
5# ssh -i /etc/keys/key.pem root@192.168.200.50  
6# ssh -i /etc/keys/key.pem root@192.168.200.100
```

46



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



## Anotações:

## Instalação do Ansible

Máquina: Devops

- 1 Instale o Ansible.

```
1# apt-get install ansible
```

- 2 Crie a estrutura de diretórios que iremos utilizar no laboratório Dexter.

```
2# cd /etc/ansible
```

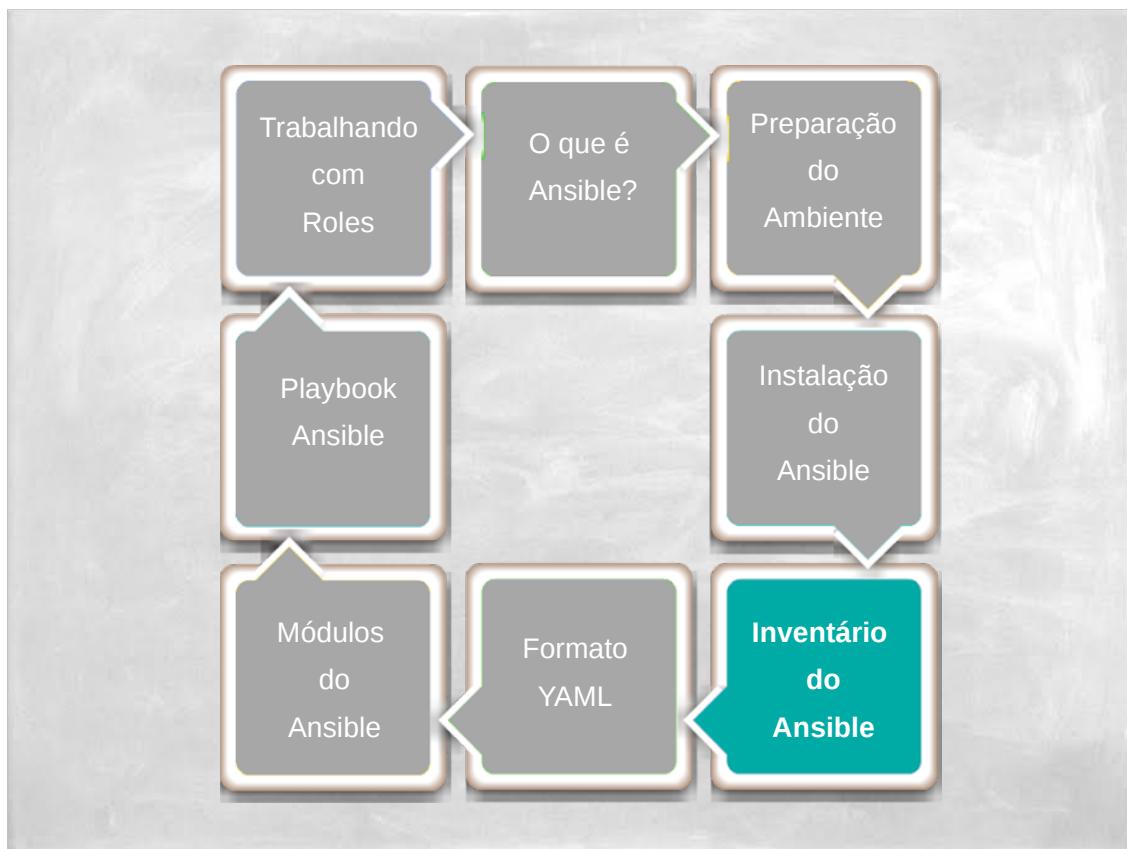
```
3# mkdir playbooks roles
```



48



O Ansible está disponível nos repositórios das principais distribuições do Linux, porém fica atento na versão que será instalada na sua distribuição. Na documentação online do Ansible você pode consultar a versão dos módulos e parâmetros que são suportados.



## Anotações:

## Inventário do Ansible

O arquivo de **Inventário (Inventory) do Ansible** armazena a lista de servidores gerenciados e acessados por ele. Por padrão, o arquivo de configuração é o **/etc/ansible/hosts**. Caso haja necessidade, podemos alterar este caminho no arquivo de configuração "/etc/ansible/ansible.cfg".

Dentro do arquivo de inventário podemos definir um host e também definir um grupo de hosts utilizando as seções separadas por "[]".



50



Exemplo de um arquivo de inventário.

```
mail.dexter.com.br

[webservers]
node01.dexter.com.br
node02.dexter.com.br    maxClient=709    user=torvalds

[dbservers]
db1.dexter.com.br    ansible_ssh_private_key_file=/etc/keys/db1.pem
db2.dexter.com.br    ansible_ssh_private_key_file=/etc/keys/db2.pem
```

No arquivo de inventário, podemos especificar uma máquina colocando o hostname ou endereço ip, ou podemos especificar um grupo utilizando as chaves. No exemplo acima, podemos escrever um playbook especificando os grupos **webservers** e todos os módulos especificando o que será executado nos dois servidores(node01 e node02).

Também podemos especificar variáveis pelo inventário e passar essas variáveis de acordo com node. No exemplo abaixo estamos passando a variável MaxClient com valor 709, e variável user com valor torvalds:

```
node02.dexter.com.br    maxClient=709    user=torvalds
```

## Inventário do Ansible

Máquina: Devops

1

No arquivo de configuração, vamos habilitar o diretório de “roles”, definir timeout de conexão, habilitar o arquivo de log e a chave privada que criamos.

```
1# vim /etc/ansible/ansible.cfg  
...  
roles_path = /etc/ansible/roles  
timeout = 30  
log_path = /var/log/ansible.log  
private_key_file = /etc/keys/key.pem  
...
```

51



Parâmetros do arquivo de configuração do ansible:

### Parâmetro roles\_path:

Caminhos adicionais para a localização das roles do ansible. Podemos passar mais de um caminho separados por vírgula (,).

### Parâmetro timeout:

Tempo de esperar para conectar ao servidor em segundos. Por padrão, o valor 10 segundos aumentará para 30 segundos.

### Parâmetro log\_path:

Por padrão, os logs do ansible são desativados. Para habilitar o log precisamos especificar o arquivo pela variável **log\_path**. Por padrão, o arquivo **/var/log/ansible.log**.

### Parâmetro private\_key\_file:

Quando definido, sempre irá utilizar a chave privada do SSH para autenticar nos nodes. Podemos também passar por parâmetro outra chave em alguns casos, utilizando o **-private\_key**.

## Inventário do Ansible

Máquina: Devops

2

Dentro do arquivo de inventário, passaremos os servidores “devops” e “docker”, que serão acessados pelo ansible no laboratório

```
1# vim /etc/ansible/hosts

[devops]

devops.dexter.com.br

[automation]
automation.dexter.com.br

[docker]

docker.dexter.com.br
```

52



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Inventário do Ansible

Máquina: Devops

3

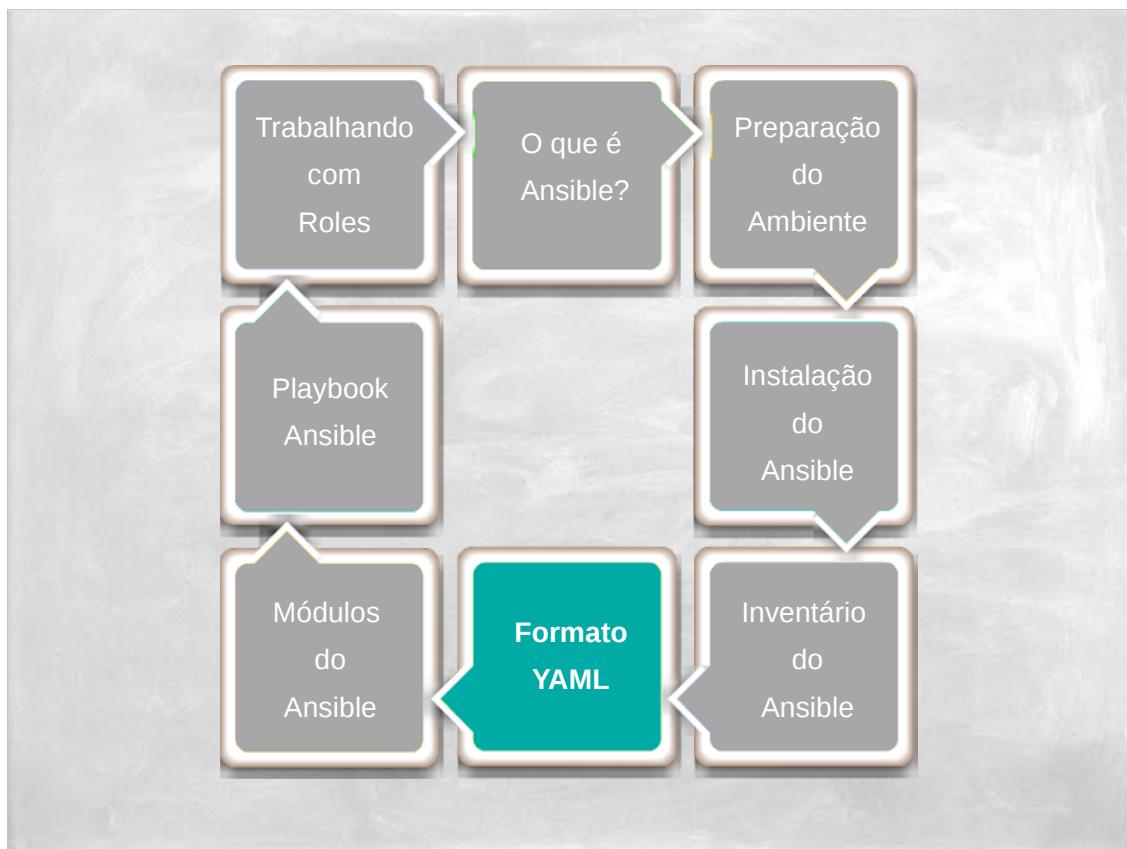
Para testar a conexão entre o ansible e os hosts, vamos executar um ls em todos eles.

```
# ansible all -m command -a "ls /"
```

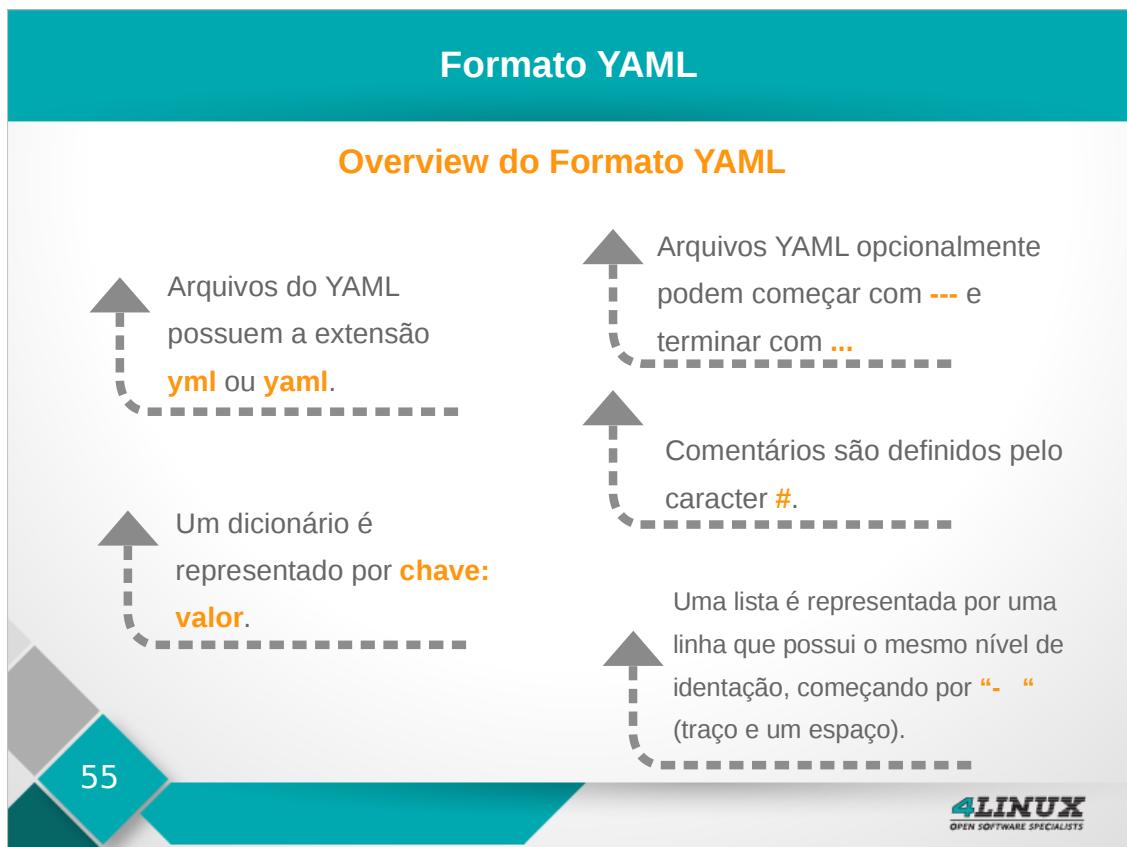
53



Utilizamos o modulo “command” para executar comandos externos dentro ou fora das playbooks, caso haja necessidade de obter informações sobre hosts remotos existe o comando ansible all -m setup, que fornece muitos fatos sobre o sistema automaticamente.



## Anotações:



YAML é uma linguagem simples de ler e escrever, o ansible utiliza o formato no arquivos de configuração dos Playbooks e roles.

Vamos ver o exemplo de dados representados no formato YAML

```
---
# Funcionário da Dexter

- name: João Devops da Silva
  job: Analista DevOps
  Skills:
    - Administração de Sistema GNU/Linux
    - Automação com Ansible
    - Gerencia de configuração com Puppet
    - Integração e CI com Jenkins
    - Versionamento com Git e Gitlab
```

## Formato YAML



**Link:** <http://www.yamllint.com/>

**Link:** <http://yaml-online-parser.appspot.com/>



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



## Anotações:

## Módulos do Ansible

**Módulos**, também conhecidos por “plugins de tarefas”, é o que de fato é executado em cada tarefa do nosso playbook.

Veja ao lado alguns módulos core do Ansible:

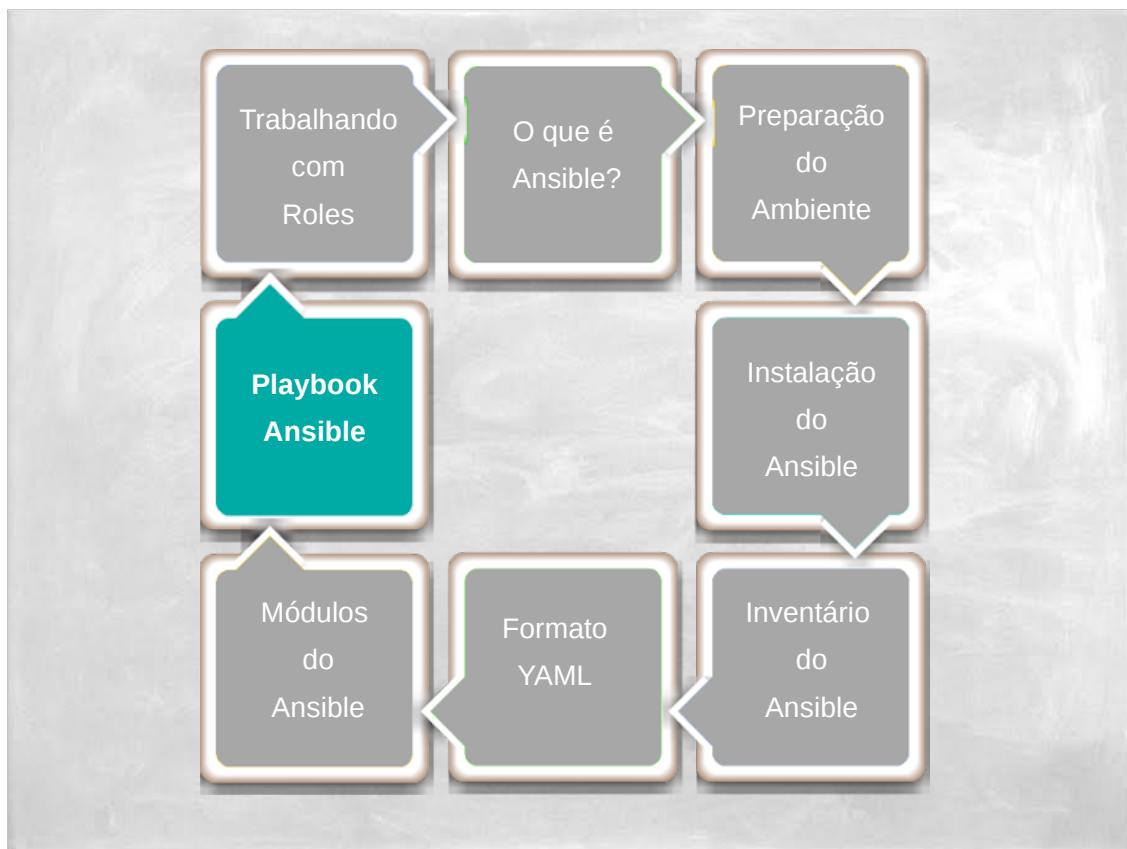
- apt
- shell
- copy
- template
- unarchive
- yum
- command
- service
- git
- mount

Os módulos do ansible são onde a mágica acontece, eles são responsável por executar as tarefas designadas a eles nos nodes gerenciados pelo ansible. Por padrão o módulos estão localizados no diretório `/urs/share/ansible`.

Podemos desenvolver os nossos próprios módulos em qualquer linguagem de programação, sendo que o ansible utiliza o JSON para se comunicar com módulos.

Lista de Módulos do Ansible:

- **apt**: gerencia instalação de pacotes utilizando o APTITUDE (debian/ubuntu);
- **yum**: gerencia instalação de pacotes utilizando o YUM (RedHat/CentOS/Fedora);
- **package**: gerencia instalação de pacotes utilizando o APTITUDE/YUM (o pacote deve possuir o mesmo nome em ambas as distribuições);
- **command**: Executa um comando em node remoto;
- **shell**: Executa um shell script dentro da máquina, após realizar as transferências;
- **service**: Gerencia serviços em máquina remotas;
- **copy**: copia arquivo na máquina local para o node remoto;
- **git**: gerencia repositório do git;
- **unarchive**: Descompactar arquivos na máquina remota;
- **mount**: Gerencia os dispositivos montados na máquina;
- **template**: Gerencia templates no ansible;
- **ec2**: Gerencia instancias ec2 ambiente cloud da Amazon;
- **docker\_container**: Gerencia containers de docker.



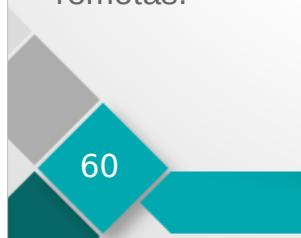
## Anotações:

## Playbooks

**Playbooks** são arquivos do Ansible em que você descreve as tarefas a serem executadas nos servidores.



Em um nível mais básico, os Playbooks podem ser usados para gerenciar as configurações e realizar deploys nas máquinas remotas.



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Meu Primeiro Playbook

Máquina: Devops

1

Vamos criar um diretório para testarmos os playbooks ansible.

```
1# mkdir -p /root/exemplos/ansible
```

2

Entre no diretório criado.

```
2# cd /root/exemplos/ansible
```

61



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Meu Primeiro Playbook

Máquina: Devops

3

Nesse exemplo, vamos manter o arquivo **resolv.conf** em todas as máquinas da Dexter via ansible.

```
1# vim resolv.conf  
# Atualizado via Ansible  
nameserver 8.8.8.8  
nameserver 8.8.4.4  
search dexter.com.br  
domain dexter.com.br
```

62



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Meu Primeiro Playbook

Máquina: Devops

4

No nosso primeiro playbook, vamos automatizar a instalação do pacote “sl” e o arquivo “resolv.conf” criado anteriormente. Daremos o nome “playbook.yml”.

```
1# vim playbook.yml
#
# Playbook de exemplo:
-
  name: Meu Primeiro playbook
  hosts: all
  tasks:
    -
      name: Atualizar o resolv.conf
      copy: src=resolv.conf dest=/etc/resolv.conf
    -
      name: Garantir a instalação do pacote sl em todas as máquinas
      hosts: devops docker
      tasks:
        -
          name: Instalar o sl
          apt: name=sl update_cache=yes state=present
```

63



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Meu Primeiro Playbook

Máquina: Devops

- 5 Para executar o playbook criado, utilizaremos o comando “ansible-playbook”.

```
1# ansible-playbook playbook.yml
```

- 6 Utilizaremos o módulo “command” para validarmos se o arquivo “/etc/resolv.conf” está presente em todos os hosts.

```
2# ansible all -m command -a "cat /etc/resolv.conf"
```



**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Usando Variáveis em Playbooks

Em alguns casos, precisamos que nosso **playbook** possua variáveis.



**Variáveis** podem ser passadas por inventário. Podemos definir as variáveis dentro do playbook ou passar as mesmas pelo próprio comando de execução dos playbooks.



**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Usando Variáveis em Playbooks

Máquina: Devops

1

Vamos construir um novo playbook com o nome “**users.yml**” para criar um usuário em todos os servidores da dexter.

```
1# vim users.yml
---
# Playbook para criar usuario
- name: Criacao de usuarios nos servidores
  hosts: all
  vars:
    - username: sysadmin

  tasks:
    - name: Criando os usuarios nos servidores
      user: name={{ username }} shell=/bin/bash createhome=yes
```

66



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Usando Variáveis em Playbooks

Máquina: Devops

- 2 Vamos utilizar o comando “ansible-playbook” para executar o playbook.

```
1# ansible-playbook users.yml
```

- 3 Para validar se o usuário foi criado com sucesso, vamos utilizar o comando getent.

```
2# getent passwd sysadmin
```

- 4 Executaremos novamente o playbook, alterando o valor da variável.

```
3# ansible-playbook -e username=developer users.yml
```

- 5 Verifique se o usuário foi criado com sucesso.

```
4# getent passwd developer
```

67



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Usando Loops em Playbooks

Em alguns casos precisamos executar várias tasks do mesmo módulo inúmeras vezes como, por exemplo, ao instalar vários pacotes em um servidor.



Para isso podemos usar o parâmetro `with_items` do Ansible para realizar um loop e executar a mesma task mais de uma vez.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Usando Loops em Playbooks

Máquina: Devops

1

Vamos modificar o arquivo **playbook.yml** colocando um loop para instalar os pacotes básicos.

```
1# vim playbook.yml
...
- name: Garatir pacotes basicos do sistema
  apt: name={{ item }} update_cache=yes state=present
  with_items:
    - vim
    - nano
    - sl
    - ccze
    - dnsutils
...
...
```

69



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Tasks Condicionais

Em alguns casos precisamos que uma tarefa do Ansible seja executada de acordo com uma condição.



Para isso, podemos usar o parâmetro **when** do Ansible para definir quando ele deve executar um módulo de acordo com alguma especificação.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Tasks Condicionais

Máquina: Devops

```
1# vim when.yml
---
# Playbook para teste condicional
- name: Exemplo de operador condicional
  hosts: all

  tasks:
    - name: Testando arquivo do profile
      command: test -f /etc/profile.d/dexter
      register: result
      ignore_errors: yes
    - name: Configurando a variavel da Dexter
      shell: echo "export DEXTER=/opt/dexter" > /etc/profile.d/dexter
      when: result.rc == 0
```

1

Vamos criar o playbook **when.yml** para realizar o teste do operador condicional.

71



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Tasks Condicionais

Máquina: Devops

2

Vamos executar o playbook com o comando “ansible-playbook”.

```
1# ansible-playbook when.yml
```

3

Verifique se o arquivo foi criado.

```
2# ansible all -a "ls /etc/profile.d"
```

4

Crie o arquivo de configuração da Dexter.

```
3# ansible all -a "touch /etc/profile.d/dexter"
```

72



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Tasks Condicionais

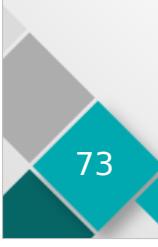
Máquina: Devops

- 5 Execute o playbook novamente.

```
1# ansible-playbook when.yml
```

- 6 Verifique se o arquivo foi criado com sucesso nos servidores.

```
2# ansible all -a "cat /etc/profile.d/dexter"
```



73



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

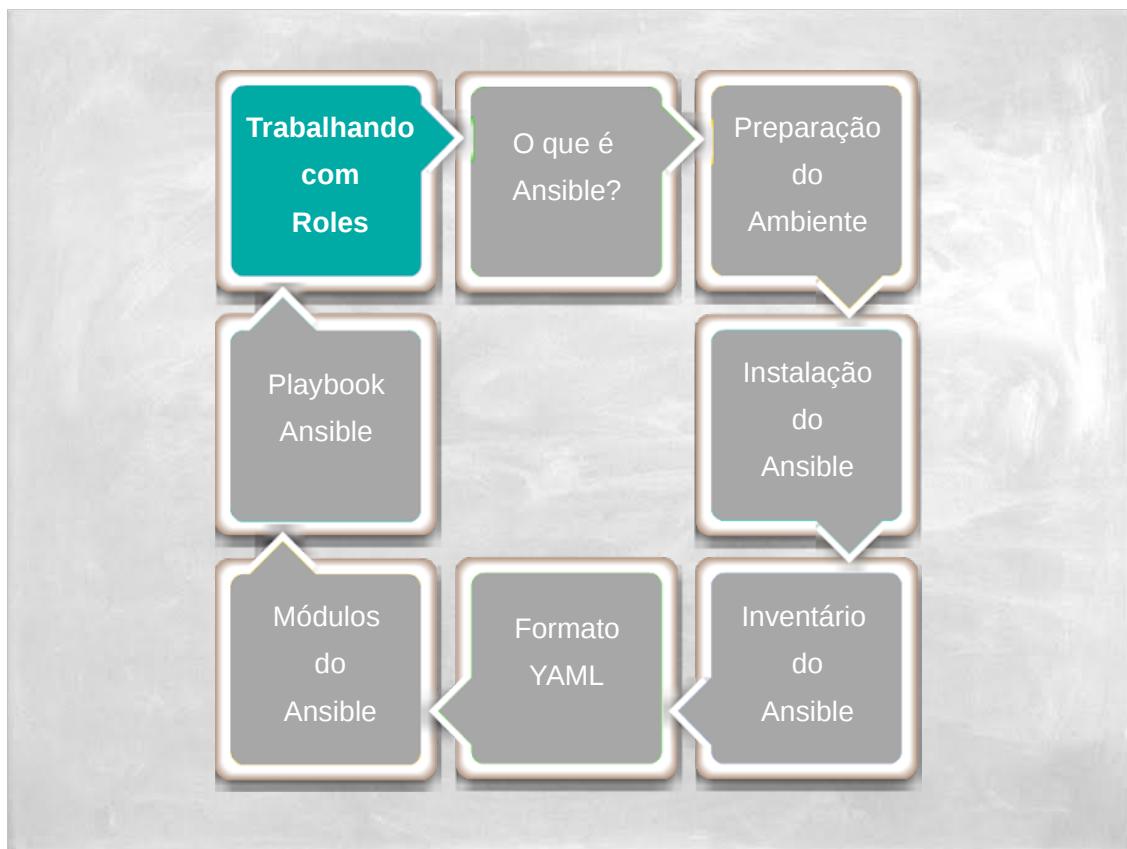
---

---

---

---

---

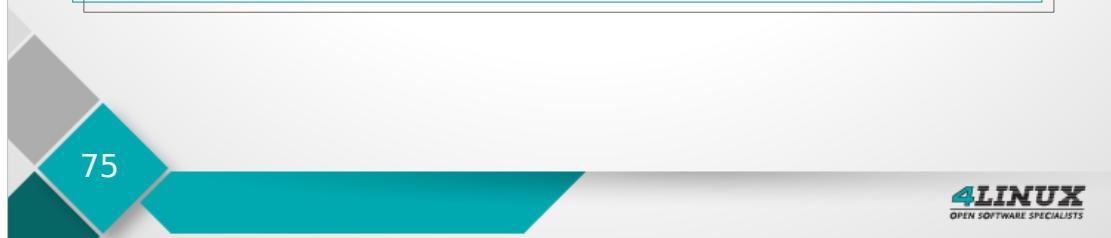


## Anotações:

## Trabalhando com Roles

A melhor maneira de organizar os playbooks do Ansible é através das **Roles**.

**Roles** é uma maneira de carregar automaticamente arquivos de acordo com uma estrutura de diretórios definidas.



### Exemplo de uma estrutura de diretórios e roles

```
/etc/ansible/roles/  
  puppet/  
    files/  
    templates/  
    tasks/  
    handlers/  
    vars/  
    defaults/  
  Meta
```

### Comportamento de carregamento automático das roles:

- Caso exista o arquivo `/roles/<role>/tasks/main.yml`, as tasks definidas dentro do arquivo serão inclusa no Playbook;
- Caso exista o arquivo `/roles/<role>/handlers/main.yml`, os handlers definidos dentro do arquivo será incluso no playbook;
- Caso exista o arquivo `/roles/<role>/vars/main.yml`, as variáveis definida dentro do arquivo será incluso no playbook;
- Qualquer tarefa de `copy`, `script`, `template` ou `include` pode utilizar arquivos dentro dos diretórios `files`, `templates` e `tasks`.

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

Entendendo a estrutura de diretório das roles:

- Default - Diretório onde é possível informar o valor de alguma variável do seu playbook, alterando o valor do “vars”;
- Files - Diretório onde é armazenado os arquivos com conteúdo fixo, que você deseja utilizar no playbook;
- Handlers - Comandos que você deseja que sejam executados, caso alguma atividade do seu playbook tenha sido concluído sucesso;
- Tasks - Todas as tarefas do seu playbook;
- Templates - Diretório onde é armazenado os arquivos com conteúdo variável, e que você deseja colocar no servidor;
- Vars - Diretório onde é informado o valor fixo de variáveis que podem ser utilizados em um playbook.

Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

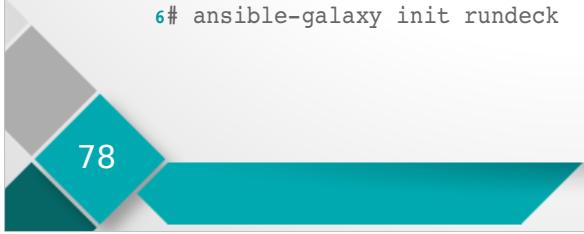
## Trabalhando com Roles

Máquina: Devops

Vamos criar uma role para cada ferramenta que vamos utilizar no laboratório, assim automatizaremos a instalação de todas elas com ansible.

- Crie o diretório e a estrutura básica das roles.

```
1# cd /etc/ansible/roles  
2# ansible-galaxy init docker  
3# ansible-galaxy init puppet  
4# ansible-galaxy init gitlab  
5# ansible-galaxy init jenkins  
6# ansible-galaxy init rundeck
```



78



O ansible galaxy é uma ferramenta do ansible utilizada para baixar e criar roles. No nosso exemplo, vamos apenas criar a estrutura de roles que será utilizada no laboratório da Dexter.

## Trabalhando com Roles

Máquina: Devops

2

Dentro do diretório de cada role, editaremos o arquivo das tasks e colocaremos as instruções de instalação de cada aplicação (Docker, Puppet, Gitlab, Jenkins e Rundeck):

```
1# vim docker/tasks/main.yml  
2# vim puppet/tasks/main.yml  
3# vim gitlab/tasks/main.yml  
4# vim jenkins/tasks/main.yml  
5# vim rundeck/tasks/main.yml
```



79

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

```
1# vim docker/tasks/main.yml
```

```
---
```

```
# tasks file for docker
```

```
- name: Instalação das dependências do Docker-Engine
```

```
  apt: name={{ item }} update_cache=yes state=present
```

```
  with_items:
```

```
    - apt-transport-https
```

```
    - ca-certificates
```

```
    - curl
```

```
    - software-properties-common
```

```
- name: Incluir o repositorio do Docker-Engine
```

```
  apt_repository:
```

```
    repo: "deb [arch=amd64] https://download.docker.com/linux/ubuntu
```

```
    {{ ansible_distribution_release }} stable"
```

```
    state: present
```

```
- name: Inclusão da chave do repositorio do Docker-Engine
```

```
  apt_key: url="https://download.docker.com/linux/ubuntu/gpg" state=present
```

```
- name: Instalar o Docker-Engine
```

```
  apt: name=docker-ce update_cache=yes state=present
```

```
2# vim puppet/tasks/main.yml
---

# tasks file for puppet
- name: Baixar o pacote do repositório do Puppet
  get_url:
    url: "https://apt.puppetlabs.com/puppetlabs-release-pc1-{{ ansible_distribution_release }}.deb"
    dest: /tmp/puppet.deb

- name: Instalar o repositório do Puppet
  apt:
    deb: /tmp/puppet.deb
    state: present

- name: Instalacao dos pacotes e dependencias do puppet
  apt: name={{ item }} update_cache=yes state=present
  with_items:
    - openssl
    - ntp
    - puppet-lint
    - puppet
    - puppet-common
=====

3# vim gitlab/tasks/main.yml
---

# tasks file for gitlab
- name: Instalacao das dependencias do GitLab
  apt: name={{ item }} update_cache=yes state=present
  with_items:
    - curl
    - openssh-server
    - postfix
    - ca-certificates

- name: Baixar o pacote com o repositorio get_url
  get_url:
    url: "http://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.deb.sh"
    dest: /tmp/script.deb.sh
    owner: root
    group: root
```

```
- name: Instalação do repositório do GitLab  
  command: /tmp/script.deb.sh
```

```
- name: Instalacao do GitLab  
  apt:  
    name: gitlab-ce  
    update_cache: yes  
    state: present
```

```
- name: Configurando GitLab para primeiro uso  
  command: gitlab-ctl reconfigure
```

---

```
4# vim jenkins/tasks/main.yml
```

```
---
```

```
# tasks file for jenkins  
- name: Importando chave do repositório do Jenkins  
  rpm_key:  
    key: https://jenkins-ci.org/redhat/jenkins-ci.org.key  
    state: present  
    validate_certs: no
```

```
- name: Adicionando repositório do Jenkins  
  get_url:  
    url: http://pkg.jenkins-ci.org/redhat/jenkins.repo  
    dest: /etc/yum.repos.d/jenkins.repo
```

```
- name: Install do jenkins  
  yum: name=jenkins state=present
```

---

```
5# vim rundeck/tasks/main.yml
```

```
---
```

```
# tasks file for rundeck  
- name: Download do repositorio
```

```
  yum: name=http://repo.rundeck.org/latest.rpm state=present
```

```
- name: Instalacao do rundeck  
  yum: name={{ item }} state=present  
  with_items:
```

## Trabalhando com Roles

Máquina: Devops

3

Para executar cada role, precisamos de um playbook utilizando o módulo "roles". Vamos criar os playbooks dentro do diretório /etc/ansible/playbooks/ambiente.

```
1# mkdir /etc/ansible/playbooks/ambiente  
2# vim /etc/ansible/playbooks/ambiente/docker.yml  
3# vim /etc/ansible/playbooks/ambiente/devops.yml  
4# vim /etc/ansible/playbooks/ambiente/automation.yml
```



82



```
1# vim automation.yml
```

---

```
- name: executando roles no host automation  
hosts: automation  
roles:  
- jenkins  
- rundeck
```

```
2# vim docker.yml
```

---

```
- name: executando roles no host docker  
hosts: docker  
roles:  
- docker
```

```
3# vim devops.yml
```

---

```
- name: criando a infraestrutura devops  
hosts: devops  
roles:  
- puppet  
- gitlab
```

## Documentação

Para mais informações sobre o Ansible e seus módulos, você pode acessar a documentação pelo site oficial:

**Link:** <http://docs.ansible.com/>



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Gerenciando e Operando o Docker

### Objetivos da Aula

- Fazer uma introdução ao Docker;
- Instalar o Docker;
- Criar o primeiro Container;
- Gerenciar os containers;
- Copiar arquivos para os containers;
- Gerenciar imagens com dockerfile;
- Gerenciar volumes;
- Realizar Network com Docker.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

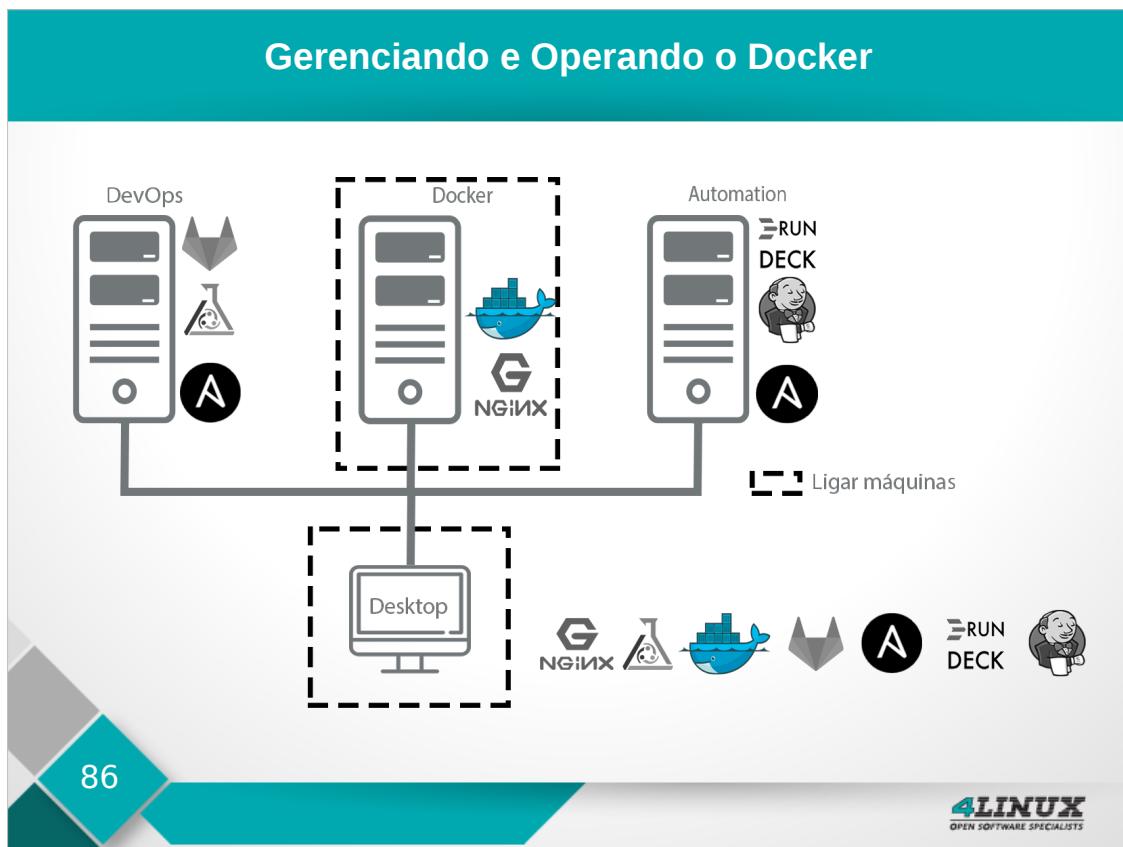
---

---

---

---

---



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

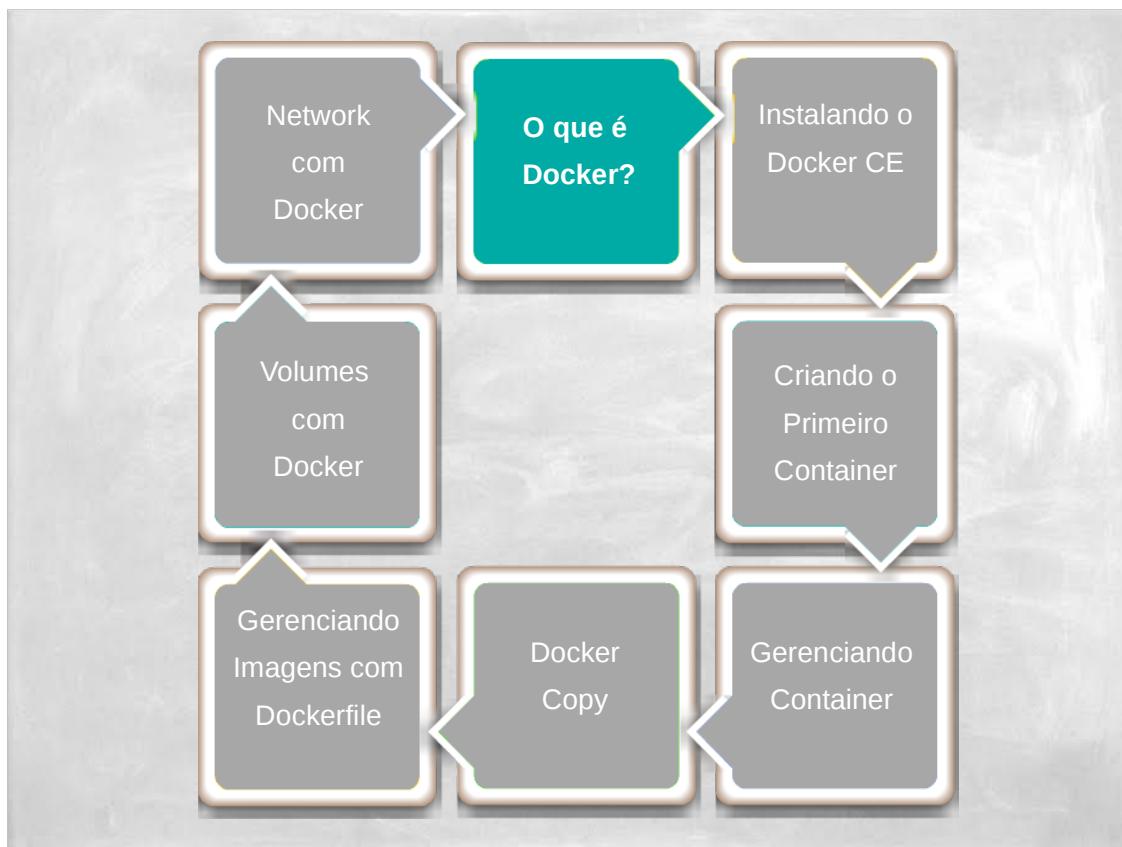
---

---

---

---

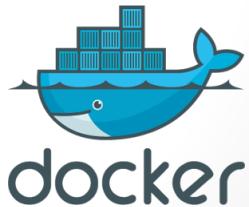
---



Não dá mais para nós SysAdmin administrarmos nossa Infra daquela maneira tradicional que fazíamos: instalação manual, alguns scripts em shell para agilizar tarefas rotineiras, contratar um estagiário para fazer as rotinas redundantes como trocar a senha dos servidores, instalar um agente de Zabbix ou Bacula, etc.

## Introdução ao Docker

**Docker** é uma ferramenta para gerenciar containers. Os containers do Docker 'enjaulam' o software em um filesystem completo que contém tudo o que precisa para rodar códigos, aplicações, ferramentas de sistema e bibliotecas, basicamente, tudo o que pode ser instalado em um servidor.



O Docker utiliza Linux Containers (LXC). Containers é um método de virtualização rodando um sistema operacional isolado (containers) sobre um máquina hospedeira (Máquina Host). Diferente de uma virtualização, este método se utiliza do mesmo kernel do sistema operacional Host. Basicamente, container é um processo isolado através de namespaces e um chroot (**Change Root**).

Executar o chroot é tão simples e rápido quanto iniciar um novo processo no sistema operacional. Porém, o chroot de um sistema operacional requer muita I/O de disco, podendo assim tornar o processo lento. Para evitar esse tipo de problema, o Docker utiliza conceitos de imagens que contém a estrutura básica do sistema operacional e, para cada container, ele utiliza um **diff** de um container base.

O Docker utiliza o AuFS para o sistema de arquivos. É um sistema de arquivo em camadas, dando a possibilidade de ter uma parte destinada para leitura e outra parte para escrita, além do compartilhamento de diretórios entre os containers.

# Introdução ao Docker

## Vantagens na Utilização de Containers

### Leveza

O Container executa sobre uma máquina que compartilha o mesmo Kernel do Sistema Operacional. Assim, este é iniciado instantaneamente e utiliza a memória RAM de uma forma mais eficiente.

89

### Segurança

O Container isola as aplicações uma das outras, adicionando uma camada de segurança para a aplicação.



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

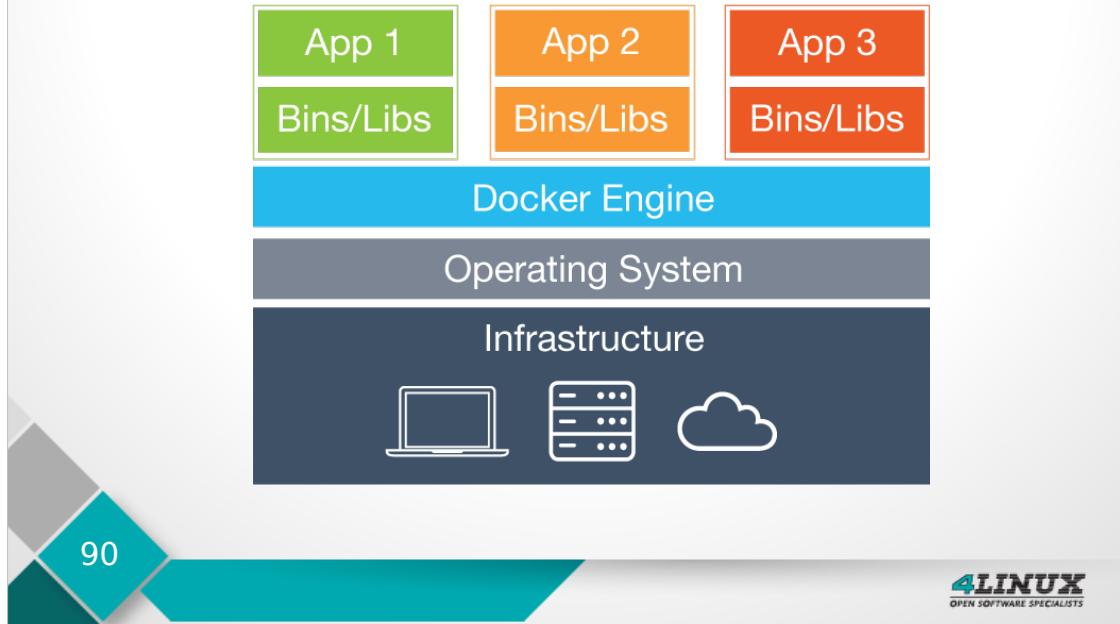
---

---

---

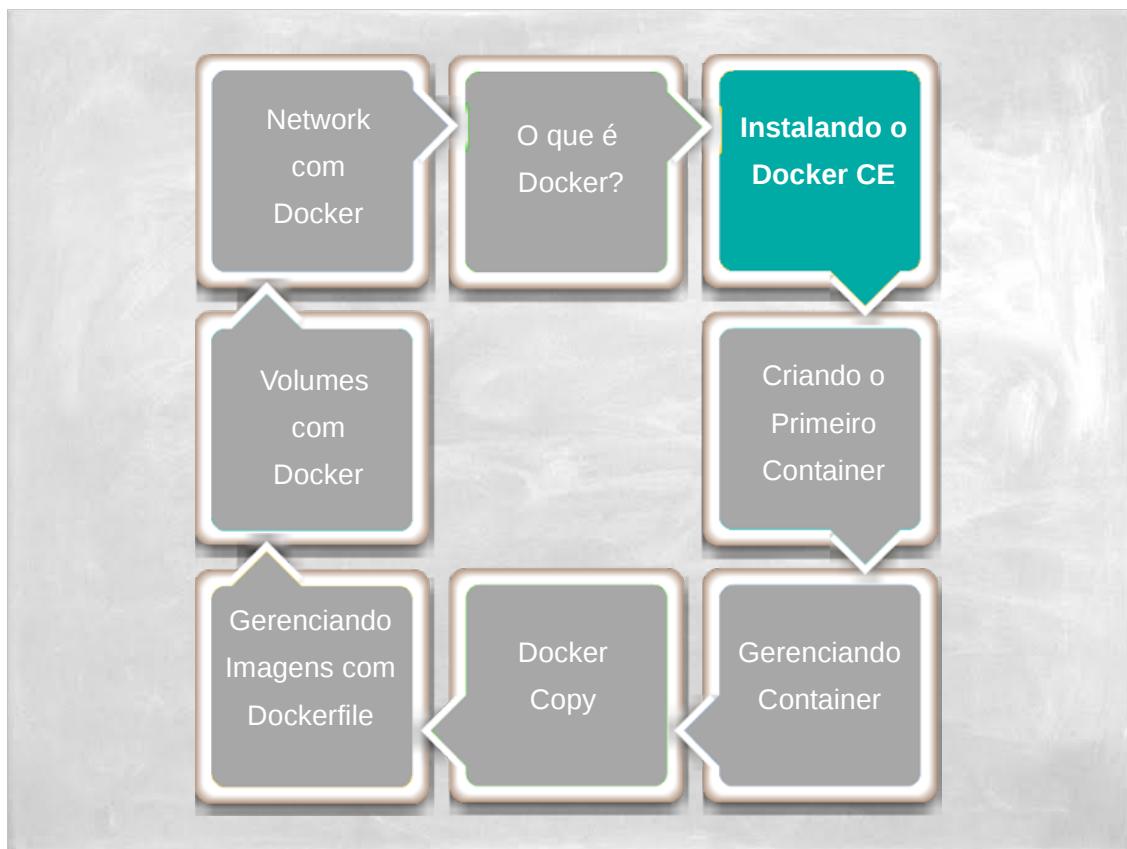
# Introdução ao Docker

## Arquitetura do Docker Engine



O Docker Engine é responsável por criar e executar os containers do Docker, ele atua entre a máquina Host e os containers.

A Daemon é comunicada pelo cliente através da própria API para gerenciar os containers. Esse cliente pode ser instalado na mesma máquina, que é o padrão utilizado nas distribuições (comando docker). O Docker Engine também pode ser acessado externamente através da rede, normalmente utilizado em ambientes de cluster ou interfaces de gerenciamento.



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Instalação do Docker CE

Máquina: Docker

Agora, vamos executar o playbook do ansible para instalação do docker CE.

1

Abra o arquivo /etc/ansible/roles/docker/tasks/main.yml para que possamos conferir os passos de instalação do Docker CE.

```
1# vim /etc/ansible/roles/docker/tasks/main.yml
```

2

Para executar o playbook, podemos usar o comando:

```
1# ansible-playbook /etc/ansible/playbooks/ambiente/docker.yml
```



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

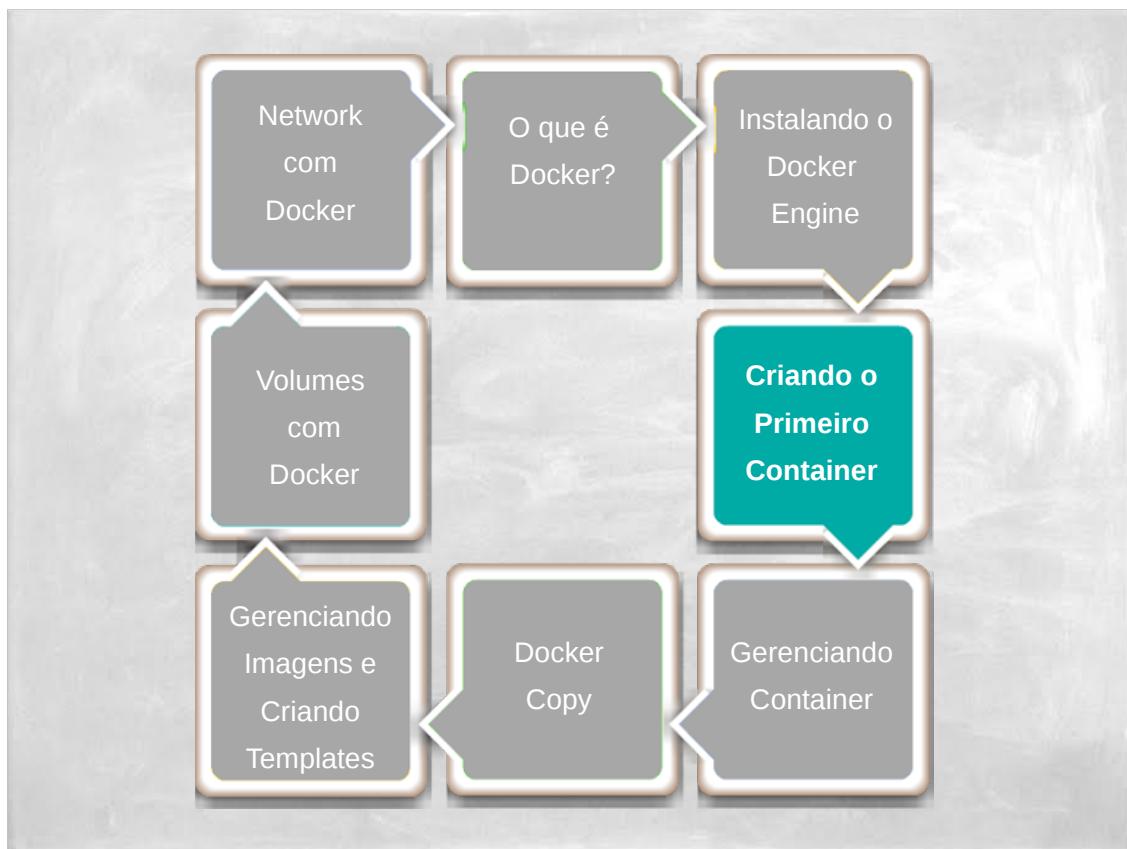
---

---

---

---

---



## Anotações:

# Criando o Primeiro Container

Máquina: Docker

Agora começaremos a criar e usar containers gerenciados pelo Docker.

- 1 Crie um container com o nome “**Primeiro**”.

```
1# docker run --name primeiro debian /bin/ls /etc
```

- 2 Liste todos os containers criados pelo Docker, inclusive os conteiners parados.

```
2# docker ps -a
```

- 3 Inicie o container novamente.

```
3# docker start primeiro
```

94



## Docker run:

O Comando **run** é utilizado para rodar um comando em um novo container.

Syntaxe do comando run:

```
docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
```

Vamos analisar cada parâmetro do primeiro comando abaixo:

```
$ docker run --name hello debian echo "Hello Container"
```

A Opção **-name** é utilizada para definir o nome do container. No caso **hello**, todo container precisa de uma imagem como base para os arquivos do sistema. Essa imagem contém os binários, bibliotecas e diretórios básicos do sistema operacional. No exemplo acima, estamos utilizando a imagem do **debian**. Por último, temos o comando a ser executado dentro do container. **echo “Hello Container”**.

### Docker PS:

O comando **ps** é o comando utilizado para listar os containers. Ao executar o comando sem nenhum parâmetro, o mesmo retorna somente os containers, que estão rodando no sistema. Para ver todos os containers, sejam eles parados ou ativos, utilizamos o parâmetro **-a**.

### Docker start:

O comando **start** é usado para iniciar os containers do docker. Podemos iniciar mais de um container no mesmo comando, como por exemplo:

```
$ docker start container1 container2 container3 ...
```

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

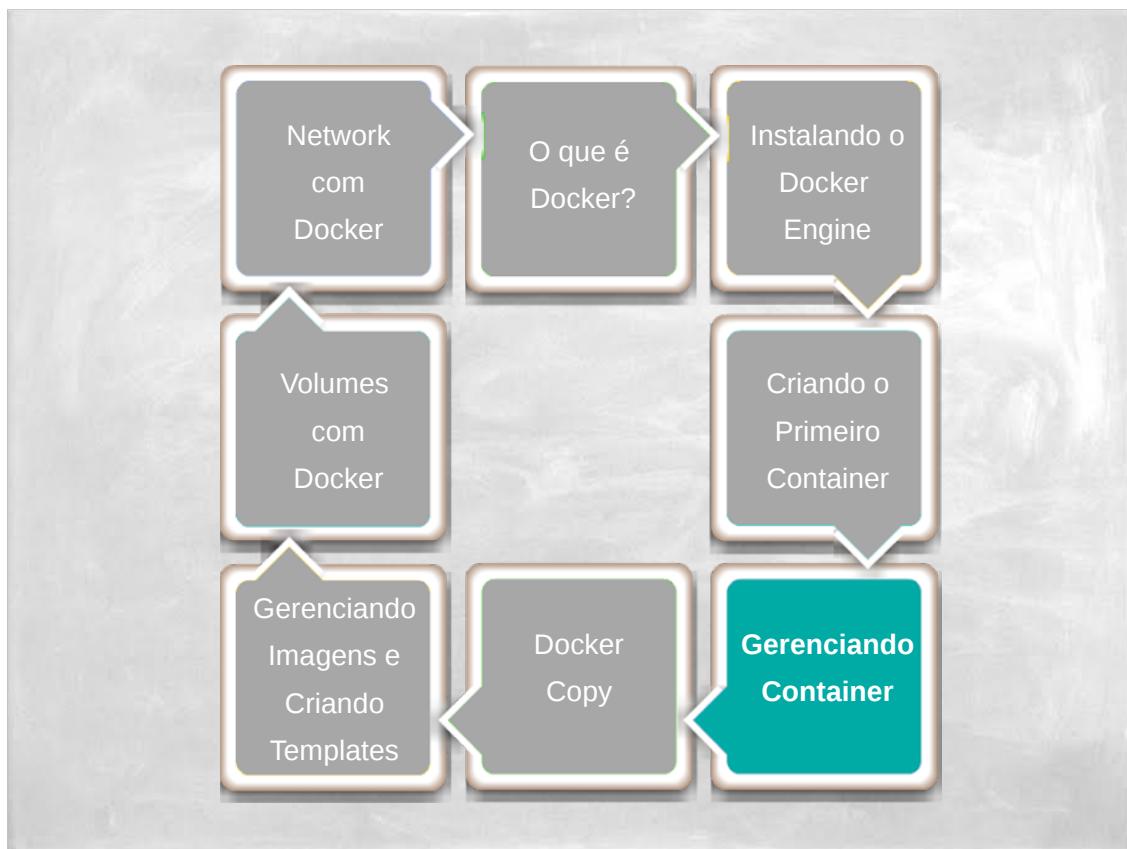
---

---

---

---

---



## Anotações:

# Gerenciando Containers

Máquina: Docker

- 1 Remova o container criado anteriormente.

```
1# docker rm primeiro
```

- 2 Crie o container novamente, porém agora passando a bash como comando.

```
2# docker run -it --name primeiro debian /bin/bash
```

- 3 Desconecte o terminal do container.

```
3# CTRL^P e CTRL^Q
```

- 4 Pare o container criado.

```
4# docker stop primeiro
```

97



## Docker rm:

O comando **rm** é utilizado para remover os containers, assim como com o comando **start** podemos remover mais de um container executando um mesmo comando, como por exemplo:

```
$ docker rm container1 container2 container3
```

## Docker inspect:

O comando **inspect** é utilizado para retornar informações sobre o container. Através dele é possível visualizar todas as informações disponíveis sobre um container criado pelo docker. Por padrão, o comando retorna as informações sobre o formato JSON. Através desse comando podemos visualizar o ID do Container, Data de Criação, comando executado, qual o status do container no momento da execução do comando, o ID da imagem do container, informações sobre a rede do container, utilização de recursos e muitas outras informações disponibilizadas pelo Docker Engine. Exemplo do comando **inspect**:

```
$ docker inspect container1
```

# Gerenciando Containers

Máquina: Docker

- 5 Inicie o container criado anteriormente.

```
1# docker start primeiro
```

- 6 Conecte o terminal ao container para interagir com o interpretador de comandos.

```
2# docker attach primeiro
```

- 7 Desconecte o terminal do container.

```
3# CTRL^P e CTRL^Q
```

**Atenção:** Caso você digite 'exit' para sair do container, o mesmo será parado.



98

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

## Docker attach:

O comando **attach** é utilizado para se conectar ao container, usando o nome ou ID do container. Ao se conectar ao container, você pode visualizar a saída do comando. Caso esteja habilitado com a opção **-i**, é possível interagir com o container.

Para desconectar do container sem parar sua execução, precisamos utilizar a sequência de teclas **CTRL^P + CTRL^Q**. Se utilizamos os comandos padrões do bash para sair do interpretador (**exit** ou **CTRL^D**), iremos parar a execução do container.

Mas por que eu não posso sair simplesmente executando o **exit**?

Os containers do Docker permanecem executando enquanto o comando passado na criação do mesmo esteja rodando. Como vimos no nosso primeiro exemplo, quando utilizamos o comando `echo "Hello Container"`, o container foi criado e após o término da execução do comando `echo`, o container é parado. Este é o comportamento de um container no Docker. Ele continua vivo enquanto o comando que foi designado a ele estiver sendo executado. Sendo assim, quando utilizamos o `exit` do **bash**, estamos na verdade parando o interpretador, por consequência, parando o container. Podemos utilizar o comando **start** para iniciar o container novamente.

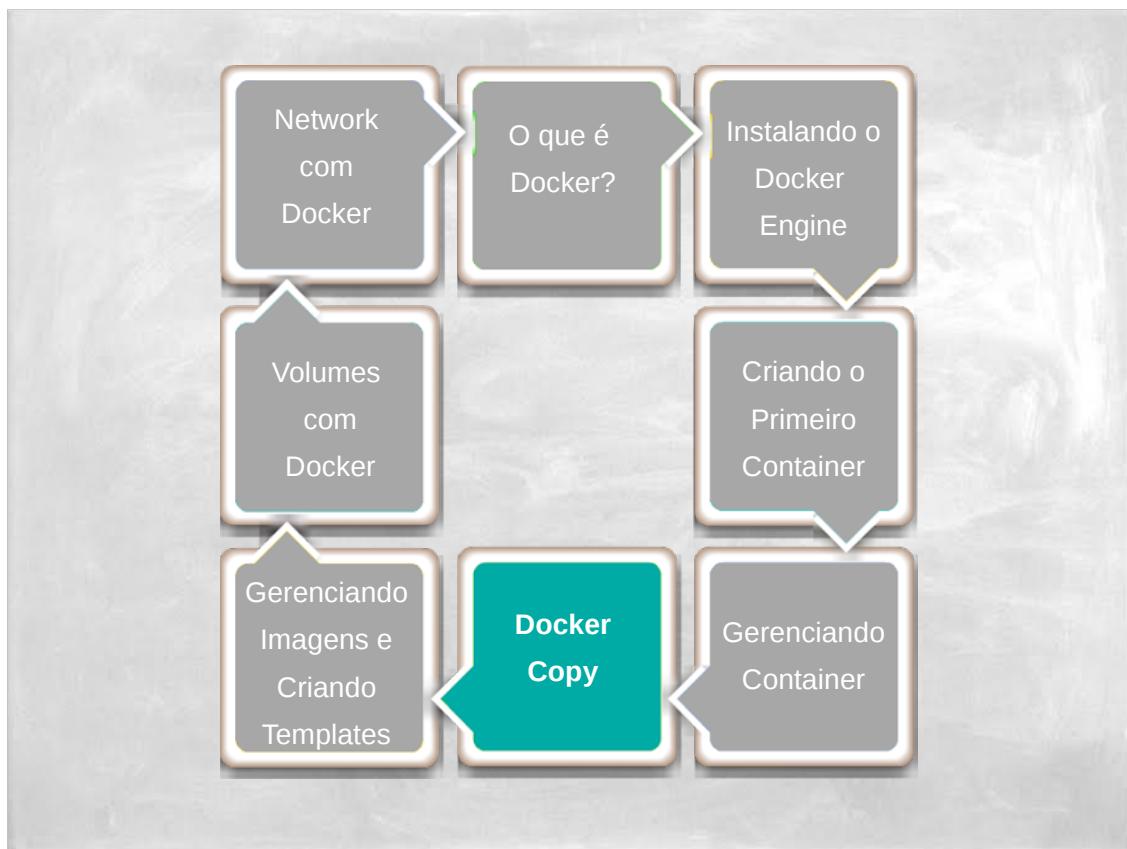
No exemplo do slide, temos dois parâmetros novos passados ao criar o container:

```
$ docker run -dit --name hello primeiro /bin/bash
```

O parâmetro **-i --interactive** na sua forma estendida, é utilizado para deixar a entrada de dados ao container aberta, caso você não esteja conectado (attached) no container em questão.

O parâmetro **-t** é utilizado para alocar um pseudo-tty ao container. Assim, podemos interagir com o interpretador de comando passando comandos para o **/bin/bash**, da mesma forma que usamos um terminal e uma máquina virtual ou máquina física.

O parâmetro **-d** (detached) é para ser utilizado caso você não queira se conectar no container no momento de sua criação.



## Anotações:

# Copiando Arquivos para Containers

Máquina: Docker

- Crie o arquivo de teste na máquina Host.

```
1# echo "Arquivo de Teste" > /tmp/arquivo
```

- Copie o arquivo para dentro do container.

```
2# docker cp /tmp/arquivo primeiro:/tmp
```

- Liste o diretório /tmp dentro do container, utilizando o comando **exec** do Docker.

```
3# docker exec primeiro ls /tmp
```

- Copie o arquivo de dentro do container para a máquina hospedeira com outro nome.

```
4# docker cp primeiro:/tmp/arquivo /tmp/arquivo.backup
```

101

**Atenção:** Não é possível realizar a cópia entre os containers.



## Comando cp

Em alguns casos, precisamos de um arquivo dentro do container, porém, o container não tem acesso ou esse arquivo é criado pela máquina host. Então, podemos utilizar o comando **cp** do docker para copiar arquivos do container para o host e vice-versa.

### Syntaxe do comando:

```
$ docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH
$ docker cp [OPTIONS] SRC_PATH- CONTAINER:DEST_PATH
```

No exemplo acima, foi criado um arquivo para realizar o teste de cópias de arquivos. No segundo exemplo, estamos copiando o arquivo da máquina host para o container. Perceba que queremos referenciar o container sempre que utilizamos o **nome\_do\_container:caminho\_arquivo**. Já no quarto exemplo, estamos copiando o arquivo de dentro do container para a máquina host novamente, porém, com outro nome.

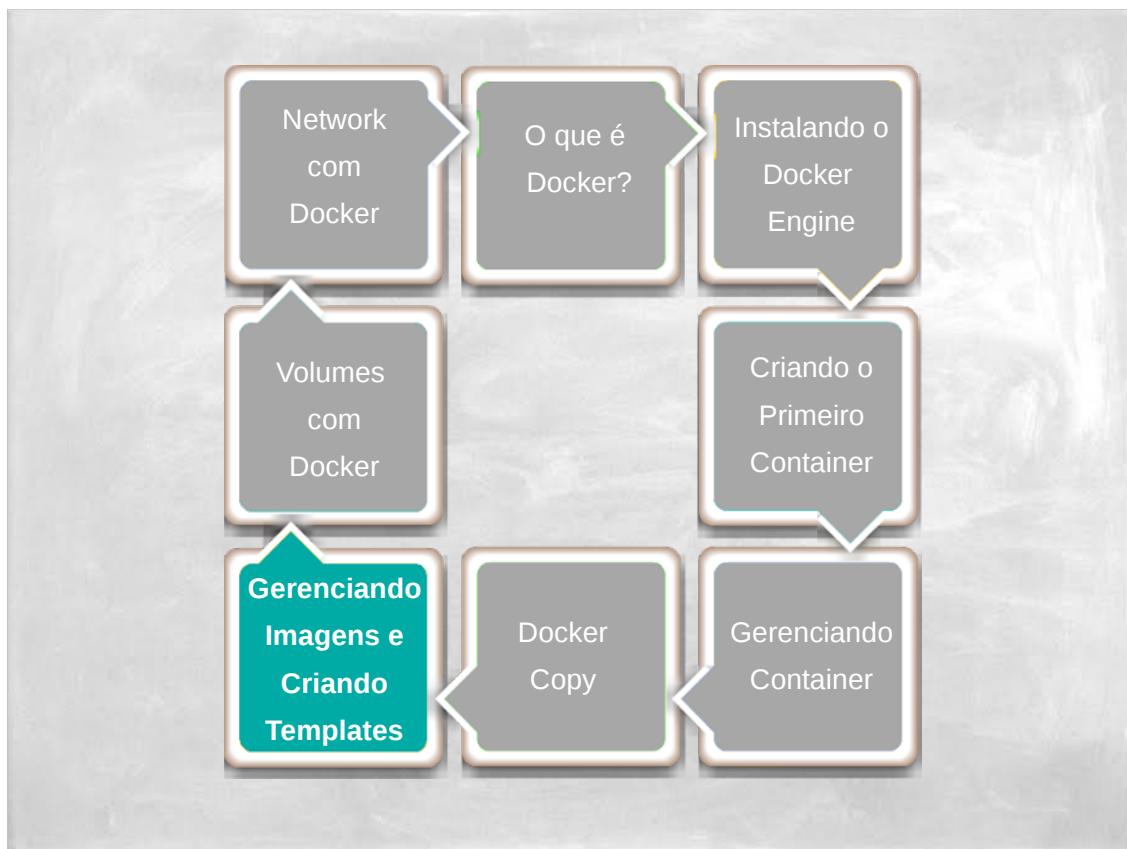
## Docker exec

O comando **exec** é utilizado para executar um comando dentro do container. O comando só é executado dentro do Docker caso o processo primário do Docker esteja rodando, ou seja não é possível executar comandos enquanto o container estiver parado.

Syntaxe do comando exec:

```
docker exec [OPTIONS] CONTAINER COMMAND [ARG...]
```

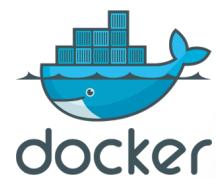
No exemplo numero três, estamos executando o comando **ls /tmp** dentro do container, sem precisar conectar no TTY alocado para o container. Iremos utilizar esse comando para gerenciar os containers.



## Anotações:

## Gerenciando Imagens e Criando Templates

Imagens do Docker são a base para os containers.  
Elas contêm o filesystem (estrutura de diretórios)  
para criação dos containers.



No ambiente da Dexter teremos uma imagem que  
será utilizada para criação do ambiente Blue e  
Green, nessas imagens teremos o agente do  
puppet instalado e alguns pacotes básicos.



104



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

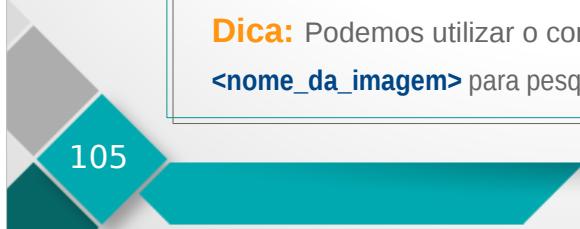
---

## Gerenciando Imagens e Criando Templates

Máquina: Docker

- 1 Realize o download da imagem do centos.  
2# docker pull centos
- 2 Liste as imagens baixadas pelo Docker.  
1# docker images
- 3 Remova a imagem do centos.  
3# docker rmi centos

**Dica:** Podemos utilizar o comando **docker search <nome\_da\_imagem>** para pesquisar imagens no **Docker Hub**.



105



### Docker images

O comando **images** lista as imagens armazenadas no repositório local do docker.

### Docker rmi

O comando é utilizado para remover uma ou mais imagens. Você não pode remover a imagem de um container enquanto ele estiver rodando, a não ser que você utilize o parâmetro **-f** para forçar a remoção dos slides.

### Docker pull

O comando é utilizado para realizar o Download de uma imagem do repositório, normalmente no repositório oficial do Docker. Caso tenha duas ou mais imagens com o mesmo nome, todas serão baixadas para o repositório local da máquina.

## Gerenciando Imagens e Criando Templates

Máquina: Docker

Agora, através do Dockerfile vamos criar a imagem que iremos utilizar no laboratório final do curso.

1

Vamos criar o diretório dos templates da Dexter.

```
1# mkdir-p /root/dockerfiles/  
2# cd /root/dockerfiles/
```

2

Dentro do diretório, criaremos o arquivo “Dockerfile” que será interpretado pelo builder do docker e nele passaremos as instruções necessárias para criação da nossa imagem.

106



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

O Docker pode construir imagens automaticamente lendo as instruções de um Dockerfile. Um Dockerfile é um arquivo de texto que contém todos os comandos que um usuário pode executar na linha de comando, entre outras funções. O Docker pode fazer a compilação automatizada que executa várias instruções de linha de comando em sequência.

O comando de build do docker cria uma imagem a partir de um Dockerfile e um contexto. O contexto da compilação são os arquivos em uma localização específica PATH ou URL. O PATH é um diretório em seu sistema de arquivos local. O URL é um local de repositório do Git. Um contexto é processado recursivamente. Assim, um PATH inclui quaisquer subdiretórios e a URL inclui o repositório e seus sub-módulos.

Um comando de compilação simples que usa o diretório atual como contexto:

```
# docker build .
```

Tradicionalmente, o arquivo Dockerfile é chamado de Dockerfile e localizado na raiz do contexto. Você usa o parâmetro -f para apontar a um Dockerfile em qualquer lugar em seu sistema de arquivos.

```
# docker build -f /path/to/a/Dockerfile .
```

## Gerenciando Imagens e Criando Templates

Máquina: Docker

Para que o nosso container já nasça com a chave privada e o arquivo de configuração do puppet agent, precisamos adicionar dentro do diretório web.

1

Copiar a chave privada que foi gerada na máquina devops.

```
1# scp root@devops:/etc/keys/key.pem /root/dockerfile/id_rsa
```

2

Vamos gerar o arquivo de configuração do puppet agent, e o config do ssh.

```
1# vim puppet.conf
```

```
2# vim config
```

108



**1# vim puppet.conf**

```
[main]
logdir=/var/log/puppet
vardir=/var/lib/puppet
ssldir=/var/lib/puppet/ssl
rundir=/run/puppet
factpath=$vardir/lib/facter
prerun_command=/etc/puppet/etckeeper-commit-pre
postrun_command=/etc/puppet/etckeeper-commit-post
[agent]
server = devops.dexter.com.br
```

**2# vim config**

Host \*

```
StrictHostKeyChecking no
UserKnownHostsFile=/dev/null
```

## Gerenciando Imagens e Criando Templates

Máquina: Docker

3

Vamos criar o arquivo “Dockerfile”.

```
2# vim Dockerfile  
FROM ubuntu  
RUN apt-get update  
RUN apt-get install puppet wget git vim -y  
RUN mkdir -p /root/.ssh/  
COPY config /root/.ssh/  
COPY id_rsa /root/.ssh/  
COPY puppet.conf /etc/puppet/puppet.conf
```

109



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Gerenciando Imagens e Criando Templates

Máquina: Docker

4

Entendendo cada parâmetro do dockerfile.

FROM – A imagem que será usada como base;

MAINTAINER – Contato de quem mantém a imagem;

RUN – Comandos que serão executados;

ENV – Tag para definir variáveis de ambiente;

CMD – Executa um comando com o container já “running”, oposto do RUN que executa durante o build;

COPY – Copia novos arquivos e diretórios e os adicionam ao filesystem do container.

110



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Gerenciando Imagens e Criando Templates

Máquina: Docker

5

Após criar todo o cenário para o nosso container, podemos fazer build.

1# docker build -t deploy .

2# docker images



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

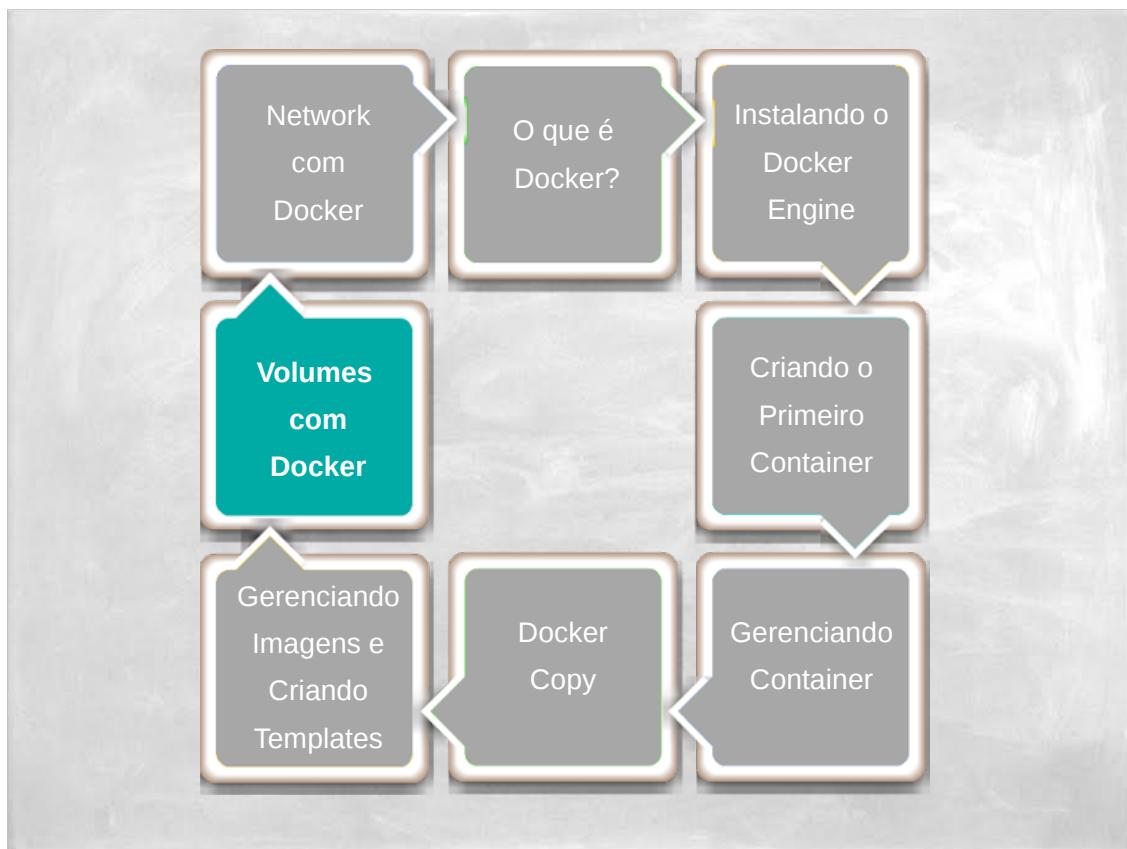
---

---

---

---

---



## Anotações:

## Gerenciando Volumes

Você pode trabalhar com volumes compartilhados entre os containers.

Por padrão, o Docker cria um volume para cada container, porém, você pode compartilhar um diretório ou arquivo da máquina hospedeira com os containers, criar volumes pelo Docker ou até mesmo criar containers e compartilhar o seu volume com outros containers.



113

4LINUX  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Gerenciando Volumes

Máquina: Docker

- 1 Crie o diretório na máquina em que será o container.

```
1# mkdir -p /srv/data
```

- 2 Crie o container mapeando o diretório da máquina host para container.  
*(Execute na mesma linha!)*

```
2# docker run -dit --name segundo -v /srv/data:/data debian  
/bin/bash
```

- 3 Crie um arquivo de teste no novo diretório.

```
3# echo "Teste Volume" > /srv/data/arg01.txt
```

- 4 Liste o diretório dentro do container.

```
4# docker exec segundo ls /data
```

114



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Gerenciando Volumes

Máquina: Docker

- 1 Crie um volume no Docker.

```
1# docker volume create --name data
```

- 2 Crie o container mapeando o volume para container. (Execute na mesma linha!)

```
2# docker run -it --name volume-teste -v data:/data debian  
/bin/bash
```

**Dica:** Você pode particionar ou montar um Storage no diretório **/var/lib/docker/volumes**, onde o docker armazena os volumes e seus arquivos.

115

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Gerenciando Volumes

### Dicas importantes para usar volumes no Docker



Múltiplos containers podem compartilhar um ou mais volumes. Porém, vários containers escrevendo em um mesmo volume compartilhado podem corromper os dados.

Volumes podem ser acessados diretamente pela máquina Hospedeira. Isso quer dizer que você pode ler e escrever através da própria máquina. No entanto, na maioria dos casos não é recomendado o uso, pois pode corromper os dados.

116

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

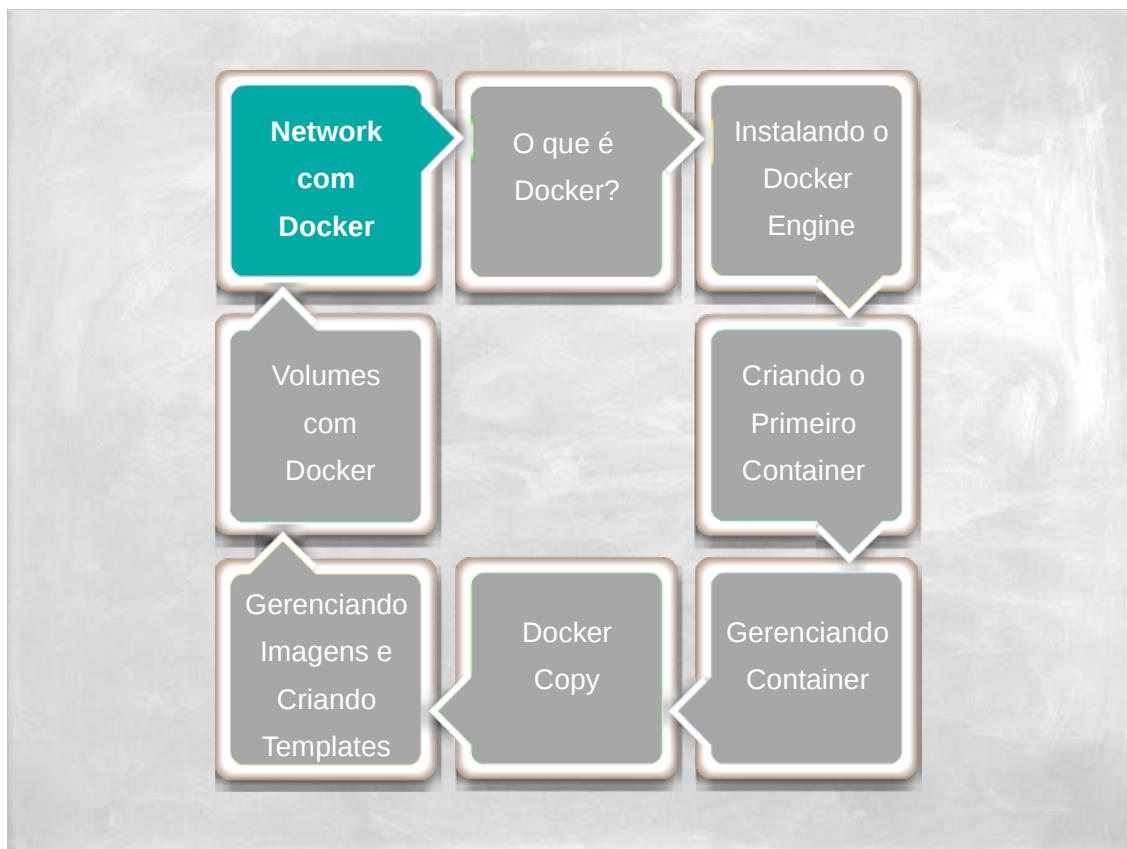
---

---

---

---

---



## Anotações:

## Network com Docker

O **Docker Engine** providencia uma rede interna entre os containers. Essa rede interna é criada automaticamente durante a instalação do pacote.

Para comunicação dos containers com a rede externa da máquina, o Docker cria uma interface de rede virtual que se comunica com a rede da máquina hospedeira, sendo que é utilizado o gateway para os containers.



118

4LINUX  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Network com Docker

Máquina: Docker

Podemos mapear a porta da máquina host para os containers com parâmetros **-p** na criação dos containers. Qualquer conexão na porta especificada será redirecionada para o container em questão.

- 1 Crie um container do Docker como o nome **webs** recebendo conexões pela porta 80 da máquina host.

```
1# docker run -it -p 80:80 --name webs nginx /bin/bash
```

- 2 Inicie o serviço do Nginx dentro do container.

```
2# service nginx start
```

- 3 Desconecte do container sem parar sua execução.

```
3# CTRL+P + CTRL+Q
```

119



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Network com Docker

A partir da versão 1.9.0 do **Docker Engine**, foi adicionado o gerenciamento da rede entre os containers, disponibilizando a criação de novas redes internas.



Iremos utilizar no nosso laboratório final uma rede para os containers da Dexter chamada de **Dexterlan**.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Network com Docker

Máquina: Docker

- Crie uma rede interna do Docker com o nome de **Dexterlan**.

```
1# docker network create --subnet 10.0.0.0/16 dexterlan
```

- Crie o container de teste utilizando a nova rede criada.

(Execute na mesma linha!)

```
2# docker run -it --name node01 --hostname node01 --net dexterlan  
debian /bin/bash
```

- Podemos conectar um container já criado à nova rede com o subcomando **connect**. Conecte o container criado anteriormente à nova rede de containers da Dexter.

```
3# docker network connect dexterlan webs
```

121



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Network com Docker

A partir da versão 1.10.0 do **Docker Engine**, foi disponibilizado a possibilidade de trabalhar com endereços de IP fixo para os containers.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Network com Docker

Máquina: Docker

1

Crie um container passando o endereço fixo:  
(Execute na mesma linha!)

```
1# docker run -dit --name webnode --hostname webnode  
--ip 10.0.0.10 --net dexterlan debian /bin/bash
```

2

Verifique o endereço IP do novo container criado:

```
2# docker exec webnode ip a
```

**Atenção:** Para utilizar o parâmetro **-ip**, é obrigatório criar uma rede no docker especificando o endereço da rede.

123

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

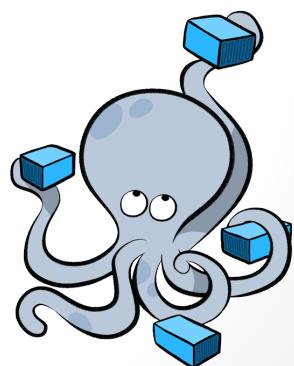
## Multiplos containers com docker-compose

1

O docker-compose é utilizado quando se há necessidade de criar mais de um container para a mesma aplicação, ou seja, ele constrói vários containers e cria ligações entre eles.

### Por exemplo:

Um wordpress precisa totalmente do banco de dados, e para segmentar cada serviço em um container vamos usar o compose.



124

4LINUX  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

Compose é uma ferramenta que possibilita a execução de vários containers Docker para uma mesma aplicação. Após criar o arquivo Compose para configurar os serviços da aplicação, e na sequência utilizando um único comando, você cria e inicia todos os serviços de sua configuração com o comando.

```
# docker-compose up
```

O docker compose é frequentemente utilizado para “Ambientes de desenvolvimento” e “Ambientes de testes automatizados”.

## Multiplos containers com docker-compose

Máquina: Docker

1

A instalação do docker-compose é bem simples, baixar diretamente do **github** oficial e salvar no servidor, em seguida dar permissão de execução.

```
1# curl -L
```

```
https://github.com/docker/compose/releases/download/1.15.0/docker-  
compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose
```

```
2# chmod +x /usr/local/bin/docker-compose
```



**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Multiplos containers com docker-compose

Máquina: Docker

1

Vamos criar a estrutura de diretórios para o compose.

```
1# mkdir -p /root/compose/wordpress && cd /root/compose/wordpress
```

```
2# vim docker-compose.yml
```

2

Dentro do arquivo, passaremos algumas instruções para o compose.

```
wordpress_dexter:  
  image: wordpress  
  links:  
    - mysql_dexter:mysql  
  ports:  
    - 8080:80  
mysql_dexter:  
  image: mariadb  
  environment:  
    MYSQL_ROOT_PASSWORD: dexter
```

127



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Multiplos containers com docker-compose

Máquina: Docker

- Vamos entender os campos do docker-compose:

**wordpress\_dexter** – O nome que daremos a nossa imagem;  
**image** – Qual imagem ele usará como base para fazer build;  
**links** – Qual container será linkado no momento do build;  
**Ports** – Qual porta será feito encaminhamento (host para o container);  
**mysql\_dexter** - O nome que daremos a nossa segunda imagem, agora com o banco de dados;  
**Image** – Imagem que usaremos como base para a nossa segunda imagem;  
**Environment** – Variável de ambiente que passaremos para o container, nesse caso com a senha.



128

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Multiplos containers com docker-compose

Máquina: Docker

2

Vamos fazer build dos nossos containers e em seguida, verificar se estão up.

```
#1 Docker-compose up -d
```

```
#1 Docker ps
```

No momento da criação nós já criamos ligação entre os containers, com isso a nossa aplicação php já se comunica com o banco de dados. Vamos acessar pelo navegador da máquina Desktop.

<http://docker.dexter.com.br:8080>



**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Documentação

Para mais informações sobre o Docker, você pode acessar a documentação pelo site oficial ou utilizar a documentação disponibilizada nas páginas de manual do sistema GNU/Linux:

**Link:** <https://docs.docker.com/>



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

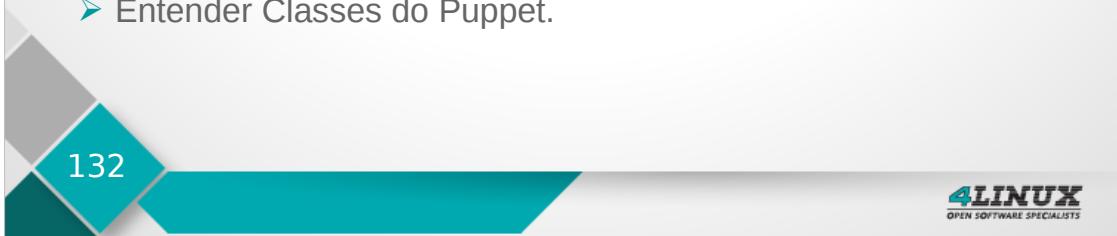
---

---

## Gerencia de Configuração com Puppet

### Objetivos da Aula

- Fazer uma introdução ao Puppet;
- Instalar o Puppet Agent;
- Entender os resources (recursos) do Puppet;
- Instalar e configurar o Puppet Master;
- Compreender e criar módulos no Puppet;
- Entender Classes do Puppet.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

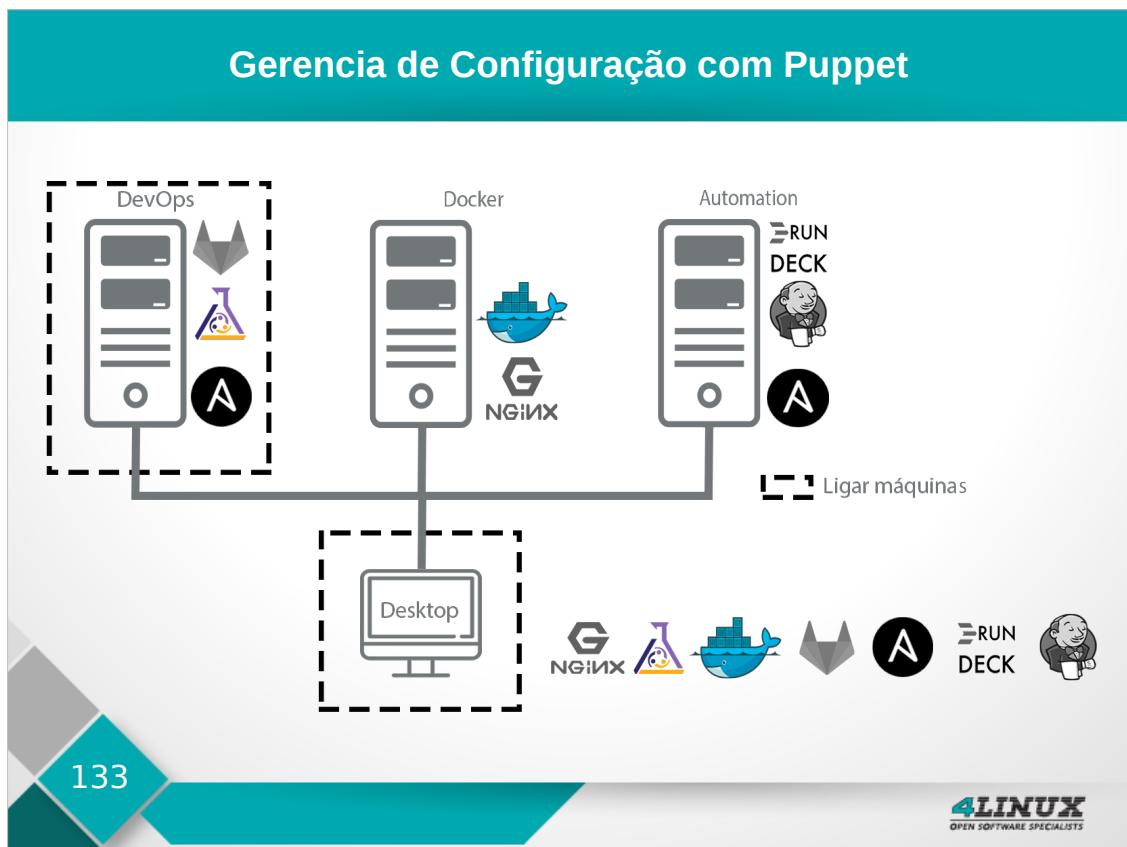
---

---

---

---

---



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

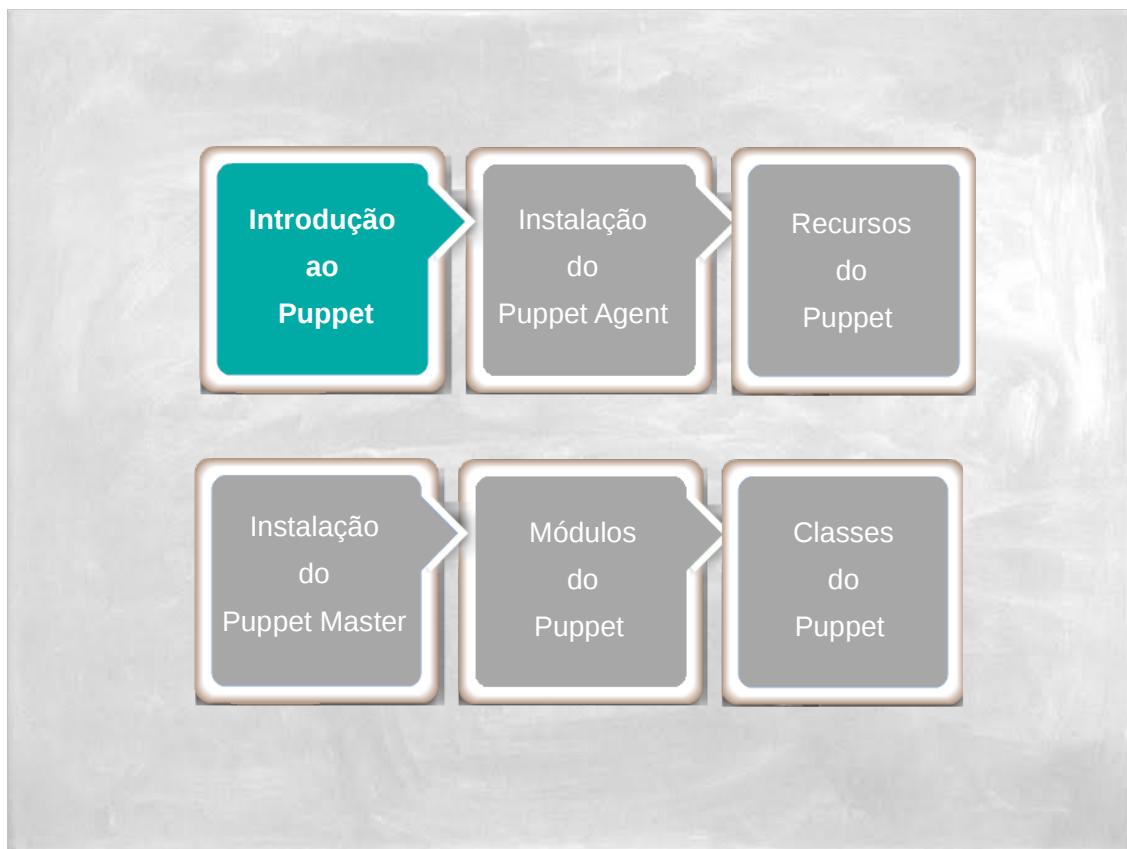
---

---

---

---

---



## Anotações:

## Introdução ao Puppet



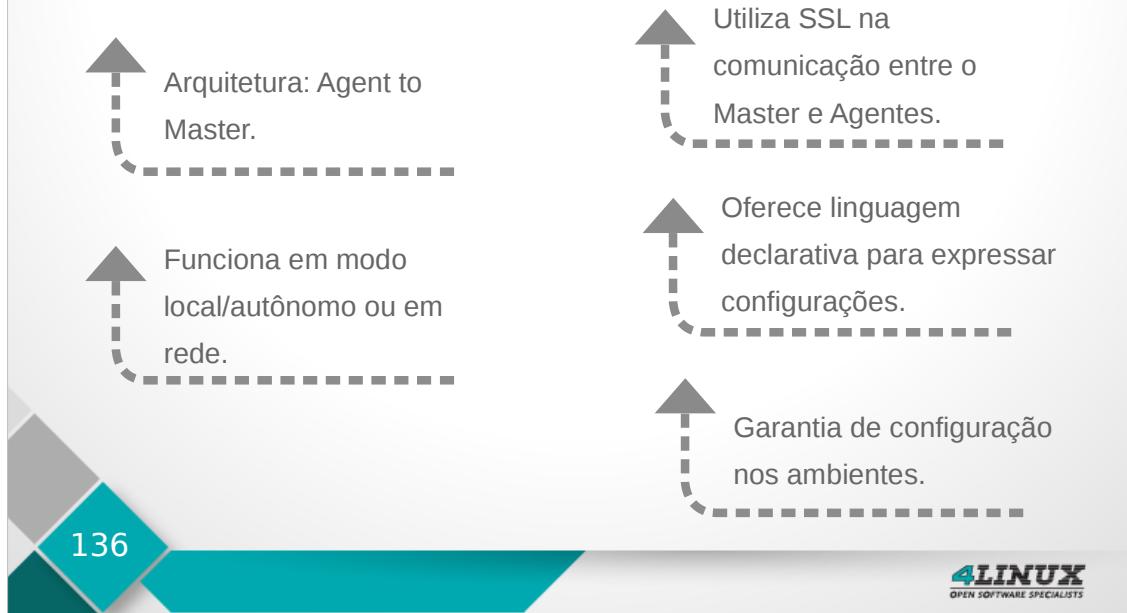
O **Puppet** é uma ferramenta de gerência de configurações que suporta várias plataformas. Ele possui sua própria linguagem declarativa, que é desenvolvida sobre a arquitetura de agente (cliente) e master (servidor).



A gerência de configuração oferece um conjunto de recursos e métodos que tem o objetivo de garantir a integridade das configurações de nossos sistemas, serviços e Infraestrutura envolvida, fazendo isto de forma: **ágil, controlada e automatizada**.

## Introdução ao Puppet

### Características do Puppet



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

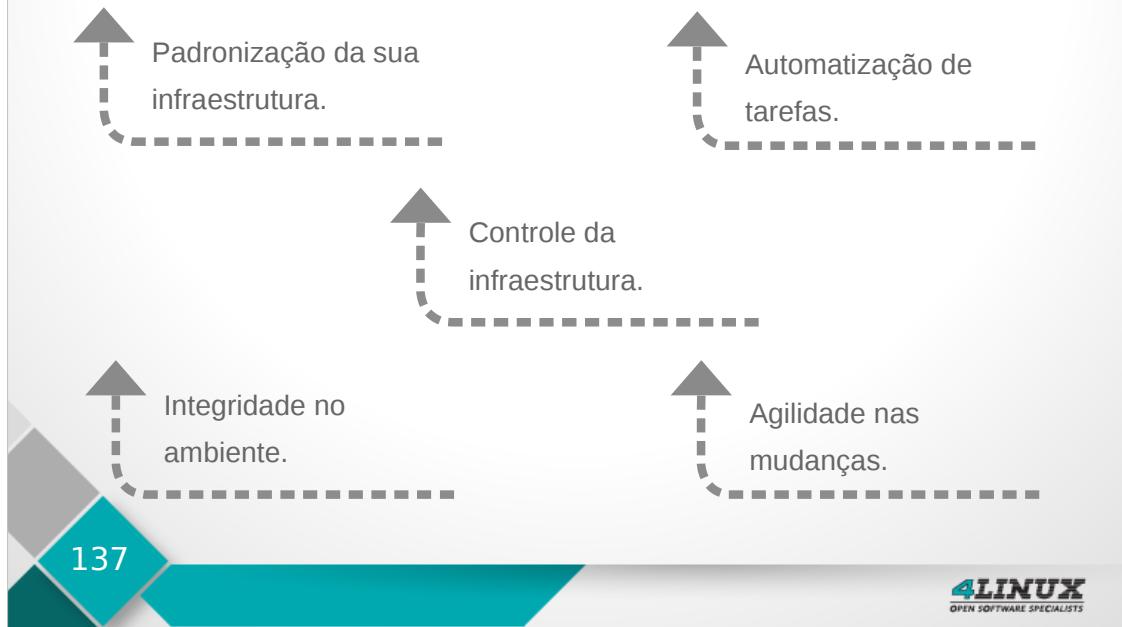
---

---

---

## Introdução ao Puppet

### Vantagens na Utilização do Puppet



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Introdução ao Puppet

### Exemplo de configuração: sem uso do Puppet

```
# aptitude install ntp  
# update-rc.d ntp defaults  
# cp ~/ntp.conf /etc/  
# service ntp restart
```

**Debian**

**RedHat**

```
# yum install ntp  
# chkconfig ntp on  
# cp ~/ntp.conf /etc/  
# service ntp restart
```

138



Trabalhando de forma manual, além de você ter processos repetitivos você precisa ter o conhecimento sobre o sistema operacional que você está usando. O puppet criar uma abstração entre o sistema utilizado.

## Introdução ao Puppet

### Exemplo de configuração: usando o Puppet

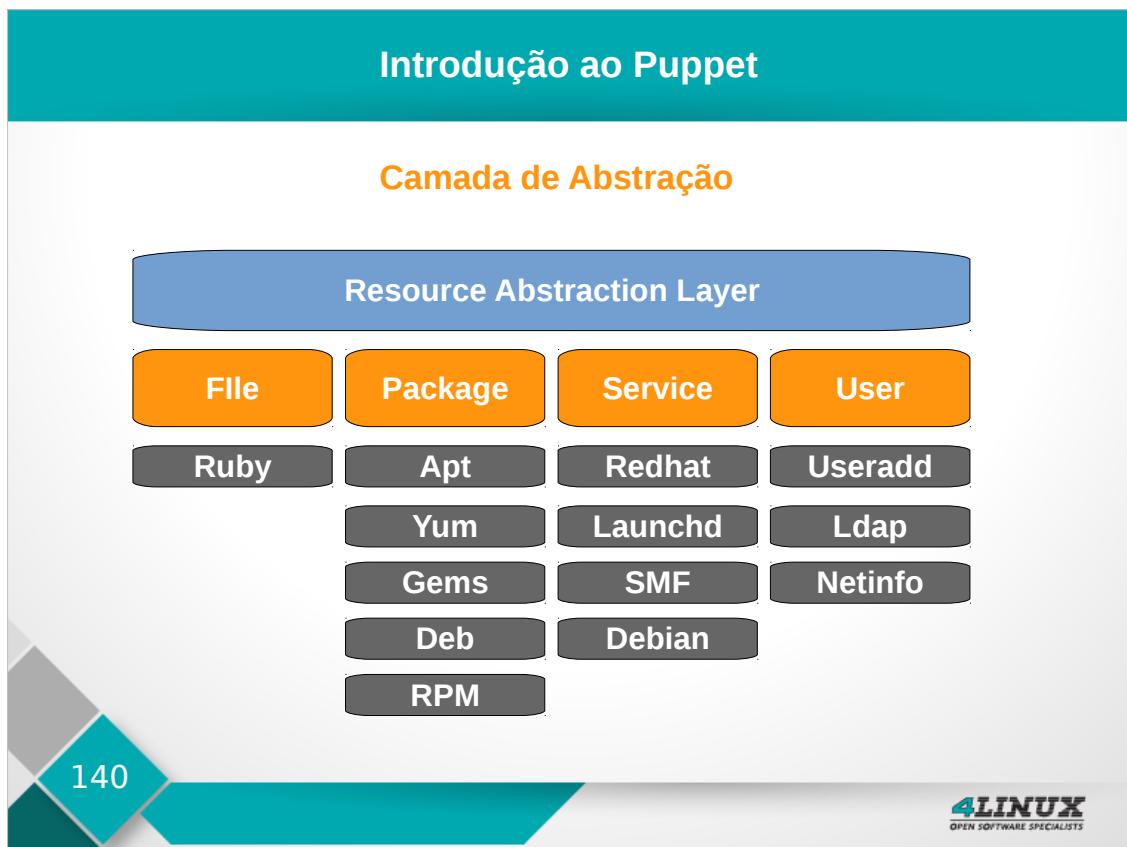
```
package { 'ntp':
  ensure => present,
}
service { 'ntp':
  ensure => running,
  enable => true,
}
file { '/etc/ntp.conf':
  ensure => file,
  source => "/backup/puppet/files/ntp/ntp.conf"
}
```

139



Desta forma, não importa qual sistema operacional, o puppet se encarrega de identificar através do facter qual sistema operacional e qual ferramente deve ser utilizada para realizar a ação.

OBS: Leia a documentação do Puppet para descobrir quais sistemas operacionais o puppet suporta, e opções e utilização do recurso para cada distribuição.



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Introdução ao Puppet

### Puppet Autônomo X Puppet Server

**Puppet Autônomo:** É também chamado de serverless. Neste modelo executamos o puppet localmente na máquina a fim de manipular diversos estados e configurações do sistema.

**Puppet Server:** É o modelo mais utilizado. As configurações ficam armazenadas no servidor e o agente se comunica com o servidor do puppet (puppet master), recolhendo as informações e depois aplicando-as na máquina.

141



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

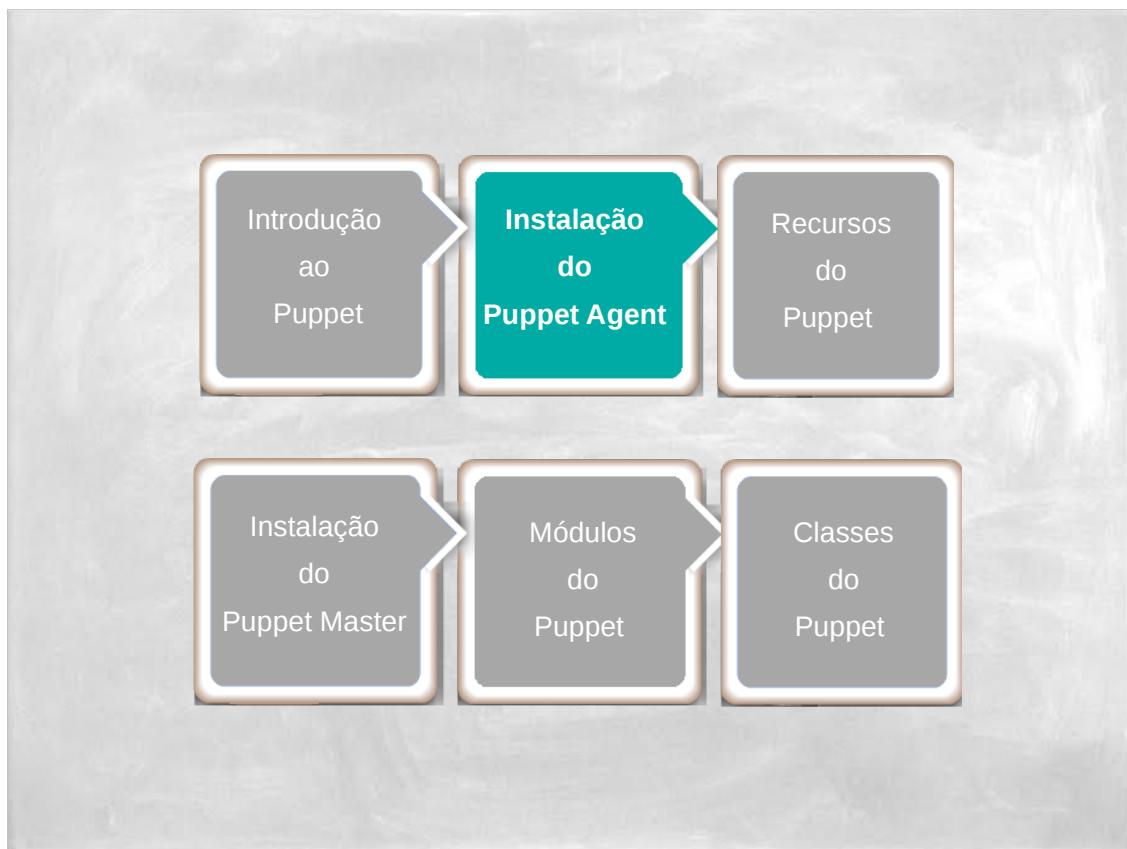
---

---

---

---

---



## Anotações:

## Instalação do Puppet Master

Máquina: Devops

- 1 Vamos utilizar o playbook criado com ansible para automatizar a instalação do puppet. Abra o arquivo main.yml da role do Puppet para que possamos conferir os passos para a instalação:

```
1# vim /etc/ansible/roles/puppet/tasks/main.yml
```

- 2 Para executar o playbook, podemos usar o comando:

```
1# ansible-playbook /etc/ansible/playbook/ambiente/devops.yml
```

143



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

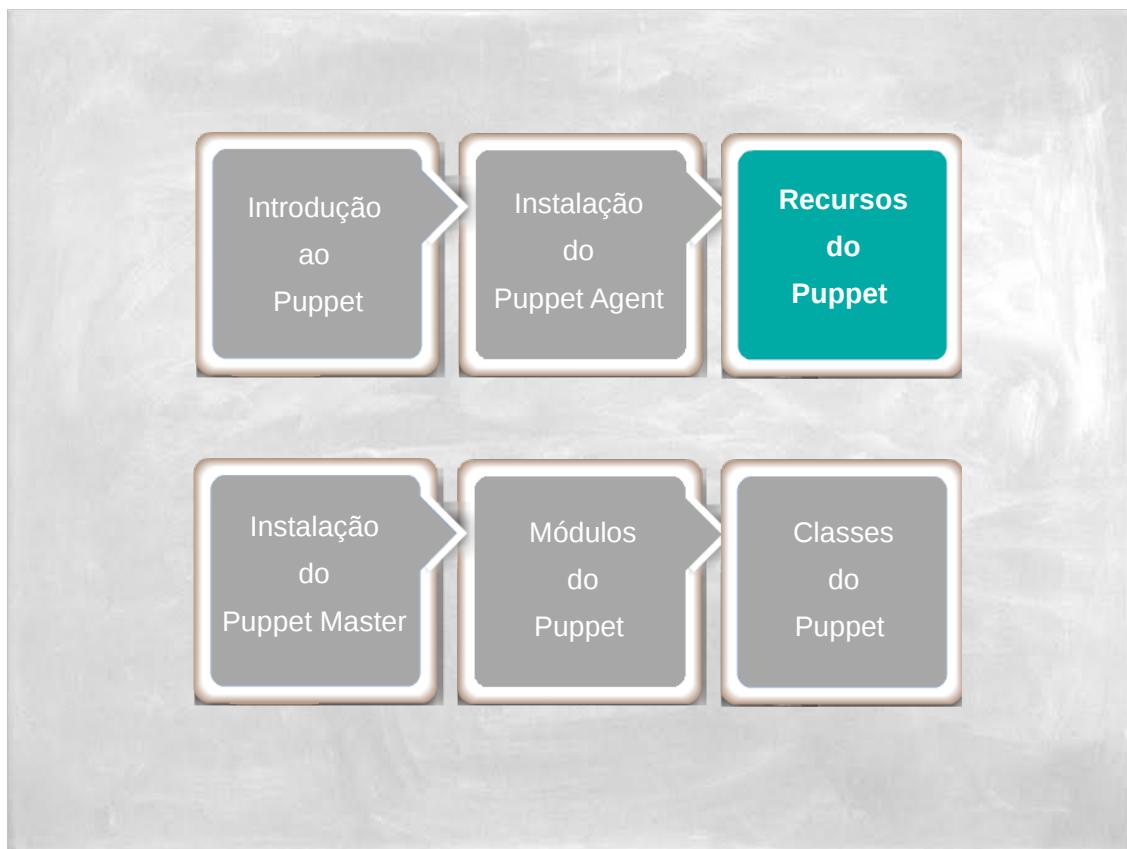
---

---

---

---

---



## Anotações:

## Recursos do Puppet

Os recursos (**Resources**) são utilizados para descrever um aspecto do sistema, como especificar um serviço, pacotes ou arquivo.

Cada recurso é associado a um tipo (**Resources Type**) que determina o tipo de configuração gerenciada.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Recursos do Puppet

Conheça alguns recursos built-in do Puppet:

– service

– group

– file

– notify

– user

– cron

– package

– mount

– exec

146



**Service:** Responsável por gerenciar os serviços do sistema. O puppet espera que o script do serviço tenha um comando *status*, caso o script não tenha e necessário que o parâmetro *hasstatus* seja definido como *false*, é possível especificar um comando para verificar o status do serviço. Em ultimo caso o puppet irá analisar a tabela de processo rodando no sistema procurando pelo nome do serviço.

### Atributos:

- Name: o nome do serviço do serviço a ser gerenciando, caso não for especificado será utilizado o título do recurso como o nome do serviço.
- Ensure: Controla o estado do serviço, por exemplo **running**.
- Enable: Habilitar o serviço para iniciar no processo de boot da máquina.
- Restart: Especificar um comando de restart do serviço manualmente.
- Start: Especificar um comando de start do serviço manualmente.
- Status: Especificar um comando de status do serviço manualmente.
- Stop: Especificar o comando de stop do serviço manualmente.

**User:** Responsável por gerenciar usuários do sistema. Esse recurso usa ferramentas nativas para criação de usuários no sistema e utiliza APIs do POSIX para recolher informações dos usuários, o recurso não modifica diretamente o arquivo /etc/passwd.

**Atributos:**

- Name: o nome do usuário, caso não for especificado será utilizado o título do recurso como o nome do serviço.
- Ensure: o estado do usuário especificado, como *present* garante que usuário esteja criando no sistema, *absent* garante que o usuário não esteja presente no sistema.
- Comment: Define uma descrição ao usuário.
- Gid: Define o grupo primário do usuário.
- Groups: Define outros grupos que usuário pertence.
- Home: Define o diretório home do usuário.
- System: Define se é usuário de sistema ou não. O padrão é *false*.

**Mount:** Responsável por gerenciar sistema de arquivos montados no sistema. Inclusive inserindo informação na tabela de partição (fstab) para garantir que mesma seja montada no boot do sistema.

**Atributos:**

- Name: o nome do caminho a ser montado, caso não for especificado será utilizado o título do recurso como o nome do serviço.
- Ensure: Controla o estado do sistema de arquivo montando.
- Atboot: Se o partition deve ser montado no processo de boot do sistema.
- Device: O dispositivo que será montado no diretório.
- Fstype: O tipo de sistema de arquivo (filesystem) do dispositivo.
- Options: As opções de montagem do dispositivo.

**Group:** Responsável por gerenciar grupos do sistema.

**Atributos:**

- Name: o nome do grupo a ser gerenciado, caso não for especificado será utilizado o título do recurso como o nome do serviço.
- Ensure: Controla o estado do grupo, se o grupo deve estar present ou não no sistema.
- Gid: Define o ID do grupo.
- Members: Define os membros desse grupo.
- System: Define se é um grupo de sistema ou não.

**Exec:** Executa um comando externo no sistema. O comando não pode ter interação de usuário ou seja não pode depender de nenhuma ação.

#### Atributos:

- command: o comando a ser executado, caso não for especificado será utilizado o título do recurso como o nome do serviço.
- CWD: O Diretório de onde o comando deve ser executado. Caso o diretório não existe o comando retorna falha de execução.
- Environment: Definir variáveis ambientais durante a execução do comando.
- Onlyif: Quando esse parâmetro é definido, o comando só será executado caso o comando especificado no onlyif retorna o código 0.
- Refreshonly: Define que o comando só deve ser executado quando o estado de outro objeto for alterado, por exemplo quando o conteúdo de um arquivo ser alterado.
- Timeout: Define o timeout do comando.
- Tries: Define o número de tentativas do comando.
- Umask: Define umask durante a execução do comando.
- Path: O diretório usado para procura do comando. Valor da variável \$PATH.

**Package:** Responsável por gerenciar os pacotes, o puppet automaticamente tenta identificar o formato do pacote baseado na plataforma, porém você pode passar através do atributo **provider**.

#### Atributos:

- Provider: Especifica o backend para usar no gerenciamento dos pacotes. Opções como: aptitude, aix, dpkg, openbsd, pacman, pip, pkg, rpm, yum, zypper.
- Name: O nome do pacote, caso não for especificado será usado o título do recurso.
- Ensure: Define o status do pacote no sistema. Opções como: installed ou present garantem que o pacote esteja instalado no sistema, absent que garante que o pacote não esteja instalado no sistema.

**Cron:** Instala e gerencia tarefas do cron. É obrigatório passar um comando e no mínimo um período de configuração da cron (minuto, hora, mês, dia do mês, dia da semana ou alguns períodos especiais).

#### Atributos:

- name: O nome simbólico da tarefa do cron.
- Ensure: Garante o status da job, como present ou absent.

- Command: O Comando a ser executado na job.
- Environment: Define variaveis de configuração associada com a job do cron.
- Hour: Define a hora que tarefa deve ser executada.
- Minute: Define o minuto que tarefa deve ser executada
- Month: Define o mês que tarefa deve ser executada
- Monthday: Define o dia que tarefa deve ser executada
- Weekday: Define o dia da semana que tarefa deve ser executada
- Special: Define um valor especial para período de execução da jobs, como por exemplo: annualy ou reboot.

**File:** Responsável por gerenciar arquivos, incluindo o conteúdo do arquivo, os permissão e donos do arquivo.

**Atributos:**

- Path: Caminho completo do arquivo a ser gerenciando.
- Ensure: Gerencia o status do arquivo, como por exemplo: present, absent, directory.
  - Present: Garante a presença do arquivo no sistema.
  - Absent: Garante que o arquivo não exista no arquivo.
  - Directory: Garante a presença de um diretorio.
- Backup: Cria um arquivo de backup antes de alterar o conteúdo do arquivo.
- Content: Uma string para definir o conteúdo do arquivo.
- Group: Define grupo dono do arquivo.
- Owner: Define o usuário dono do arquivo.
- Mode: Define a permissão do arquivo.
- Checksum: O algorítimo utilizado para determina se o arquivo deve ser alterado ou não. exemplos: md5, sha256, mtime, ctime or none.

## Recursos do Puppet

Máquina: Devops

1

Vamos utilizar o recurso de gerenciamento de usuários.

```
1# puppet resource user
2# puppet resource user root
3# puppet resource user linus
4# puppet resource user linus ensure=present
5# puppet resource user linus ensure=absent
6# puppet resource user linus ensure=present home="/srv/linus"
managehome=true
```



150



Vamos analisar o comandos usados acima:

**\$ puppet resource user**

No exemplo, estamos utilizando o resouce para gerenciamento de usuário, quando executado sem nenhum parâmetros ele retorna todos os usuário e sua opções.

**\$ puppet resource user root**

Podemos consultar o usuário específico passando como parâmetro o seu nome, caso o usuário exista o comando irá trazer as informações do usuário, se o usuário não existir no sistema o comando informa que o usuário está **absent**.

**\$ puppet resource user linus ensure=present**

Para criar um usuário no sistema podemos utilizar o parâmetro ensure, que utilizado no puppet quando queremos trocar o estado do sistema.

```
$ puppet resource user linus ensure=absent
```

Para deletar o usuário podemos utilizar o valor **absent**.

```
$ puppet resource user linus ensure=present
```

```
home="/srv/homes/linus" managehome=true:
```

Na criação de usuário podemos passar opções que define as informações do usuário como por exemplo: **home** que especifica a home do usuário, e parâmetro **managehome** que define se a home do usuário deve ser criada na criação do mesmo.

## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Recursos do Puppet

Máquina: Devops

1

Vamos utilizar o recurso de gerenciamento de serviços do Puppet.

```
1# puppet resource service
2# puppet resource service cron
3# puppet resource service ntp
4# puppet resource service cron ensure=stopped
5# ps aux | grep cron
6# puppet resource service cron ensure=running
7# px aux | grep cron
```

152



Vamos analisar o comandos usados acima:

```
$ puppet resource service
```

O recurso service é utilizado para gerenciar os serviços do sistema tais como ssh, apache, bind, ftp, ntp entre outros. O comando acima irá listar todos os serviços e o status atual no sistema.

```
$ puppet resource service cron ensure=stopped
```

```
$ puppet resource service cron ensure=running
```

Com parâmetro *ensure* podemos alterar o estado do serviço , para garantir que serviço esteja rodando ou garantir que serviço esteja parado.

## Recursos do Puppet

Máquina: Devops

- Vamos utilizar o recurso de gerenciamento de pacotes.

```
1# puppet resource package
2# puppet resource package cron
3# puppet resource package ntpdate
4# dpkg -l | grep ntpdate
5# puppet resource package ntpdate ensure=present
6# dpkg -l | grep ntpdate
7# puppet resource package ntpdate ensure=absent
```

153



Vamos analisar o comandos usados acima:

```
$ puppet resource package
```

Comando é utilizado para listar os pacotes instalado no sistema.

Podemos utilizar o parâmetro *ensure* para alterar o estado dos pacotes no sistema:

como por exemplo:

Instalar o pacote do ntpdate:

```
$ puppet resource package ntpdate ensure=present
```

Desintalar o pacote do ntpdate:

```
$ puppet resource package ntpdate ensure=absent
```

## Facter

Máquina: Devops

- 1 Verifique se o Facter está instalado no sistema.  
`1# dpkg -l | grep facter`
- 2 Execute o comando para coletar todas as informações do sistema.  
`2# facter`
- 3 Execute o comando para consultar o hostname da máquina.  
`3# facter hostname`
- 4 Execute o comando para consultar o endereço IP da máquina.  
`4# facter ipaddress`

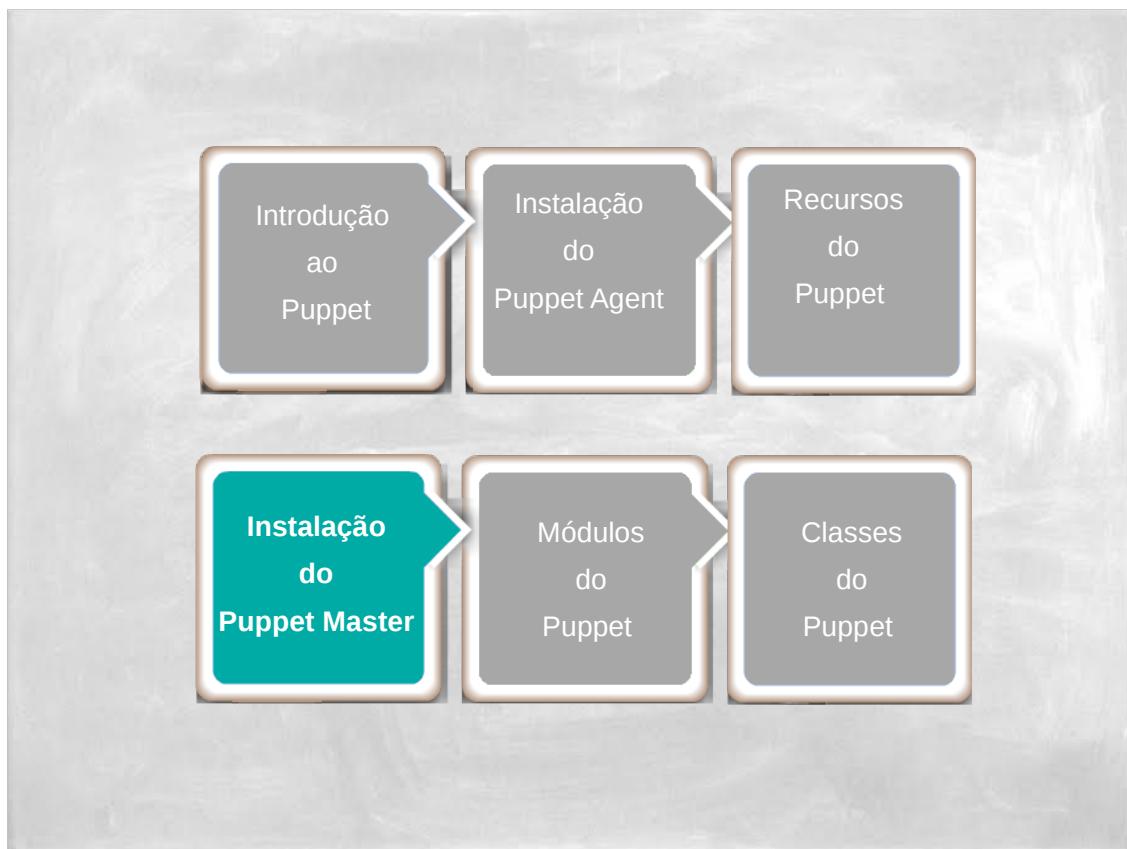
154

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

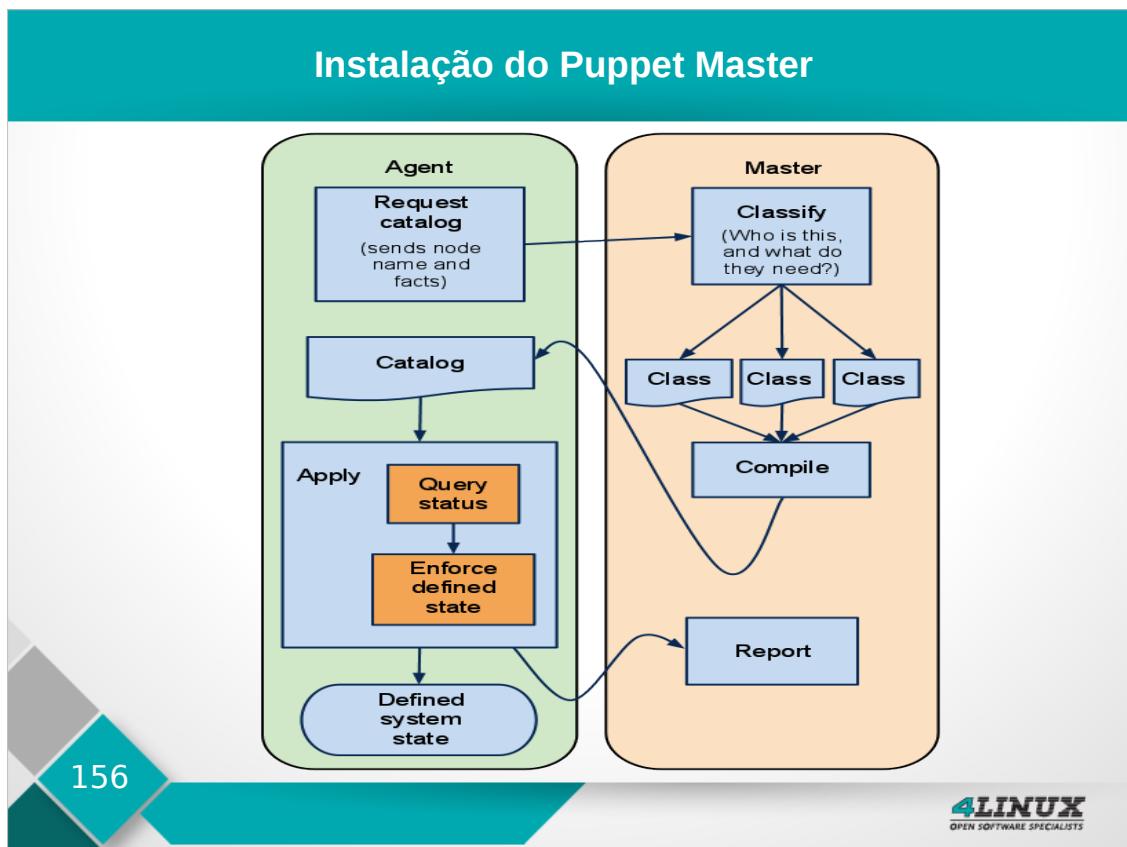
Facter é responsável por recolher informações sobre os nodes, tais como detalhes de hardware, configurações de rede, tipo, versão e família do sistema operacional, endereço IP, Mac address, Chaves SSH e muito mais. As variáveis do facter pode ser utilizado na manifest do Puppet. O facter pode ser estendido, e facilmente você consegue implementar suas próprias customizações.

O link abaixo contém a lista de todas as variáveis que compõe o sistema core do facter.

[http://docs.puppetlabs.com/facter/latest/core\\_facts.html](http://docs.puppetlabs.com/facter/latest/core_facts.html)



## Anotações:



Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Instalação do Puppet Master

Máquina: Devops

- 1 Abra o arquivo de configuração do Puppet e realize as alterações.

```
2# vim /etc/puppet/puppet.conf  
dns_alt_names=devops.dexter.com.br  
autosign=$confdir/autosign.conf
```

- 2 Vamos colocar “\*” para assinar todos automaticamente.

```
2# vim /etc/puppet/autosign.conf  
*.dexter.com.br
```

157



### Opção dns\_alt\_names:

Uma lista de nomes separados por vírgula que define o nome do host local. O nome é usado também para gerar o certificado da CA que será utilizada na comunicação do master com os agents.

### Opção environment:

Diretório onde fica armazenada os arquivos de ambiente do puppet, sendo que a pasta é um ambiente isolado do puppet, normalmente é utilizado em ambientes como production, homolog, test, entre outros ambientes.

### Opção autosign:

Habilita as opções de auto assinatura de certificados. Você pode especificar com o valor true e todos os certificados são assinados automaticamente. Se o valor for definido como false, nenhum certificado é assinado automaticamente. Ou você pode usar o caminho do arquivo e dentro do arquivo, você determina um padrão a ser auto-assinado pelo puppet master.

## Manifest do Puppet

Máquina: Devops

- 1 Vamos criar a estrutura do módulo web.

```
1# cd /etc/puppet/modules
```

- 2 Vamos gerar o módulo utilizando o próprio comando do Puppet.

```
2# puppet module generate puppet-web
```

What version is this module? [0.1.0] **1.0.0.**

Who wrote this module? [dexter] **Seu Nome**

How would you describe this module in a single sentence?

---> **Gerencia os servidores web da Dexter Courier.**

158



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Instalação do Puppet Master

Máquina: Devops

- 3 Vamos criar o nossa estrutura de Manifest.

```
1# cd puppet-web/manifests  
2# vim init.pp  
  
class puppet-web {  
    package { "apache2": ensure => installed }  
    ->  
    service { "apache2": ensure => "running" }  
    ->  
    file{/etc/apache2/sites-enabled/web.conf':  
        source => 'puppet:///modules/puppet-web/web.conf',  
        ensure => present}  
}
```

159



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Instalação do Puppet Master

Máquina: Devops

- 4 Crie o diretório "files", onde ficará o virtualhost que será mantido com o puppet.

```
1# cd ../  
2# mkdir files  
2# cd files  
3# vim web.conf  
  
Listen 80  
  
<Virtualhost 80>  
    DocumentRoot "/var/www/web"  
</Virtualhost>
```

160



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Instalação do Puppet Master

Máquina: Devops

- 5 Crie o manifest que iremos utilizar no laboratório Dexter, para fazer include.

```
1# cd /etc/puppet/manifests  
2# vim site.pp  
  
node 'default' {  
    include puppet-web  
}
```

161



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Instalação do Puppet Master

Máquina: Devops

- 6 Para que o puppet master esteja ouvindo na porta 8140, precisamos promovê-lo a master.

```
1# puppet master  
2# ss -ntpl | grep 8140
```

162



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Manifest do Puppet

Os **Manifests** em sua essência são os programas que escrevemos usando linguagem declarativa do Puppet.



É através deste **Manifest** que aplicamos configurações em um node. Os **Manifests** podem ser criados e aplicados no modo cliente e servidor (agent/master) ou no modo autônomo.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Manifest do Puppet

Máquina: Devops

- 1 Vamos validar a sintaxe do arquivo criado.

```
1# puppet parser validate manifests/site.pp
```

- 2 Valide o “code style” do manifest criado.

```
2# puppet-lint manifests/site.pp
```

- 3 Podemos usar o parâmetro -f para forçar a correção do código.

```
3# puppet-lint -f manifests/site.pp
```

- 4 Vamos validar o style do nosso manifest criado.

```
4# puppet-lint manifests/site.pp
```

164



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Manifest do Puppet

Máquina: Devops

1

Acesse a Máquina Docker, e crie um container para testar a configuração do nosso Manifest:

```
1# docker run -it  
--name node1  
--add-host devops.dexter.com.br:192.168.200.100  
deploy /bin/bash
```

165



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Manifest do Puppet

Máquina: Devops

- Aplicar o manifest no Agent do Puppet.

```
1# puppet agent --enable  
2# puppet agent -t --server devops.dexter.com.br
```

166



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Meta-parâmetros

**Meta-parâmetros:** Em geral os meta-parâmetros afetam o comportamento dos recursos do puppet. Eles são usados para adicionar metadados, prevenir que o puppet faça mudanças, mudar informações do login e, no nosso caso, iremos utilizar os meta-parâmetros para criar dependências entre os recursos.



**Before:** Define um ou mais recursos que depende dele, múltiplos recursos podem ser representados por um array como referência.

**Require:** Define um ou mais recursos no qual ele é dependente, múltiplos recursos podem ser representados por um array como referência.

Exemplo:

```
Package { 'apache2': ensure => present }
```

```
Service {
    Ensure => running,
    Enable => true,
    Require => Package [ 'apache2' ],
}
```

No exemplo acima, o recurso service será executado depois do package, sempre garantindo que o pacote seja instalado primeiro para depois gerenciar o status do serviço.

**Subscribe:** Define um ou mais recursos no qual ele depende, caso o puppet realizar qualquer mudança no recurso subscrito, e enviando um refresh para o recurso especificado.

O evento de Refresh muda de acordo com o tipo de recurso especificado, por exemplo: para recursos como service é realizado um restart, mount o dispositivo e desmontando e remontando, nem todos os recursos possuem o evento de refresh.

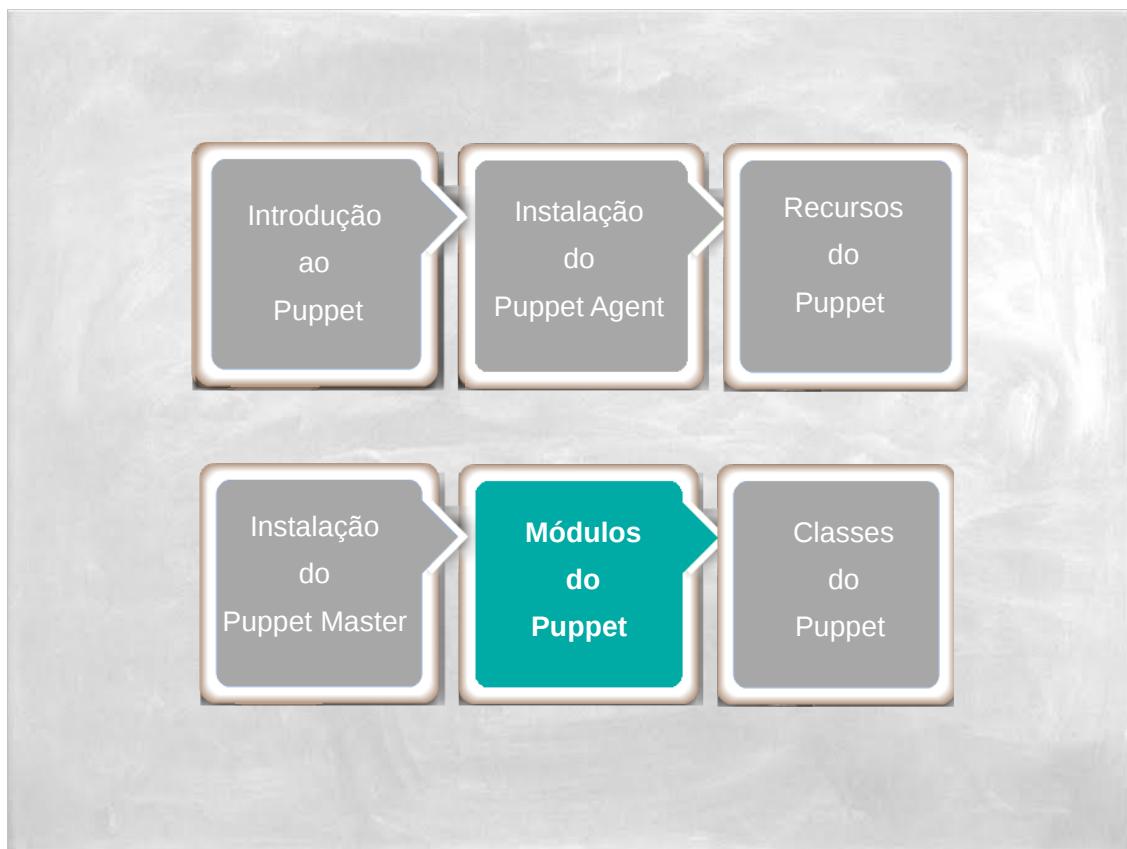
Exemplo:

```
package { 'apache2': ensure => present }

service {
    Ensure => running,
    Enable => true,
    Require => Package ['apache2'],
    Subscribe => File['/etc/apache2/apache2.conf'],
}

file {
    source => "puppet:///fileservers/apache/apache2.conf",
    owner => root,
    require => Package['apache2'],
}
```

No exemplo acima, caso o puppet realizar alguma mudança no arquivo de configuração do apache, o serviço será reiniciando após a alteração.



## Anotações:

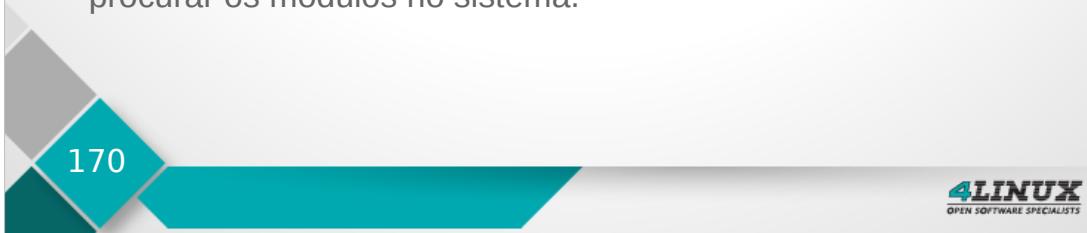
## Módulos do Puppet

**Módulos** são conjuntos de código e dados que podem ser reutilizados.



O Puppet carrega automaticamente os módulos a partir de arquivos manifest presentes dentro dos módulos.

O Puppet analisa o conteúdo da variável modulepath para procurar os módulos no sistema.



**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

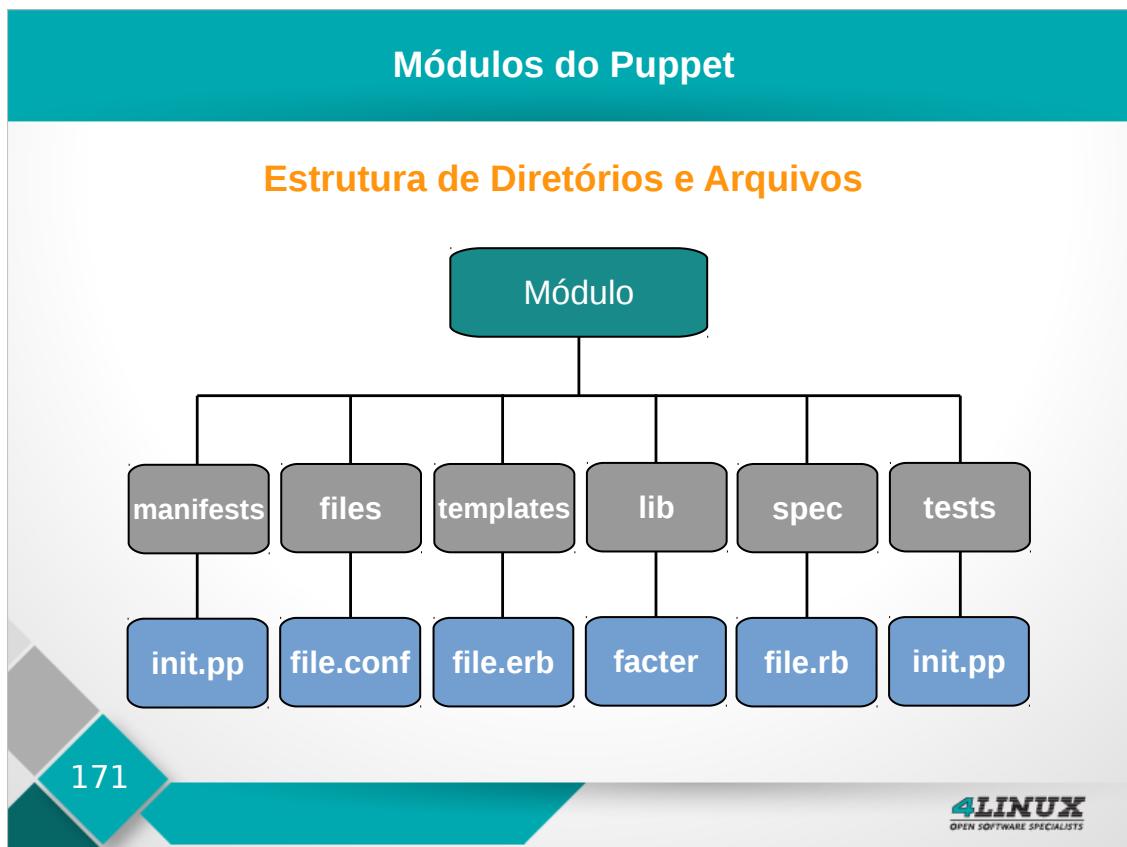
---

---

---

---

---



Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

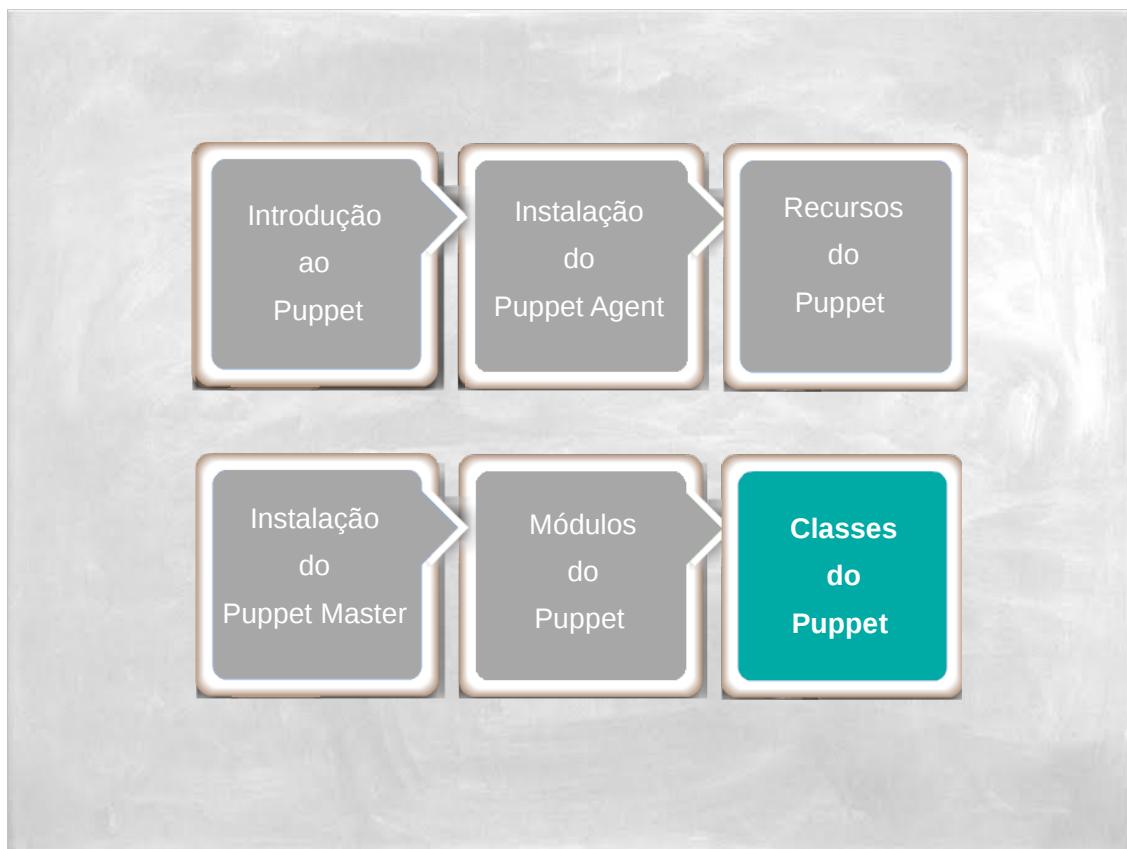
---

---

---

---

---



## Anotações:

## Classes do Puppet

**Classes** são blocos de códigos do Puppet que são armazenados nos módulos para serem reutilizadas quando necessário.



As classes do Puppet podem ser parametrizadas e podemos utilizar herança de classe deixando, assim, o nosso módulo mais robusto.

173

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

A gerência de configuração oferece um conjunto de recursos e métodos que tem o objetivo de garantir a integridade das configurações de nossos sistemas, serviços e Infraestrutura envolvida, fazendo isto de forma: **ágil, controlada e automatizada**.

## Classes do Puppet

Concluímos o Módulo WEB!  
Agora iremos utilizar o módulo  
no laboratório final da Dexter.



174



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Versionamento com GIT

### Objetivos da Aula

- Entender o GIT;
- Dar os primeiros passos com GIT;
- Trabalhar com Tags;
- Trabalhar com Branches.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

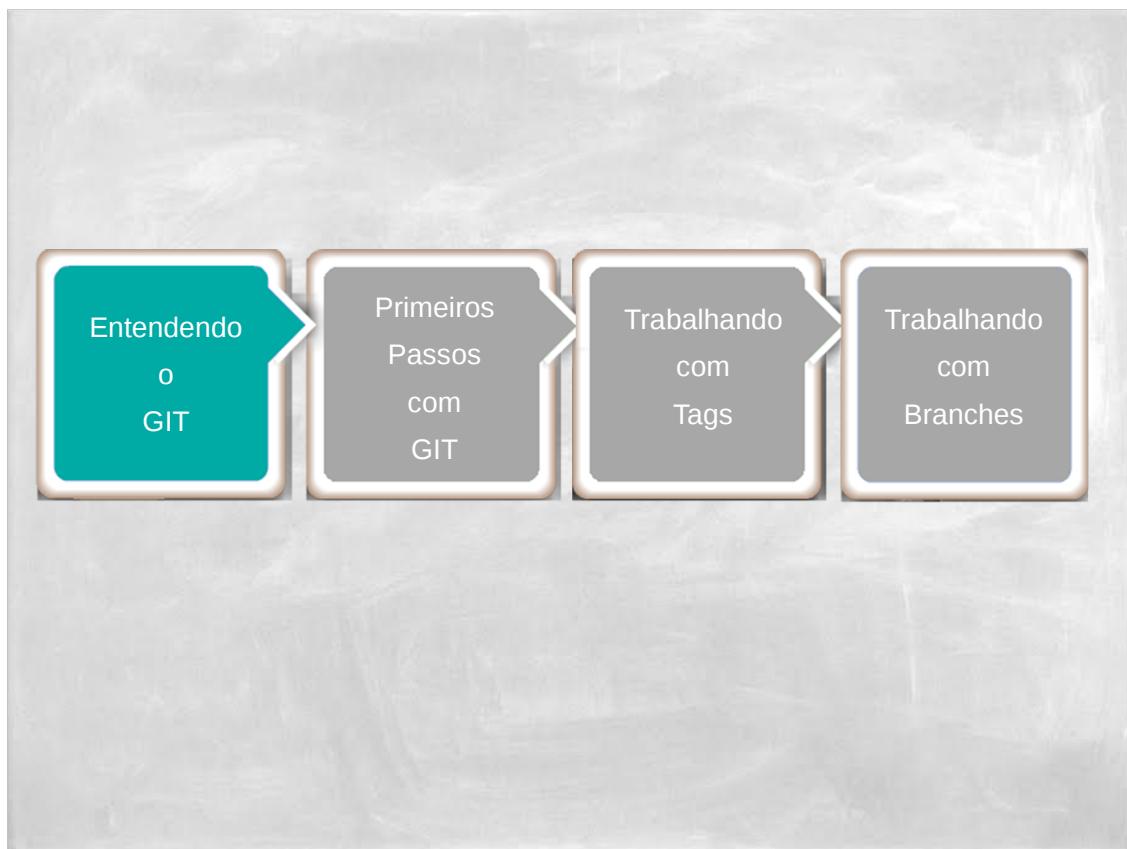
---

---

---

---

---



## Anotações:

## Introdução ao GIT

**Git** é uma ferramenta para controle de versão (vcs: version control system) desenvolvida para trabalhar de forma fácil e distribuída.



É utilizado em grandes projetos e empresas como: Linux, Google, Facebook, Twitter, Linkedin, Netflix e muitos outros.



**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

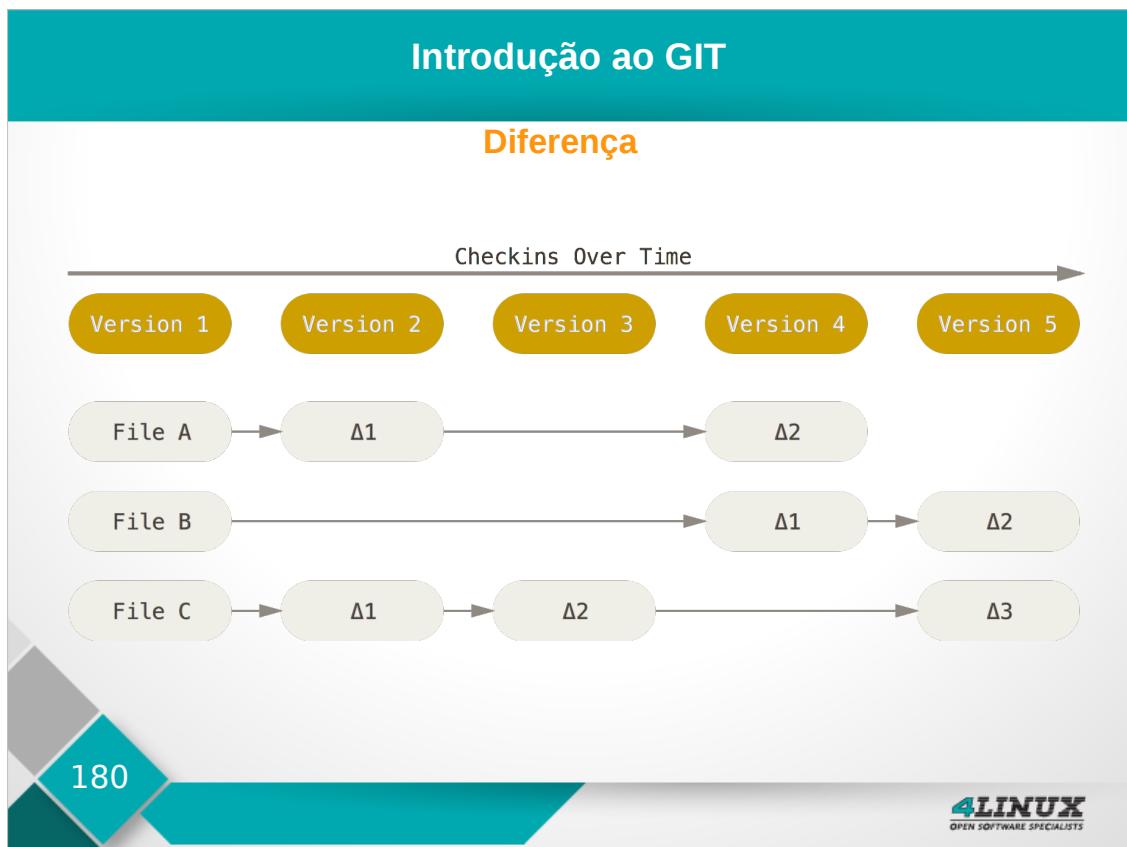
---

---

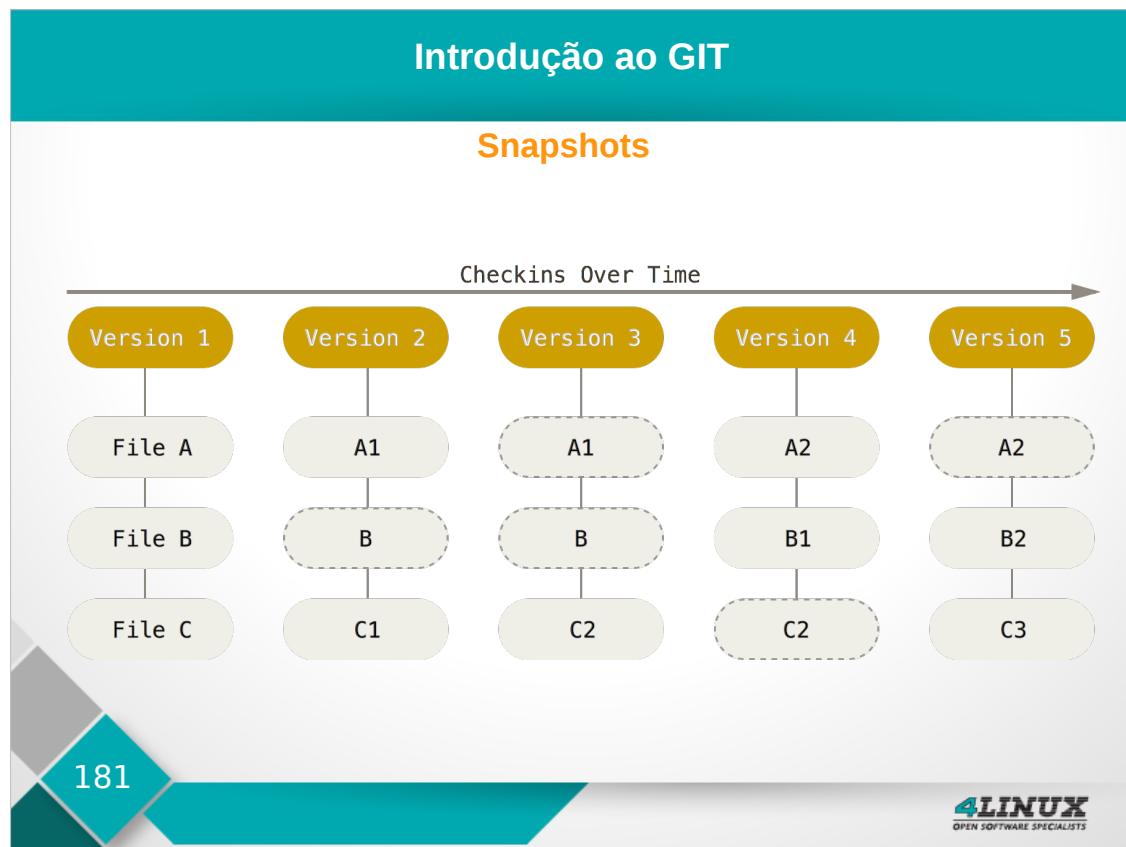
---

---

---

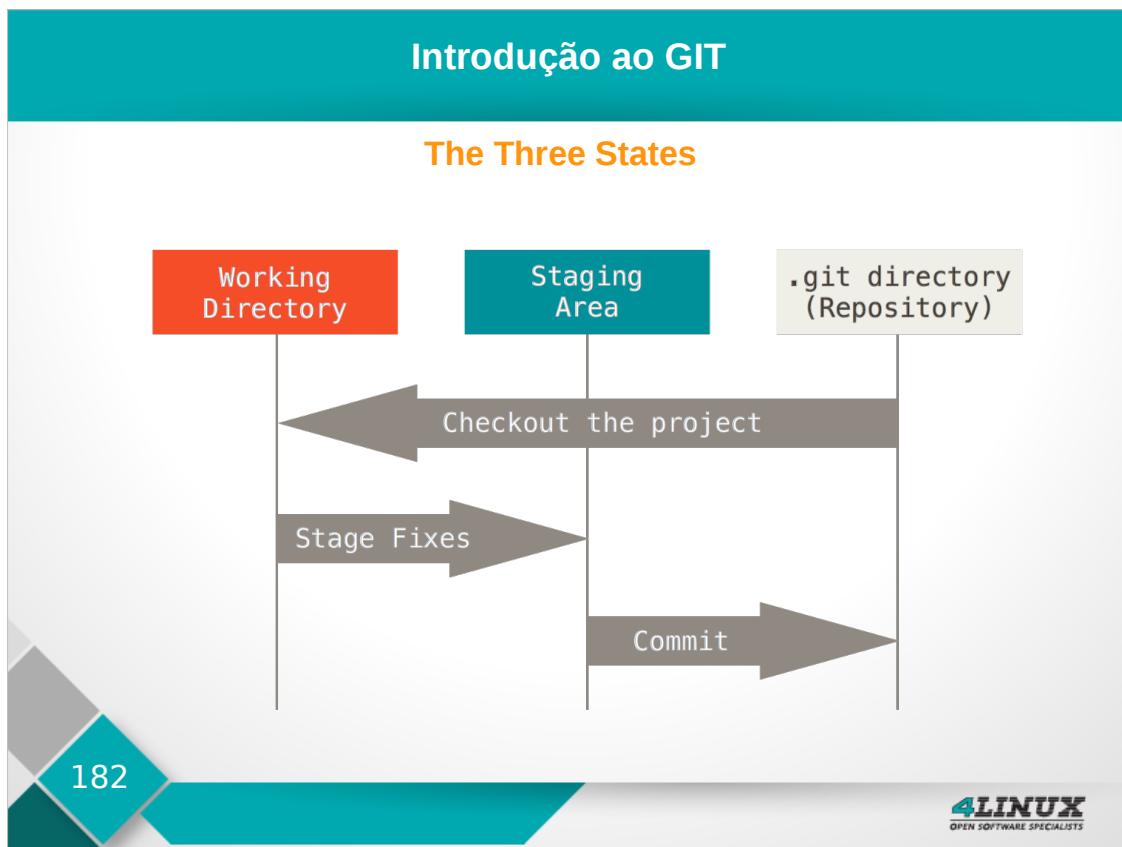


A grande diferença entre o GIT e outros VCS (Subversion) é a forma pela qual o GIT pensa sobre os dados armazenados, que basicamente são as versões dos arquivos gerenciados. A grande maioria dos sistemas de controle de versão armazena as informações em uma lista baseada em mudanças. Estes sistema (CVS, Subversion, Perforce, Bazaar) pensam nas informações como cada arquivo e suas mudanças durante o tempo de vida do projeto.



O GIT trabalha com os dados como se fossem um grupo de snapshots de um pequeno sistema de arquivo. Toda vez que o usuário realiza um commit ou salva o estado atual do projeto, ele basicamente tira uma imagem do estado de todos os arquivos e grava uma referência a esse estado (snapshot). Para ser eficiente, se o arquivo não possui nenhuma alteração, o GIT não grava os arquivos novamente, ele simplesmente cria um link para o arquivo anterior. O GIT pensa como se os arquivos e suas alterações fossem um fluxo de snapshots.

Essa arquitetura faz com que GIT trabalhe mais como um pequeno filesystem que um sistema de versionamento comum, com ótimas ferramentas construídas por cima desse “filesystem”.



Essa é uma das partes mais importantes para entender como o GIT funciona. Os arquivos do GIT podem trabalhar em três estados diferentes: committed, modified e staged.

**committed:** Estado em que o arquivo está salvo em banco de dados local, fechado em uma versão.

**modified:** Estado em que o arquivo foi modificado, mas não foi salvo dentro do banco de dados local.

**stage:** Estado em que o seu arquivo modificado está marcado para ser salvo no próximo commit (snapshot) a ser realizado.

Isso nos leva às três sessões de um repositório do GIT: Git Directory, Working Directory e Staging Area.

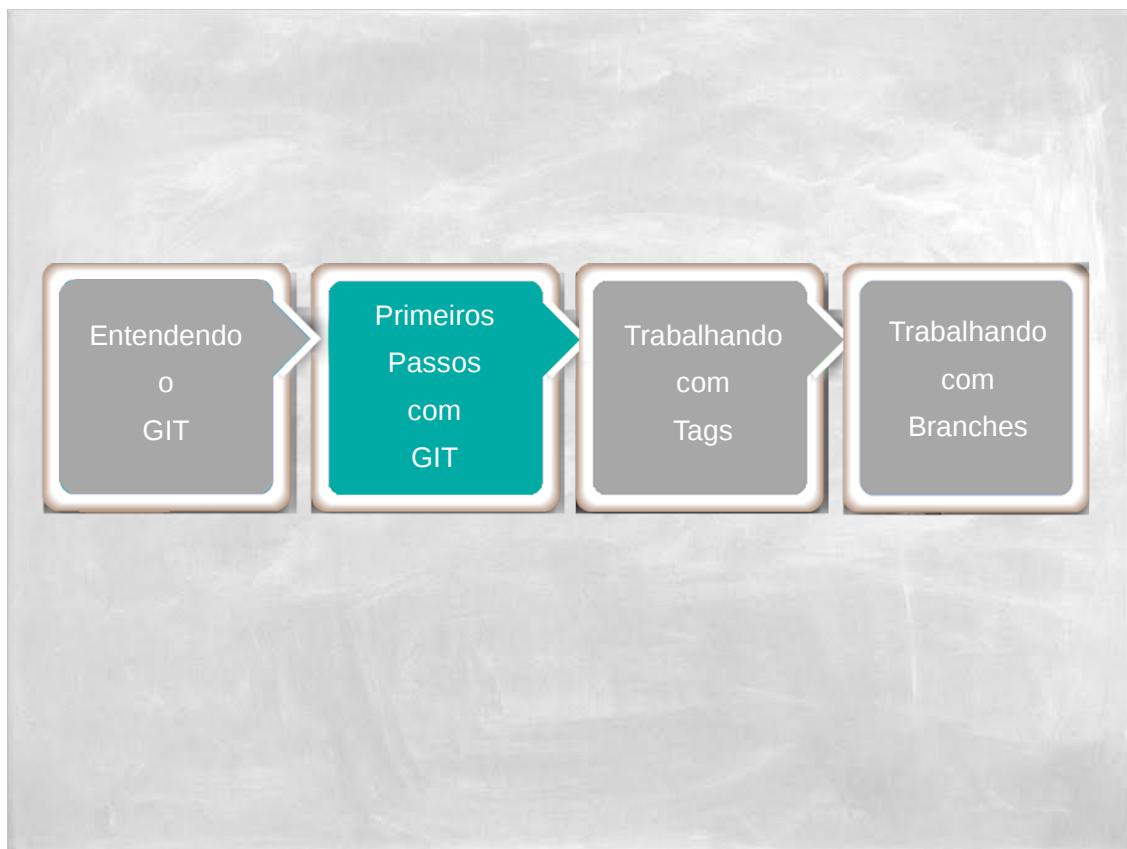
O **Git Directory** (Diretório do GIT) é onde o GIT armazena os metadados e banco de dados dos objetos do repositório. Essa é a parte mais importante do repositório.

**Working Directory** (Diretório de trabalho) é simplesmente uma versão do projeto. Esses arquivos são retirados do banco de dados do GIT e armazenados para você poder modificar e usar os arquivos.

**Staging Area** é o arquivo onde ficam armazenadas as informações do que será armazenado no próximo commit. É também conhecida como **index**.

O fluxo básico para trabalhar com GIT:

1. Você modifica os arquivos no seu diretório de trabalho (working directory).
2. Você adiciona os arquivos alterados no **index** ou na staging area.
3. Você realiza o commit, os arquivos marcados na sua área de staging são armazenados como snapshot no repositório do GIT (**Git Directory**).



## Anotações:

## Primeiros Passos com GIT

Máquina: Desktop

Agora, vamos começar a dar os nossos primeiros passos no GIT.

- 1 Instale o pacote do git.

```
1# apt-get install git
```

- 2 Crie o repositório de teste.

```
2# mkdir -p /root/aula/git/repo-teste  
3# cd /root/aula/git/repo-teste  
4# git init
```

185



**GIT init:** o comando git init inicia um repositório vazio ou reinicia um repositório já existente.

## Primeiros Passos com GIT

Máquina: Desktop

Agora, vamos criar o arquivo e salvar as informações no repositório.

- 1 Crie o arquivo de teste.

```
1# vim arquivo1  
Infraestrutura Ágil
```

- 2 Adicione o arquivo na área de Staging.

```
2# git status  
3# git add arquivo1  
4# git status
```

186



### GIT add

Esse comando atualiza a index usando o conteúdo encontrado na **Working Tree**, preparando o conteúdo para o próximo commit. Ao modificar o arquivo é necessário adicionar para **staging area**. Ao realizar o commit o conteúdo do arquivo é salvo dentro do repositório local do GIT.

Podemos usar o parâmetro **--all** para adicionar todos os arquivos no diretório atual.

### GIT status

Mostra o estado da **Working Tree**. Basicamente é usado para ver os arquivos que foram alterados. O comando é utilizado para verificar quais arquivos foram adicionados no diretório (working directory), o que foi selecionado para entrar no próximo commit (staging area).

## Primeiros Passos com GIT

Máquina: Desktop

Antes de salvar as alterações, vamos configurar as informações no GIT.

1

Configure o nome do usuário e e-mail do usuário.

```
1# git config --global user.name "Dexter Devops"  
2# git config --global user.email "devops@dexter.com.br"
```

Quando realiza-se o commit, o GIT armazena o e-mail e nome do usuário que o realizou.



187

4LINUX  
OPEN SOFTWARE SPECIALISTS

**GIT config:** o comando é responsável por definir configurações do repositório ou opções globais do GIT. Você pode buscar informações, definir, remover ou trocar informações de configurações.

## Primeiros Passos com GIT

Máquina: Desktop

Vamos salvar as mudanças realizadas no arquivo.

- 1 Crie o arquivo de teste.

```
1# git commit -m "Adcionado o arquivo arquivo1"  
2# git status
```

- 2 Verifique o log de commit do projeto.

```
1# git log  
2# git log --oneline
```

188



### GIT commit

Este comando é utilizado para armazenar mudanças do repositório local do GIT. É muito comum o parâmetro **-m** ou em sua forma estendida **-message** para definir uma mensagem simples ao commit a ser realizado. Podemos verificar que essa mensagem é mostrada quando queremos visualizar o log de commits do projeto.

### GIT log

Mostra o histórico de commit do projeto, mostrando informações como o HASH do commit, nome e e-mail do responsável, data e hora do commit e mensagem que foi armazenada no commit. Podemos utilizar **--oneline** para ocultar os metadados mostrando apenas o começo do Hash e mensagem do commit.

## Primeiros Passos com GIT

Máquina: Desktop

Vamos utilizar o revert para realizar rollback das configurações.

1

Altere o arquivo e salve as alterações no GIT.

```
1# vim arquivo1
```

LINHA DE CONFIGURAÇÃO COM ERROR

2

Salve as alterações realizadas.

```
2# git add arquivo1
```

```
3# git commit -m "Teste de ERROR"
```

189



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Primeiros Passos com GIT

Máquina: Desktop

Vamos utilizar o revert para realizar rollback das configurações.

- 3 Reverta o commit realizado.

```
1# git log --oneline  
2# git revert <id_commit>
```

- 4 Verifique se o arquivo voltou no seu estado anterior.

```
3# cat arquivo1
```

190



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

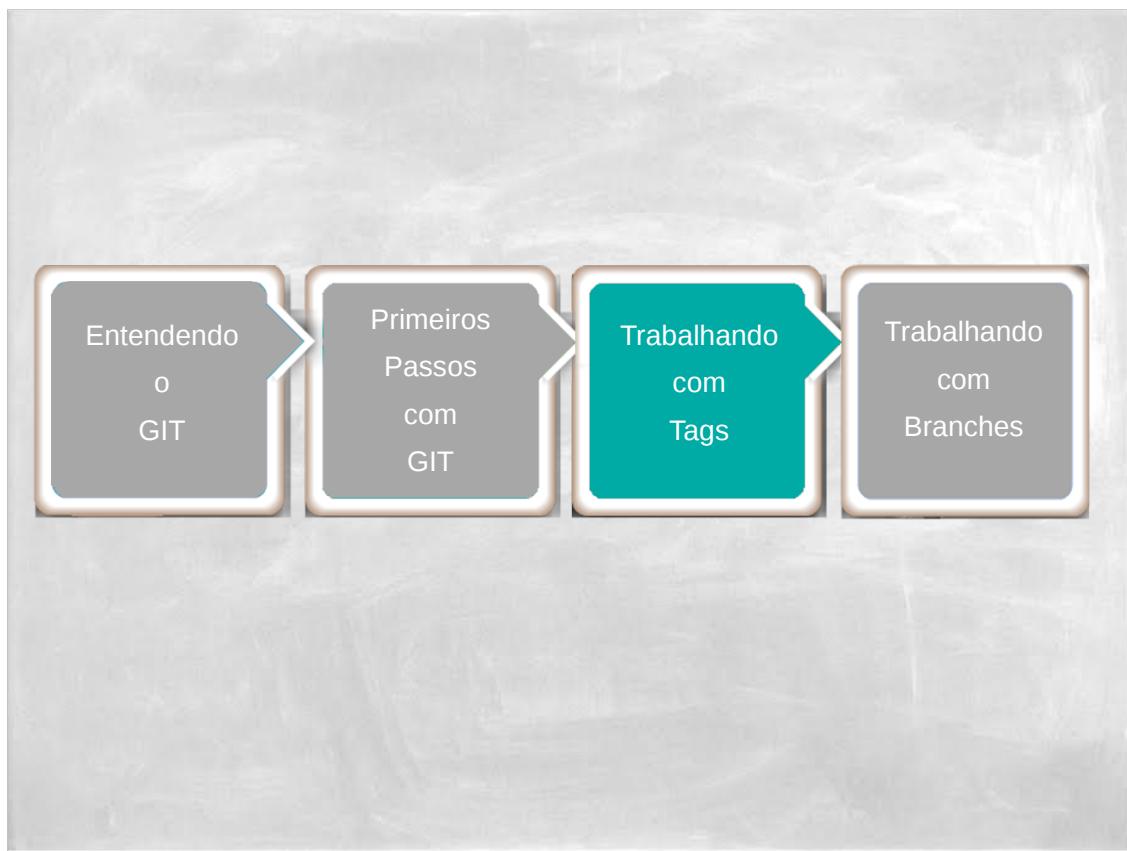
---

---

---

---

---



## Anotações:

## Trabalhando com Tags

Através das **Tags** há possibilidades de marcar pontos importantes durante o projeto. Normalmente as pessoas utilizam esta funcionalidade para marcar versões de release (v1.0, v1.1, v2.0).



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Trabalhando com Tags

Máquina: Desktop

- 1 Liste as tags presentes no repositório.

```
1# git tag
```

- 2 Modifique o arquivo.

```
2# echo "Infraestrutura como código" >>  
arquivol
```

```
3# git add arquivol
```

- 3 Crie uma tag no projeto com o nome v1.0.

```
4# git tag -a "v1.1" -m "novo conteúdo adicionado"  
5# git tag
```

193



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Trabalhando com Tags

Máquina: Desktop

Podemos utilizar tags em commit já realizados anteriormente.

- 1 Identifique o ID do commit.

```
1# git log --oneline
```

- 2 Crie uma tag para o primeiro commit do projeto.

```
2# git tag -a v1.2 <id_commit>
```

194



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

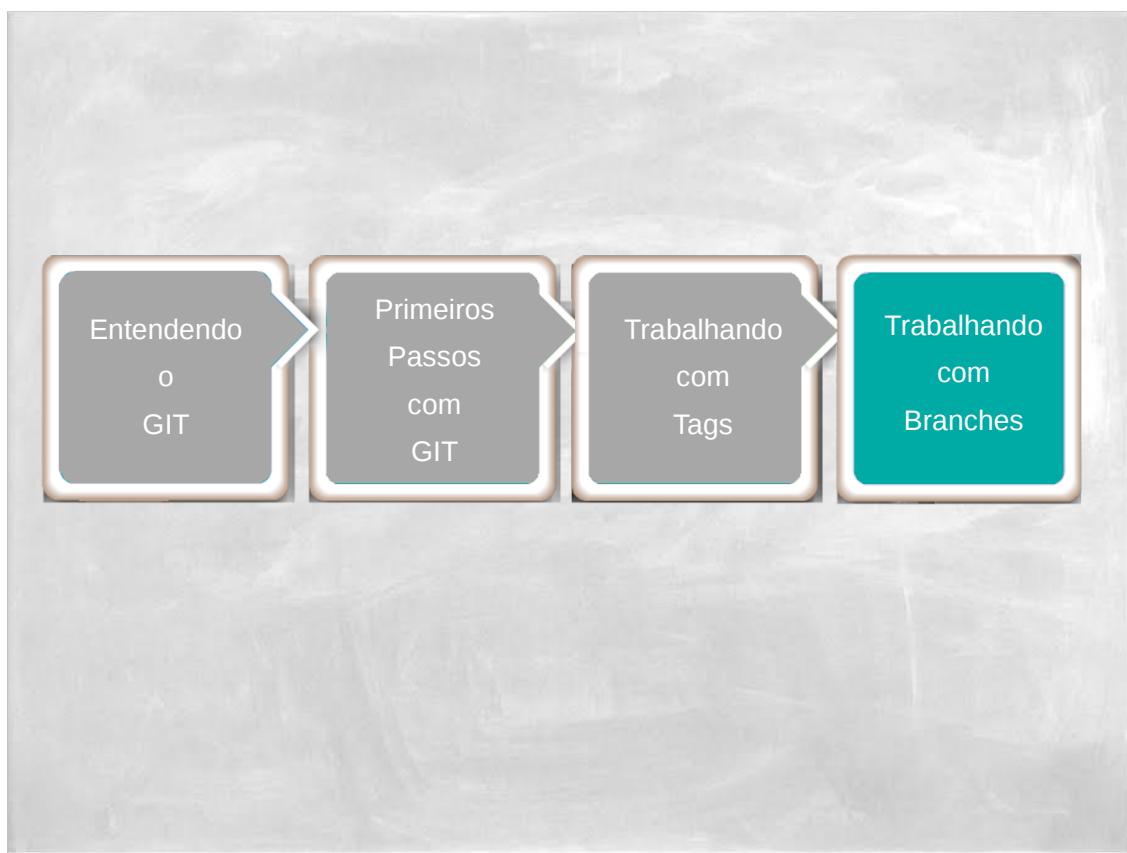
---

---

---

---

---



## Anotações:

## Trabalhando com Branches

**Branches** (“ramos”) são utilizados para fazer uma ramificação do projeto a partir da sua linha principal.



O **Branch padrão** do repositório é a **Branch Master**.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

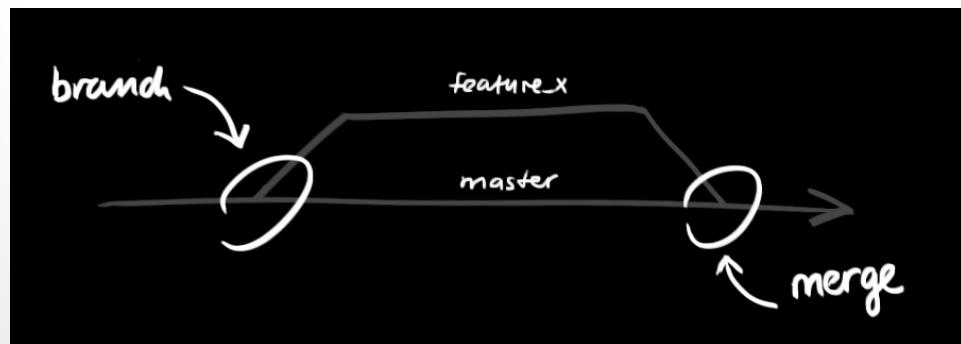
---

---

---

## Trabalhando com Branches

Os **Branches** podem ser mesclados (merge) uns com os outros para unificar o projeto novamente.



197

4LINUX  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Primeiros Passos com GIT

Máquina: Desktop

Vamos criar um branch testing.

- 1 Crie um branch de nome testing.

```
1# git checkout -b testing
```

- 2 Liste os branches do projeto.

```
2# git branch
```

- 3 Salve as mudanças no novo branch.

```
3# echo "Movimento Ágil" >> arquivo2  
4# git add --all  
5# git commit -m "Novo arquivo adicionado"
```

198



### GIT checkout

O comando checkout é utilizado para você navegar entre as branches do seu projeto. Podemos usar o parâmetro **-b** para criar a branch caso ela não exista e transferir os arquivos atuais para a branch.

### GIT branch

O comando branch é usado para gerenciar as branches do projeto. Você pode criar, listar ou deletar branches do projeto.

## Primeiros Passos com GIT

Máquina: Desktop

Vamos mesclar o branch de testing com o master.

- 1 Visualize as diferenças entre os branches.

```
1# git checkout master
```

```
2# ls
```

- 2 Visualize as diferenças entre os branches.

```
3# git diff master testing
```

- 3 Realize o merge entre os branches.

```
4# git merge testing
```

```
5# ls
```



### GIT diff

Este comando é utilizado para verificar diferenças entre arquivos, commits, entre commit e working tree, etc.

### GIT merge

O comando é utilizado para mesclar dois objetos no GIT.

## Trabalhando com GITHUB

**Github** é uma aplicação web para armazenamento e gerenciamento de projetos do GIT. Disponibiliza uma interface gráfica, realiza controle de acesso, wikis, entre outros serviços.



**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Gerenciando Repositórios com Gitlab

### Objetivos da Aula

- Fazer uma introdução ao Gitlab;
- Instalar o Gitlab;
- Gerenciar usuários;
- Gerenciar repositórios;
- Gerenciar repositórios remotos com Git.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

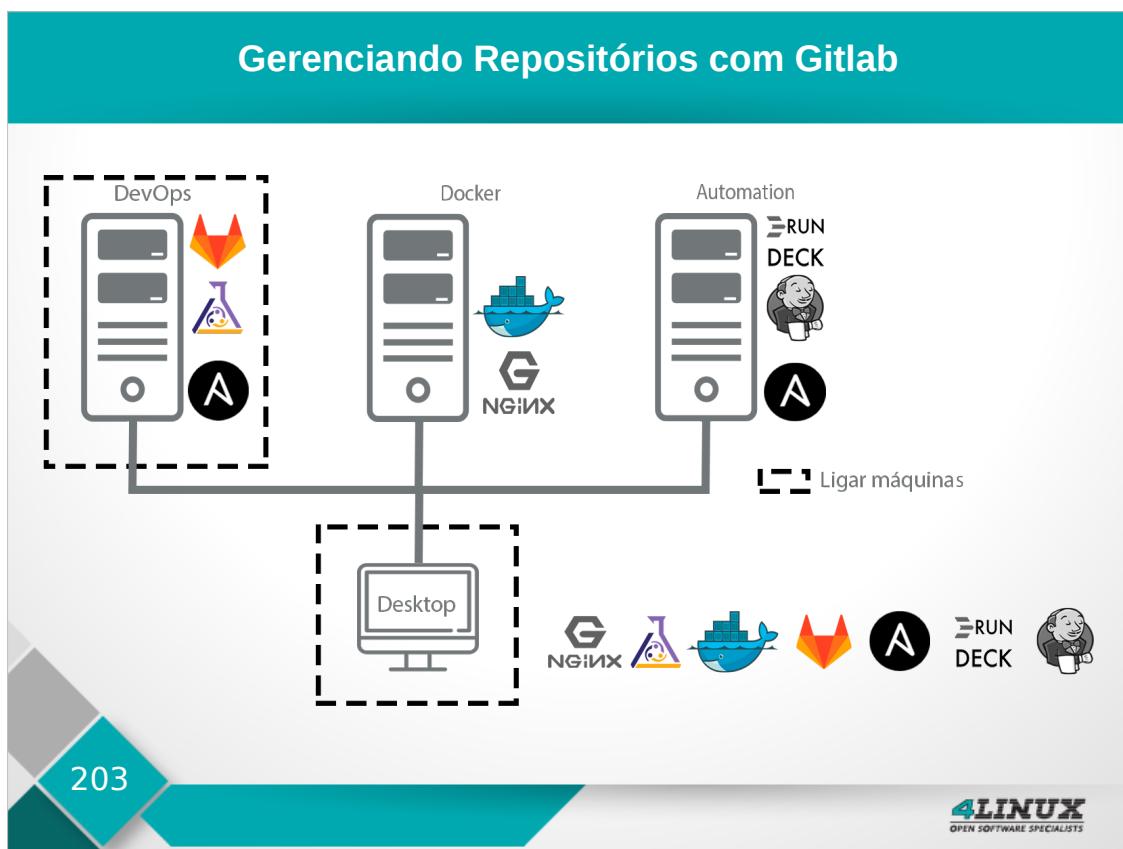
---

---

---

---

---



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

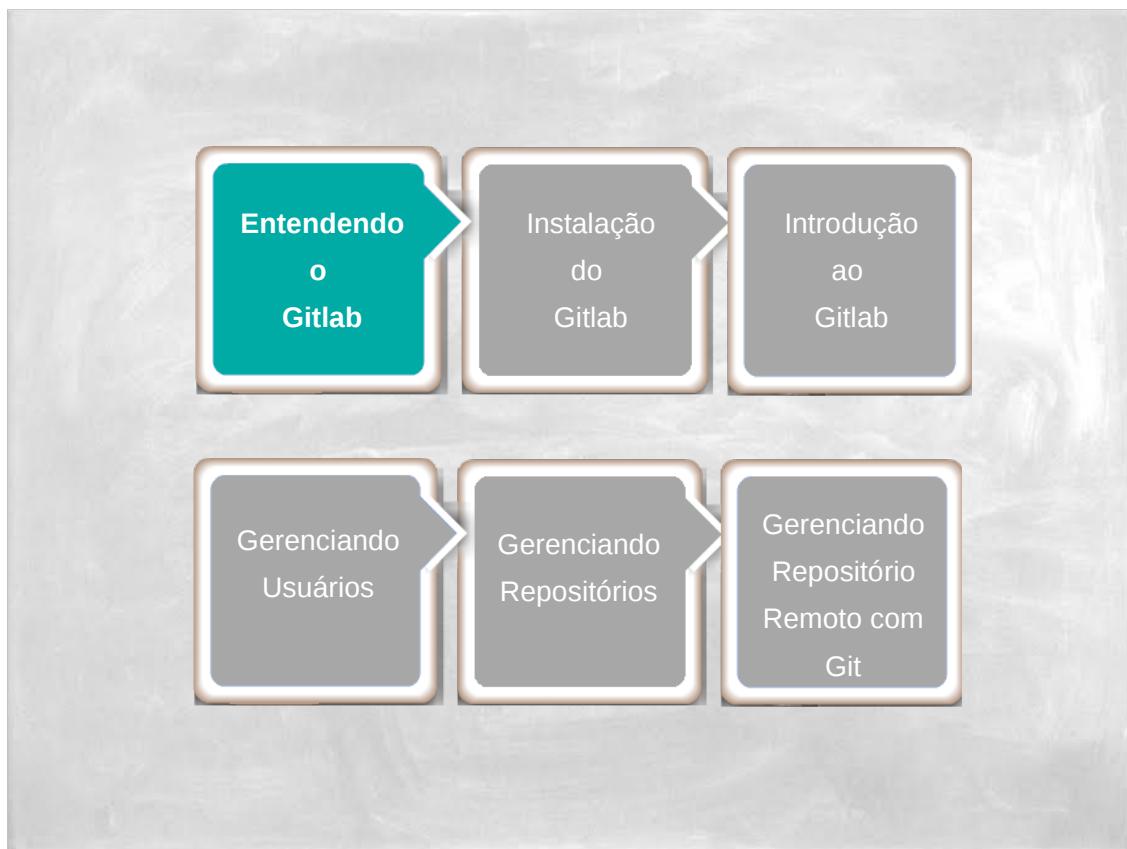
---

---

---

---

---



## Anotações:

## Introdução ao Gitlab

**Gitlab** é uma aplicação web para gerenciar repositórios privados.

Possui a mesma função do Github, porém você pode instalar em sua infraestrutura e gerenciar o serviço internamente.



205

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

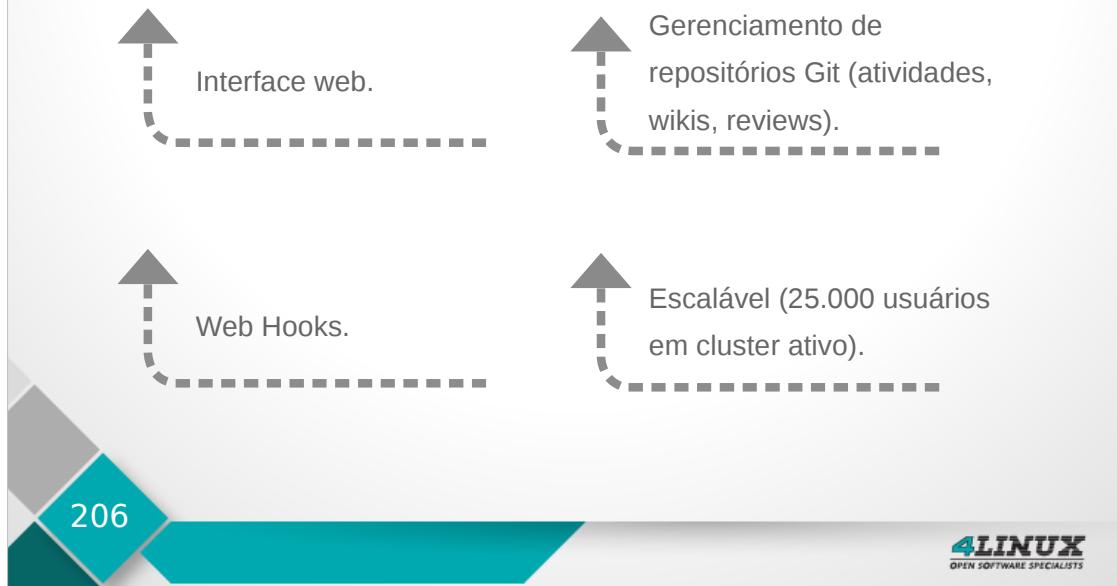
---

---

---

## Introdução ao Gitlab

### Características do Gitlab



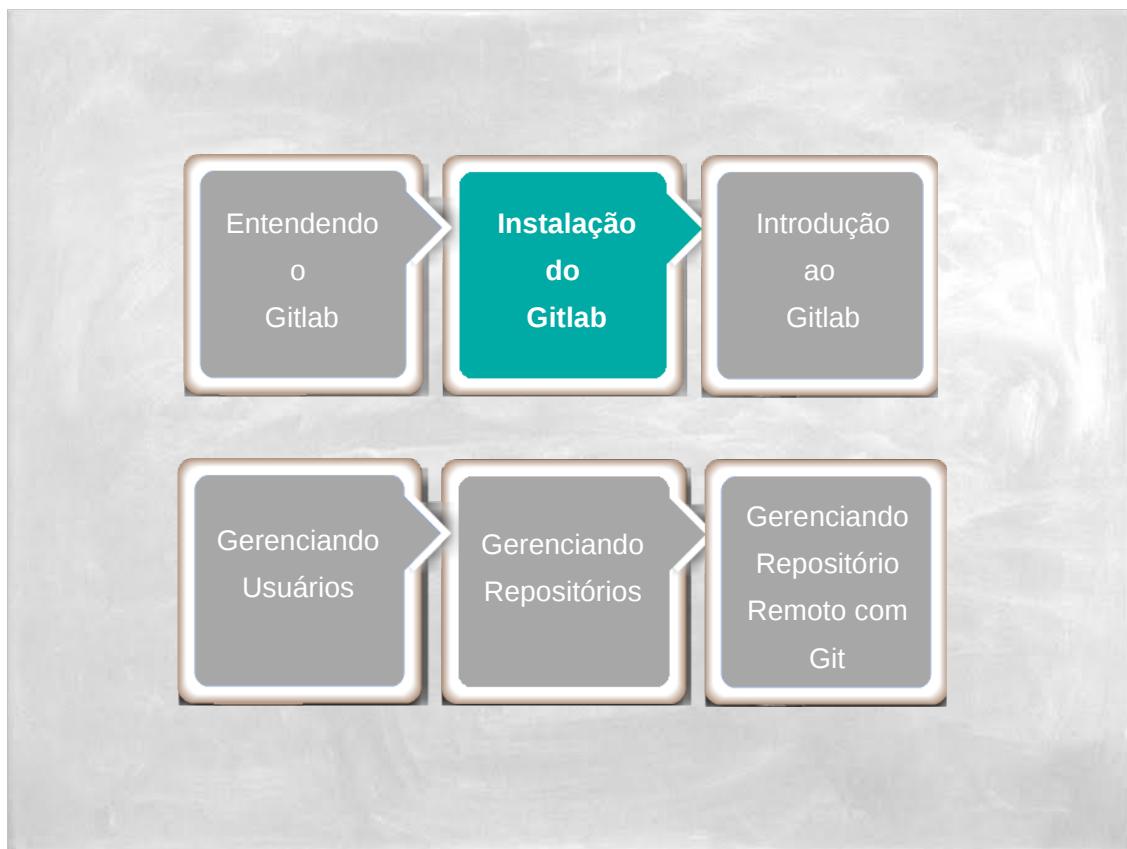
### Continuous Integration

**Continuous Integration** ou **Integração Contínua** é uma prática de desenvolvimento em que os membros de um time integram seu trabalho frequentemente, normalmente essa integração é realizada diariamente. Cada integração é verificada por uma tarefa ou build automatizada para detectar erros de integração da forma mais rápida possível. A integração contínua diminui a probabilidade de erros dentro de um projeto.

### Continuous Delivery

**Continuous Delivery** ou **Entrega Contínua** é uma prática de desenvolvimento usada na construção de software, de uma maneira que você possa disponibilizar mudanças em seu ambiente de produção a qualquer momento.

Em um ambiente de entrega contínua são realizadas mudanças pequenas e com maior frequência de Deploy, o que diminui o risco de erros em ambientes de produção, simplesmente pelo fato de serem pequenas mudanças.



## Anotações:

## Instalação do Gitlab

Máquina: Devops

Agora, vamos executar o playbook do ansible para instalação do GitLab.

1

Abra o arquivo `/etc/ansible/roles/gitlab/tasks/main.yml` para que possamos conferir os passos de instalação do GitLab.

```
1# vim /etc/ansible/roles/gitlab/tasks/main.yml
```

2

Para executar o playbook, podemos usar o comando:

```
1# ansible-playbook /etc/ansible/playbooks/ambiente/devops.yml
```

208



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Instalação do Gitlab

Máquina: Devops

Vamos configurar o Gitlab.

- 1 Modifique o arquivo de configuração.

```
1# vim /etc/gitlab/gitlab.rb  
external_url 'http://devops.dexter.com.br'
```

- 2 Reconfigure e restart o Gitlab.

```
2# gitlab-ctl reconfigure  
3# gitlab-ctl restart
```



209



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

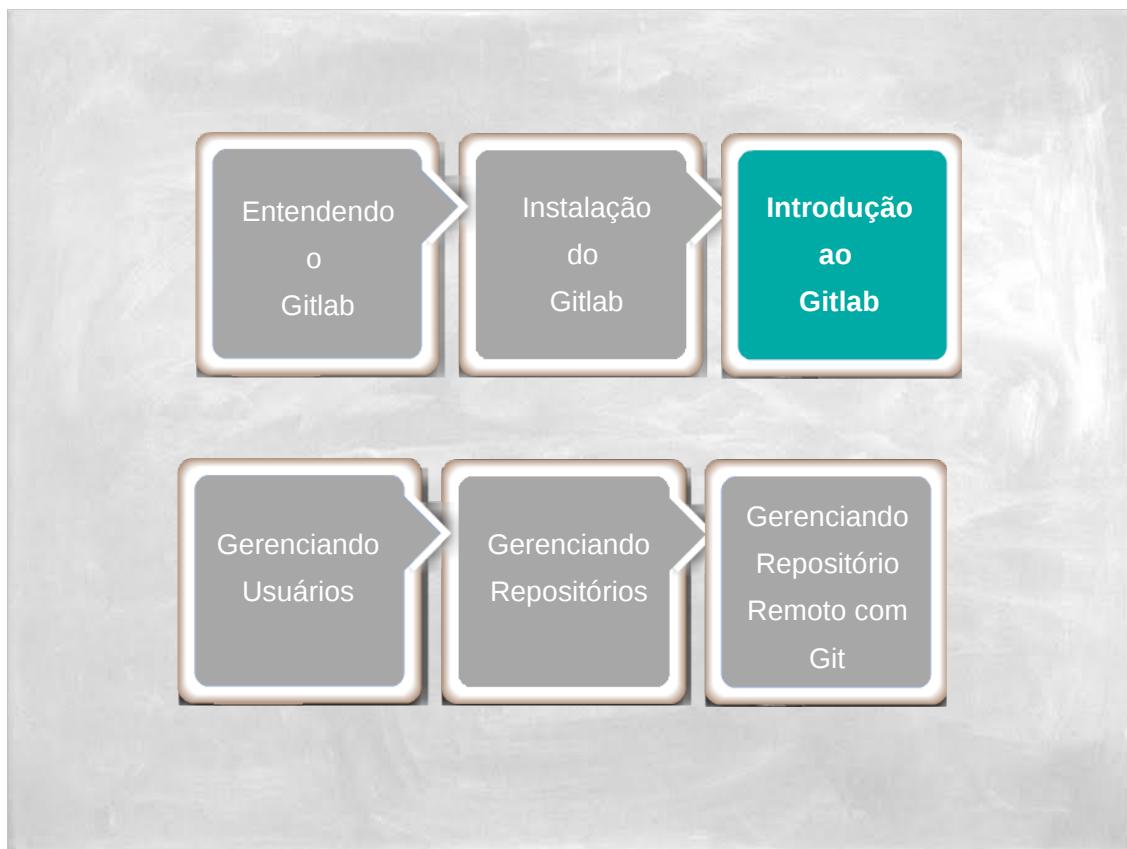
---

---

---

---

---



## Anotações:

## Introdução à Interface

Na tela inicial do Gitlab, ele pedirá para definir a nova senha do usuário root. Definiremos como “dexter2017”.

Please create a password for your new account.

### GitLab Community Edition

**Open source software to collaborate on code**

Manage git repositories with fine grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

Explore Help About GitLab

211

Change your password

New password

Confirm new password

Change your password

Didn't receive confirmation instructions?

Already have login and password? [Sign in](#)

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

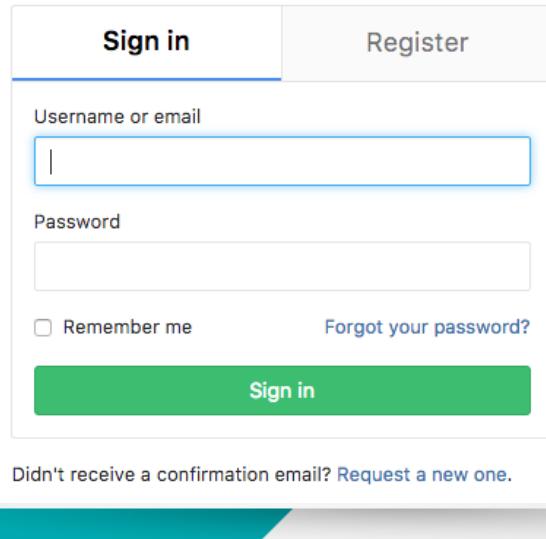
---

---

## Introdução à Interface

Após definir a senha, vamos logar no Gitlab.

Usuário: **root** e Senha: **dexter2017**



The image shows a screenshot of a Gitlab sign-in page. At the top, there are two tabs: "Sign in" (which is active) and "Register". Below the tabs, there are two input fields: "Username or email" and "Password". Underneath the password field is a "Remember me" checkbox and a "Forgot your password?" link. At the bottom of the form is a large green "Sign in" button. Below the form, a message says "Didn't receive a confirmation email? Request a new one." A small watermark "212" is visible in the bottom left corner of the slide.



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

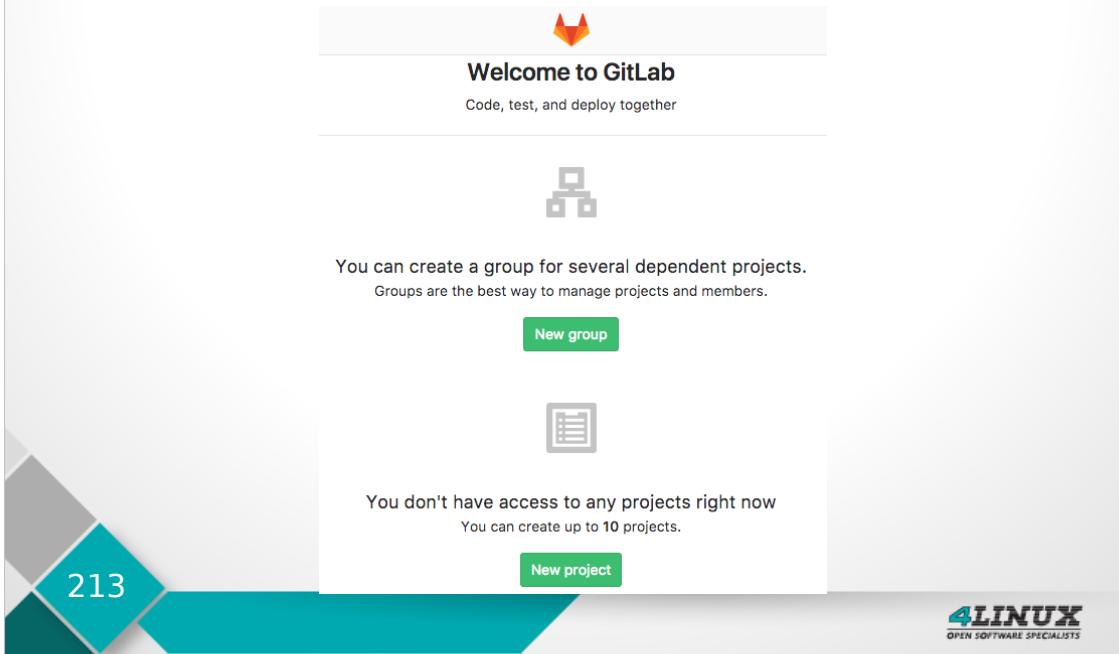
---

---

---

## Introdução à Interface

Menu principal do Gitlab:



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

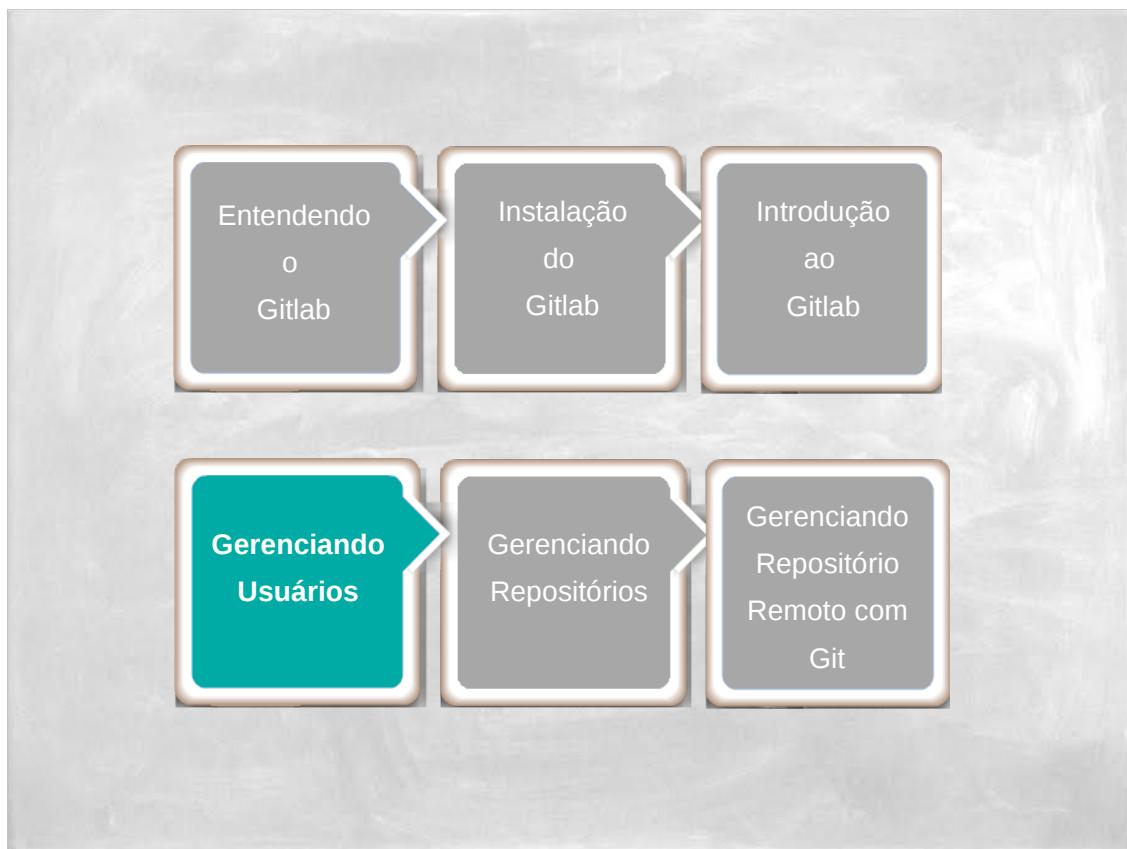
---

---

---

---

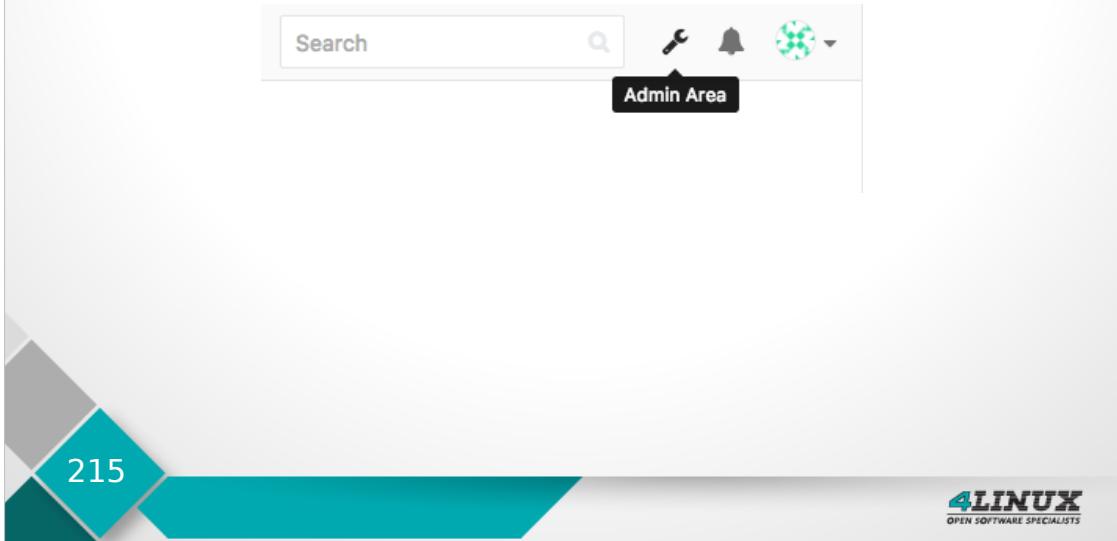
---



## Anotações:

## Gerenciando Usuários

Vamos acessar a área de administração do Gitlab para adicionar um novo usuário.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

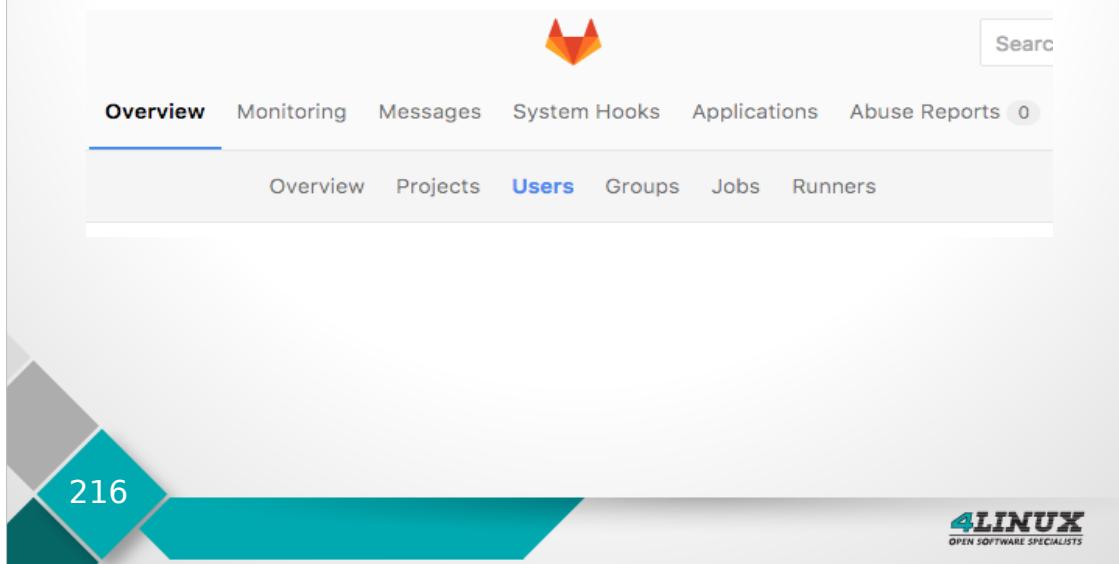
---

---

## Gerenciando Usuários

Acesse o menu **users** para criação do usuário **Devops**.

Vamos utilizar o usuário no Projeto Dexter.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Gerenciando Usuários

Clique no botão **New User**.

217

4LINUX  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Gerenciando Usuários

Insira os dados do usuário:

**Name:** Devops

**Username:** devops

**Email:** Devops@dexter.com.br

Admin Area	
Account	
Name	Devops * required
Username	devops * required
Email	devops@dexter.com.br * required

218



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Gerenciando Usuários

Insira os dados sobre o acesso da conta:

**Projects limit:** 100

Marque a opção do usuário Admin.

Para finalizar a criação do usuário, clique no botão **Create User** no final da página.

**Access**

Projects limit	100
Can create group	<input checked="" type="checkbox"/>
Admin	<input checked="" type="checkbox"/> 



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

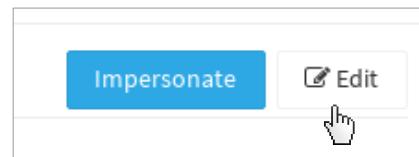
---

---

## Gerenciando Usuários

Agora, vamos definir uma senha para o novo usuário criado.

Dentro do perfil do usuário Devops, clique no botão: **Edit**.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Gerenciando Usuários

Defina a senha do usuário como: **Devops@4linux**

Password	
Password	<input type="password"/> 
Password confirmation	<input type="password"/> 

Agora, faça o logout do sistema e entre novamente com o usuário Devops.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

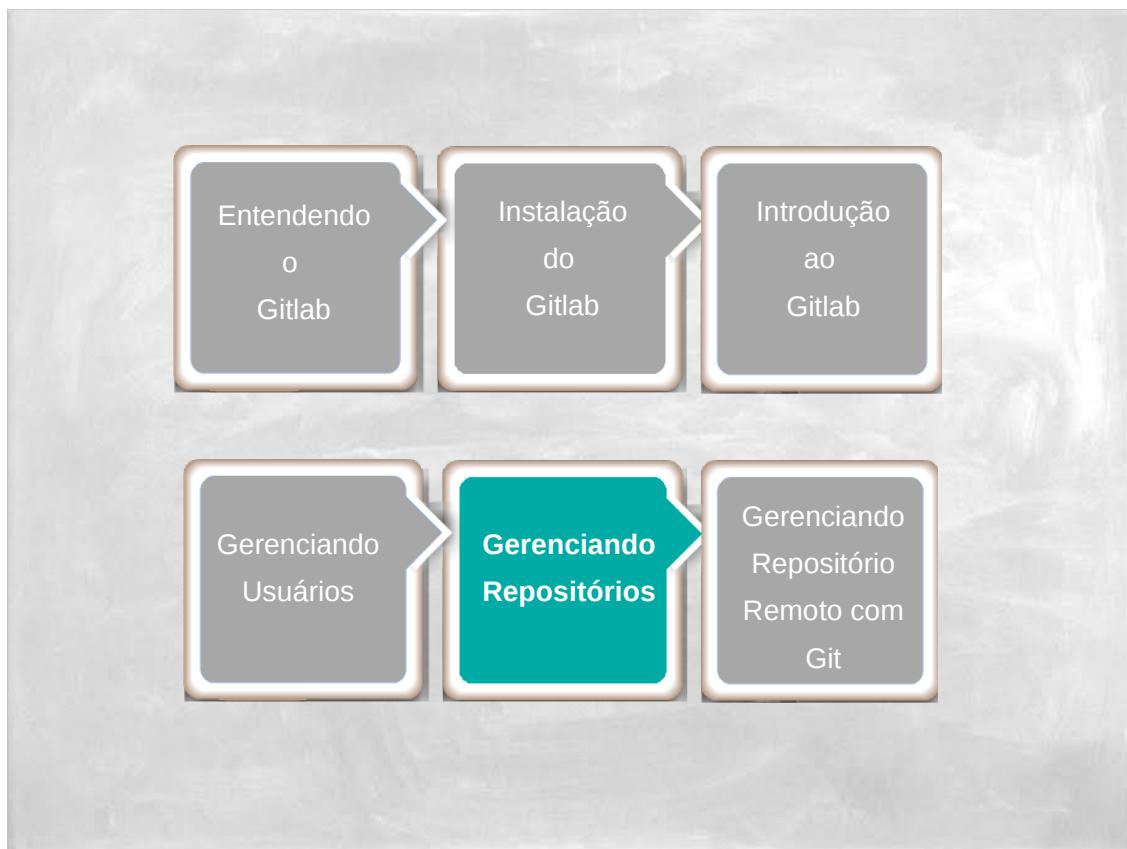
---

---

---

---

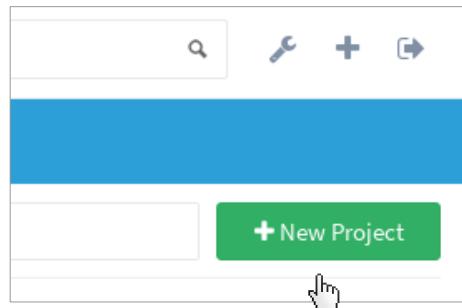
---



## Anotações:

## Gerenciando Repositórios

Clique no botão **New Project** para criar um novo projeto.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

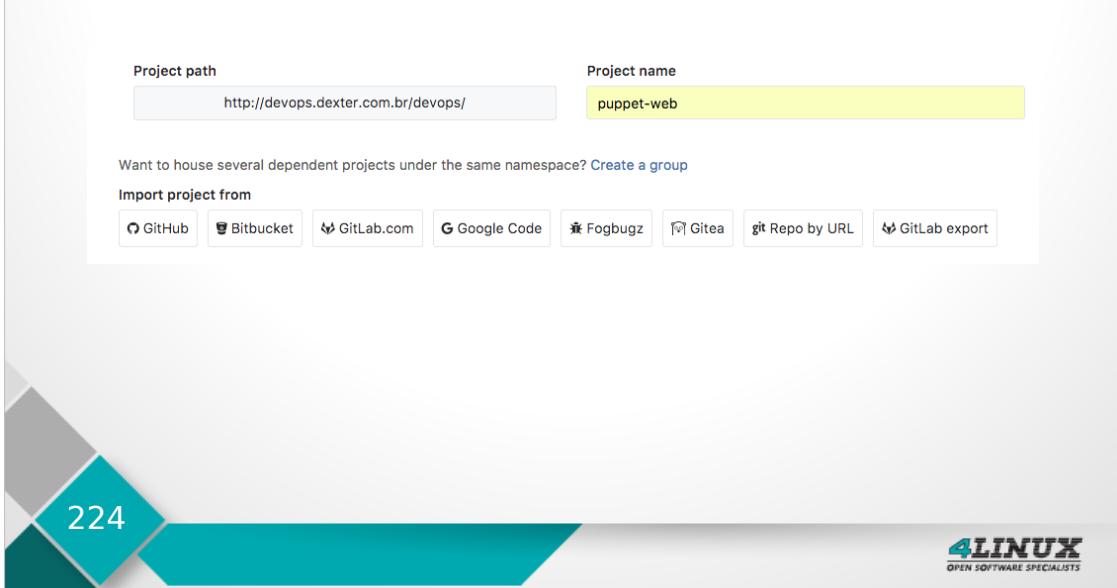
---

---

---

## Gerenciando Repositórios

Insira o nome do projeto: **puppet-web**. O projeto do web será o primeiro módulo a ser utilizado no Projeto Dexter.



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Gerenciando Repositórios

Insira uma descrição simples para o repositório e clique no botão **Create Project** para finalizar a criação do Projeto.

Project description (optional)  
Modulo responsável por gerenciar as configurações de servidor web da Dexter.

Visibility Level (?)  
Visibility Level ⓘ  
 Private  
Project access must be granted explicitly to each user.  
 Internal  
The project can be cloned by any logged in user.  
 Public  
The project can be cloned without any authentication.

**Create project**



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

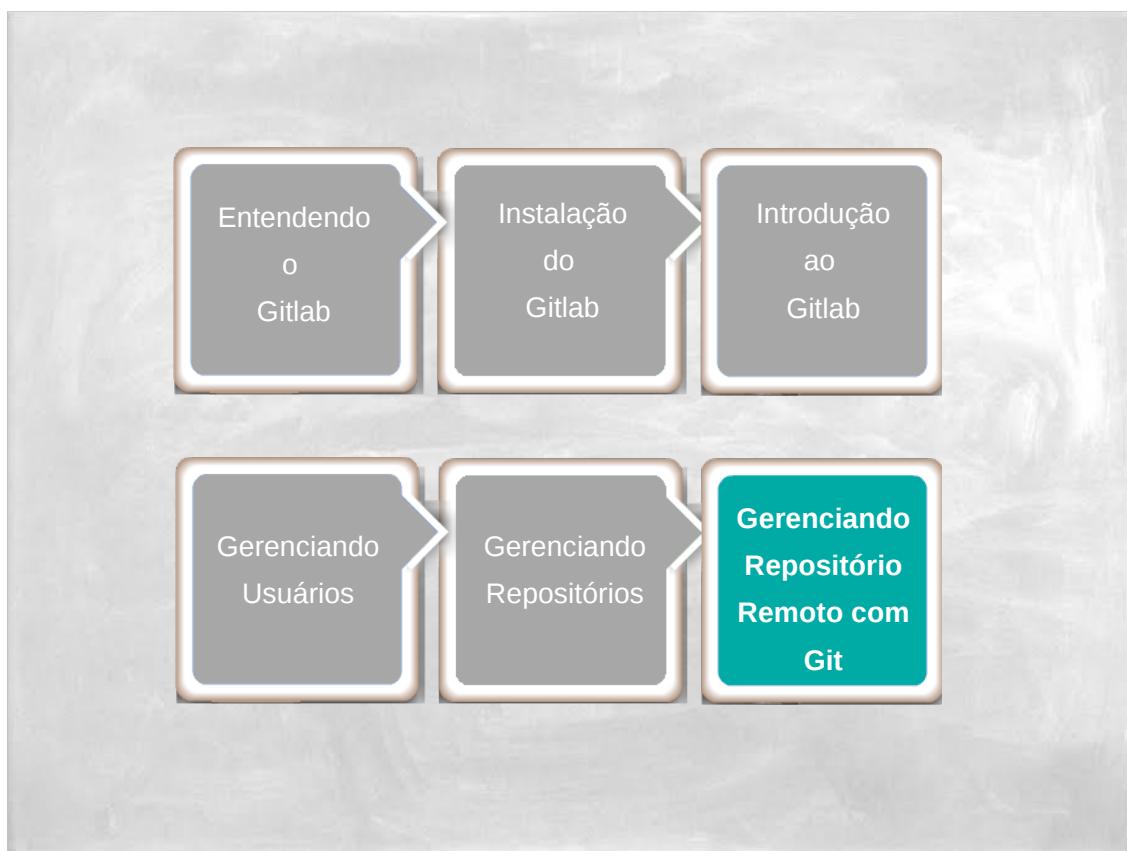
---

---

---

---

---



## Anotações:

## Repositório Remoto com Git

Máquina: Devops

Vamos configurar o módulo de DNS dentro do Gitlab.

1

Vamos configurar o repositório.

```
1# cd /etc/puppet/modules/puppet-web/
```

2

Inicie o repositório dentro do diretório.

```
2# git init
```

3

Adicione o repositório criado anteriormente.

(Execute na mesma linha!)

```
3# git remote add origin
```

```
http://devops.dexter.com.br/devops/puppet-web.git
```

227



### Git remote:

O comando **remote** é utilizado para gerenciar repositórios remotos, sendo eles o Github, Gitlab, BitBucket, entre outros.

#### Subcomandos do **remote**:

- add: Adiciona um novo repositório remoto ao projeto.

## Repositório Remoto com Git

Máquina: Devops

4

Adicione todos os arquivos que serão enviados para o gitlab.

```
1# git add --all
```

5

Faça um commit, enviando o comentário “subindo o projeto”.

```
2# git commit -m "subindo o projeto"
```

6

Envie o código para o repositório remoto.

```
3# git push origin master
```

228



Subcomandos do **remote**:

- show: lista as informações dos repositórios. Podemos passar o nome do repositório para verificar mais informações.

### Git Push

O comando **push** é utilizado para atualizar informações em um repositório remoto. Ao executar o comando, é necessário passar dois parâmetros, o nome do repositório e o nome da branch a ser atualizada.

## Repositório Remoto com Git

Concluímos o repositório WEB.  
Iremos utilizar o módulo no  
laboratório final da **Dexter**.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Introdução ao Rundeck

### Objetivos da Aula

- Fazer uma Introdução ao Rundeck;
- Dar os Primeiros Passos com Rundeck;
- Criar Projetos, Grupos e Jobs com o Rudeck;
- Automatizar o Deploy com Ansible e Rundeck;
- Integrar o Jenkins ao Rundeck.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

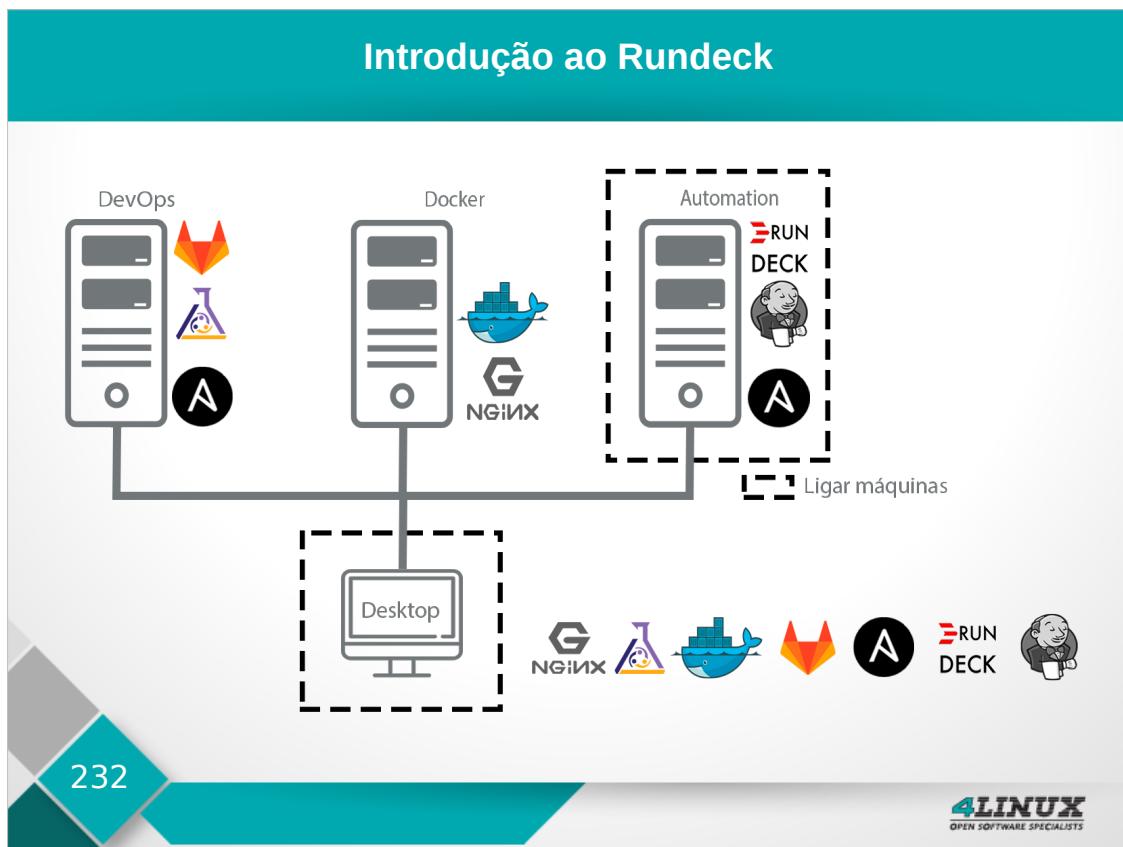
---

---

---

---

---



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

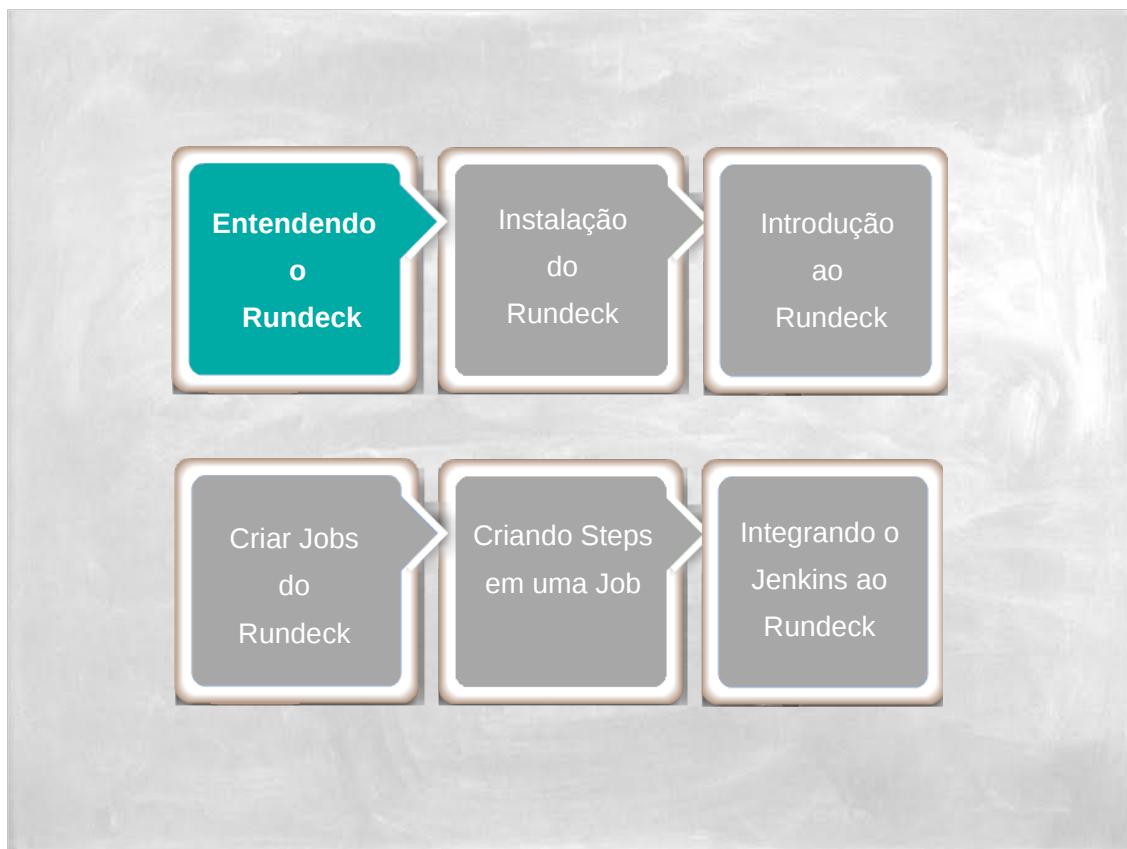
---

---

---

---

---

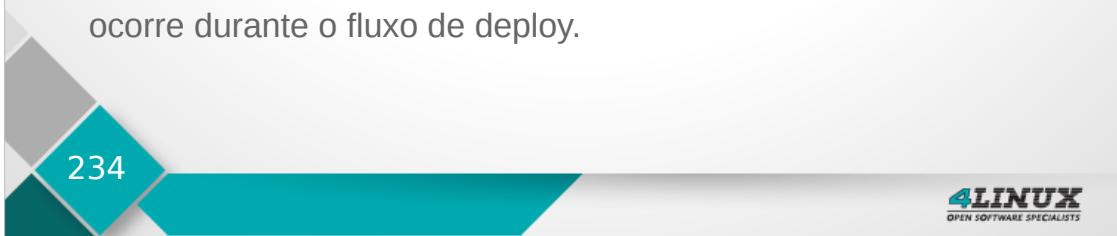


## Anotações:

## Entendendo o Rundeck

O **Rundeck** é a aplicação mais utilizada como **Continuous Delivery (CD)**.

Rundeck é um agendador de tarefas open source, que pode executar comandos “multi-nó” e sempre mantém registro em log de tudo que ocorre durante o fluxo de deploy.



### Continuous Integration

**Continuous Integration** ou **Integração Contínua** é uma prática de desenvolvimento em que os membros de um time integram seu trabalho frequentemente. Normalmente essa integração é realizada diariamente. Cada integração é verificada por uma tarefa ou build automatizada para detectar erros de integração da forma mais rápida possível. A integração contínua diminui a probabilidade de erros dentro de um projeto.

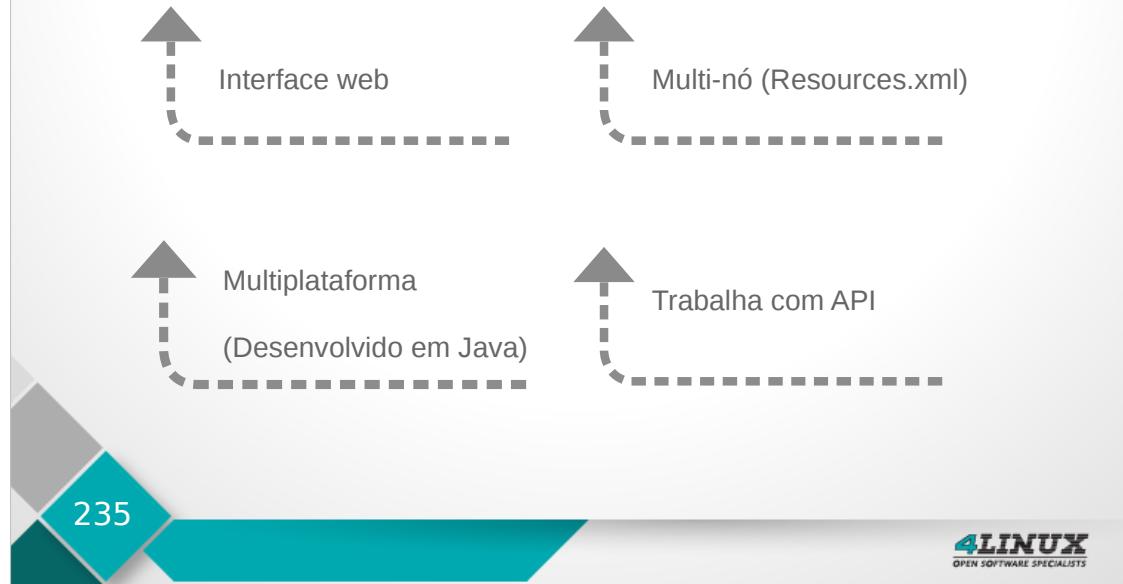
### Continuous Delivery

**Continuous Delivery** ou **Entrega Contínua** é uma prática de desenvolvimento usada na construção de software, de uma maneira que você possa disponibilizar mudanças em seu ambiente de produção a qualquer momento.

Em um ambiente de entrega contínua, são realizadas mudanças pequenas e com maior frequência de Deploy, o que diminui o risco de erros em ambientes de produção, simplesmente pelo fato de serem pequenas mudanças.

## Entendendo o Rundeck

### Características do Rundeck



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

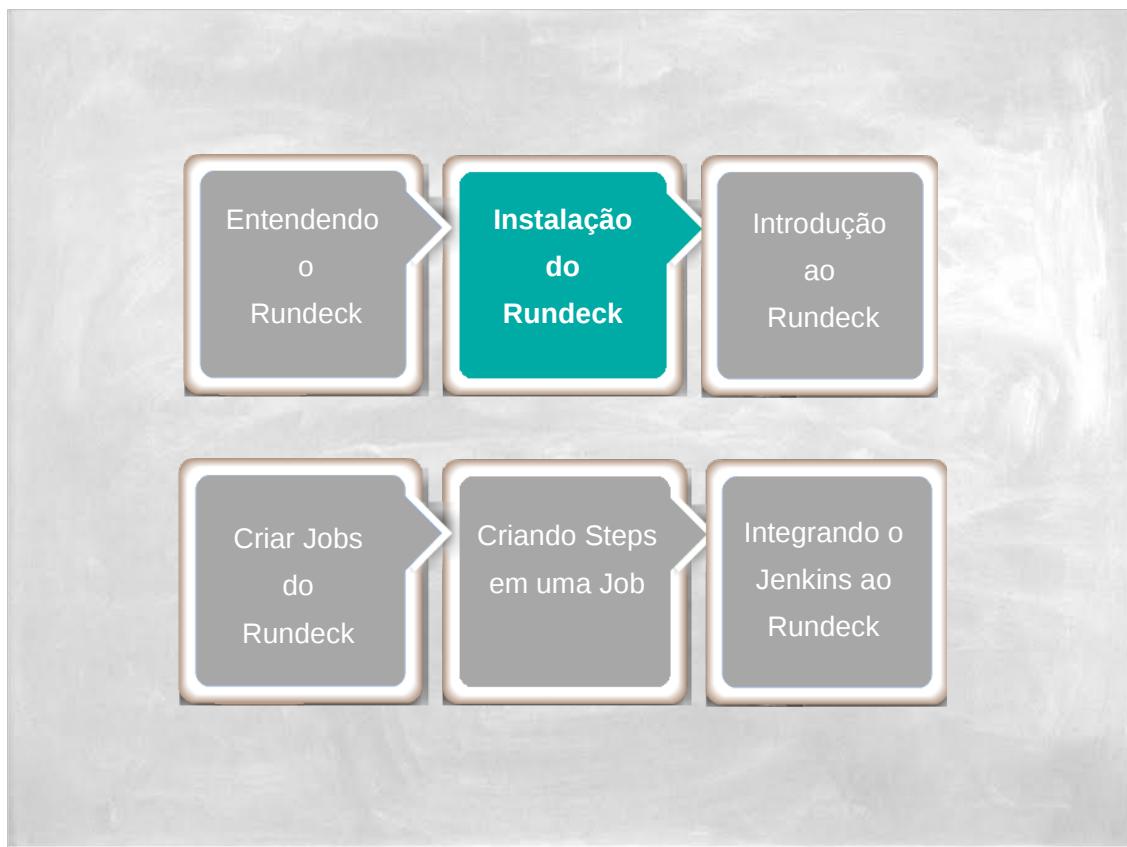
---

---

---

---

---



## Anotações:

## Instalação do Rundeck

Máquina: Devops

Agora, vamos executar o playbook do ansible para instalação do Rundeck.

1

Abra o arquivo /etc/ansible/roles/rundeck/tasks/main.yml para que possamos conferir os passos de instalação do Rundeck.

```
1# vim /etc/ansible/roles/rundeck/tasks/main.yml
```

2

Para executar o playbook, podemos usar o comando:

```
1# ansible-playbook  
/etc/ansible/playbooks/ambiente/automation.yml
```

237



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Instalação do Rundeck

Máquina: Devops

Após a instalação, precisamos fazer algumas configurações dentro do diretório do Rundeck.

1

O diretório de configurações do Rundeck é o /etc/rundeck/

```
1# cd /etc/rundeck
```

2

No arquivo framework.properties vamos tirar o localhost e colocar nosso ip da máquina Automation.

```
1# vim framework.properties
```

```
framework.server.name = dexter
```

```
framework.server.hostname = automation.dexter.com.br
```

```
framework.server.port = 4440
```

```
framework.server.url = http://automation.dexter.com.br:4440
```

238



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Instalação do Rundeck

Máquina: Devops

3

Precisamos alterar também o arquivo “rundeck-config.properties”, pois o mesmo também vem com localhost por padrão.

```
1# vim rundeck-config.properties  
grails.serverURL=http://192.168.200.10:4440  
  
1# vim rundeck-config.properties  
grails.serverURL=http://automation.dexte.com.br:4440
```

4

Feito isso, podemos reiniciar o serviço.

É comum que o rundeck demore um pouco para subir, porém podemos ver o momento que ele ficará em listen.



```
1# ss -ltn | grep 4440
```

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

239

## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

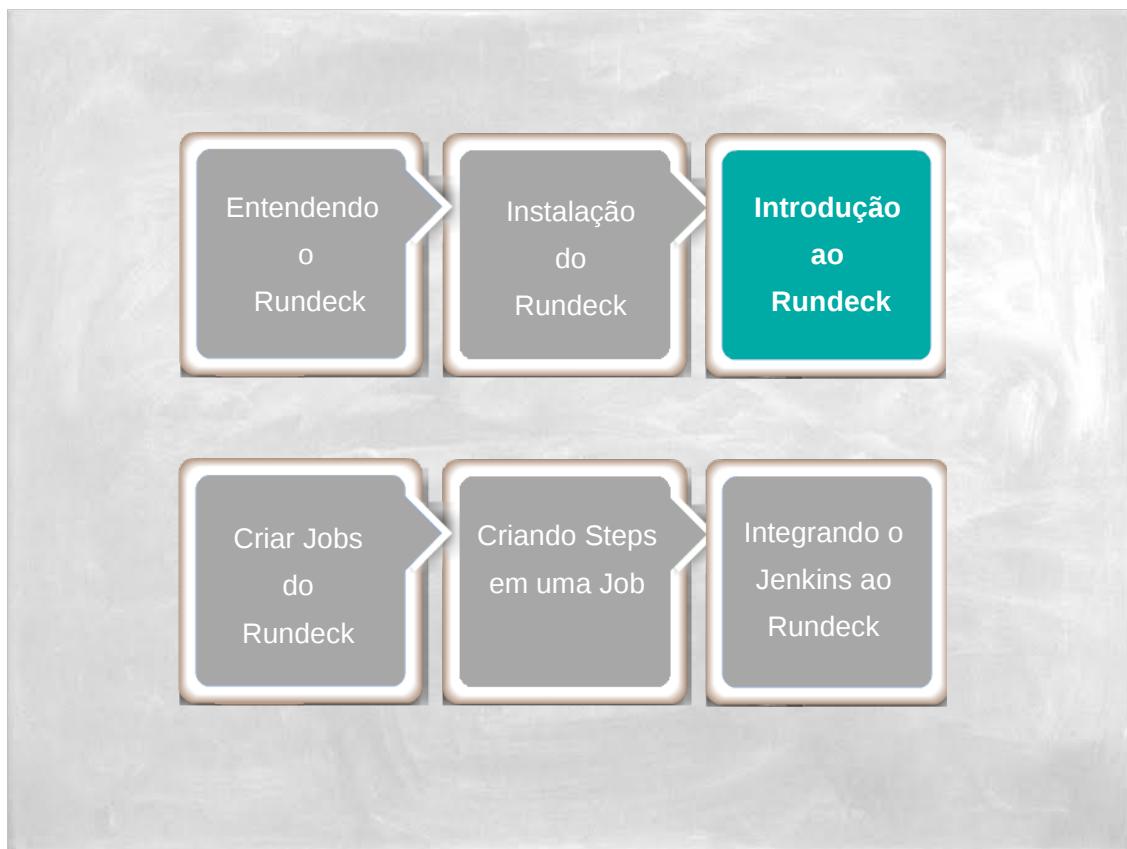
---

---

---

---

---



## Anotações:

# Introdução ao Rundeck

Ao instalar o Rundeck por padrão, ele já disponibiliza a interface.

Usuário: **admin** / Senha: **admin**

Username

Password

**Login**

241

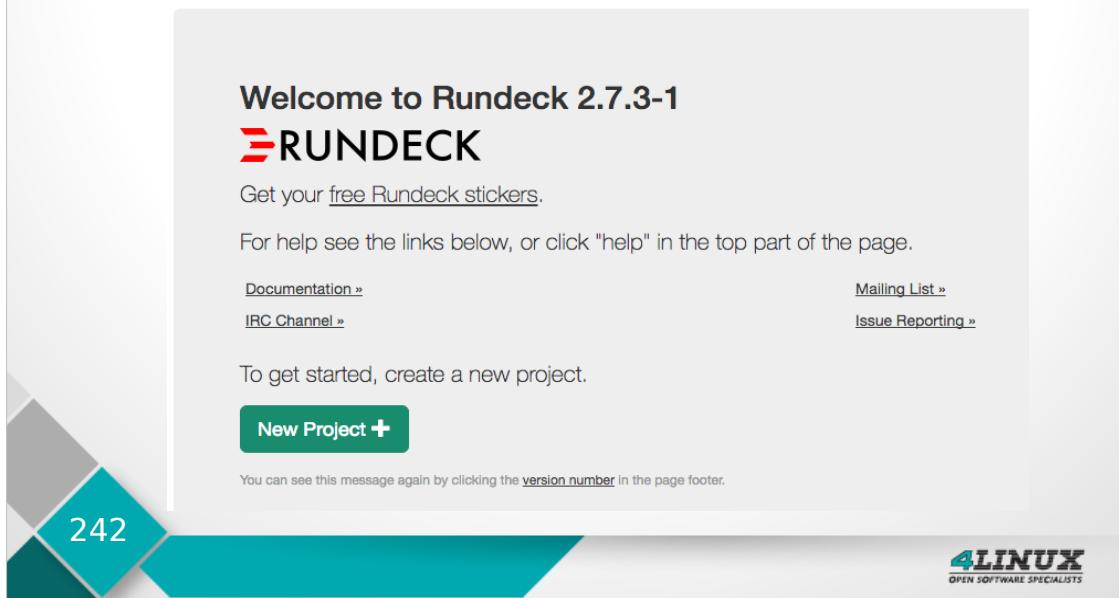
**4LINUX**  
OPEN SOFTWARE SPECIALISTS

## Anotações:

## Introdução ao Rundeck

1

Após colocarmos a senha, o Rundeck nos dará a opção de criar um projeto.



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Introdução ao Rundeck

2

Vamos criar um novo projeto.

To get started, create a new project.

New Project +



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Introdução ao Rundeck

3

Após criarmos o projeto, devemos definir um nome e uma descrição.

Create a new Project

Project Name

dexter

Description

primeiro projeto com rundeck

244

4LINUX  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

**Se você é um novo usuário, saber alguns conceitos irá ajudá-lo a usar ou integrar Rundeck em seu ambiente.**

**- Políticas de controle de acesso baseadas em funções:**

Uma política de controle de acesso do Rundeck, concede aos usuários e grupos de usuários certos privilégios para executar ações com recursos do rundeck como projetos, jobs, nós, comandos e API.

**- Projects**

Um projeto é o local para separar a atividade de gerenciamento. Todas as atividades de Rundeck ocorrem dentro do contexto de um projeto.

**- Nodes**

Um nó é um recurso (instância física ou virtual) de um host de rede acessível.

**- Commands**

Um comando é uma cadeia de execução em um Nó.

**- Executions**

Uma execução é uma representação da atividade, trabalho executado ou concluído. Os dados sobre a execução são utilizados pelo rundeck para monitorar o progresso de um trabalho ou comando, e depois para relatar o que aconteceu.

**- Plugins**

A maioria do que o Rundeck faz é através de um de seus plugins. Existem plugins para executar comandos em nós, executar etapas de uma Job, enviar uma notificação sobre o status da Job, coletar informações sobre os hosts em sua rede, copiar um arquivo para um servidor remoto, armazenar e transmitir logs ou conversar com um diretório de usuário.

## **Recursos do Rundeck**

- Web API;
- Execução de comandos distribuídos;
- Fluxos de trabalho em várias etapas;
- Execução de tarefas programadas;
- Interface gráfica para gerenciamento e execução de tarefas;
- Política de controle de acesso com suporte para LDAP/ActiveDirectory;
- Histórico e logs de auditoria;

## Introdução ao Rundeck

1 Para esse projeto de teste, vamos usar as configurações padrões.

The screenshot shows the Rundeck configuration interface for a project. It includes fields for SSH Key File path (set to /var/lib/rundeck/.ssh/id\_rsa), SSH Key Storage Path (with a 'Select...' button), SSH Password Storage Path (with a 'Select...' button), and SSH Authentication (set to privateKey). Below these are two radio button options: 'Script Execution' (selected) and 'Stub'. A note under 'Script Execution' says: 'Delegates file copying to an external script. Can be configured project-wide or on a per-node basis.' A note under 'Stub' says: 'Prints information about file copy request instead of copying it. (Useful for mocking processes.)'. At the bottom is a 'Create' button. The background features a teal and grey geometric design with the number 246.

246

4LINUX  
OPEN SOFTWARE SPECIALISTS

## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Introdução ao Rundeck

Após a criação, já estaremos dentro do projeto.

The screenshot shows the Rundeck web interface. At the top, there is a dark header bar with the Rundeck logo and navigation links for 'Jobs', 'Nodes', 'Commands', and 'Activity'. Below the header, the main content area has a light gray background. On the left, there is a sidebar with a teal and gray geometric pattern containing the number '247'. In the center, the word 'dexter' is displayed in a large, bold, black font, with the subtitle 'primeiro projeto com rundeck' in smaller text below it. To the right of the project name, the text '0 Executions In the last day' is shown. At the bottom right of the main area, there is a logo for '4LINUX OPEN SOFTWARE SPECIALISTS'. The footer of the page contains copyright information: 'Rundeck 2.7.3-1 (cafecito tomato paperclip) 2017-03-11' and '© Copyright 2017 #SimplifyOps. All rights reserved. Licenses'.

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

A barra de navegação superior contém um menu para selecionar o projeto desejado. Se apenas existir um projeto, o menu será automaticamente selecionado.

### **Jobs**

- Na página Jobs, é possível listar, criar e executar Jobs. Um filtro configurável permite que um usuário limite a lista de Jobs para as Jobs que correspondem aos critérios de filtragem. Essas configurações de filtro podem ser salvas em perfis de Usuários. Somente os trabalhos autorizados serão visíveis.

### **Nodes**

- A página Nodes é utilizada para procurar seus nós configurados no modelo de recursos do projeto. Um controle de filtro pode ser usado para limitar a listagem apenas aos recursos de nós que correspondem aos critérios de filtro. Dada a autorização apropriada, você também pode executar comandos apenas para o seu conjunto de nós filtrados.

### **Commands**

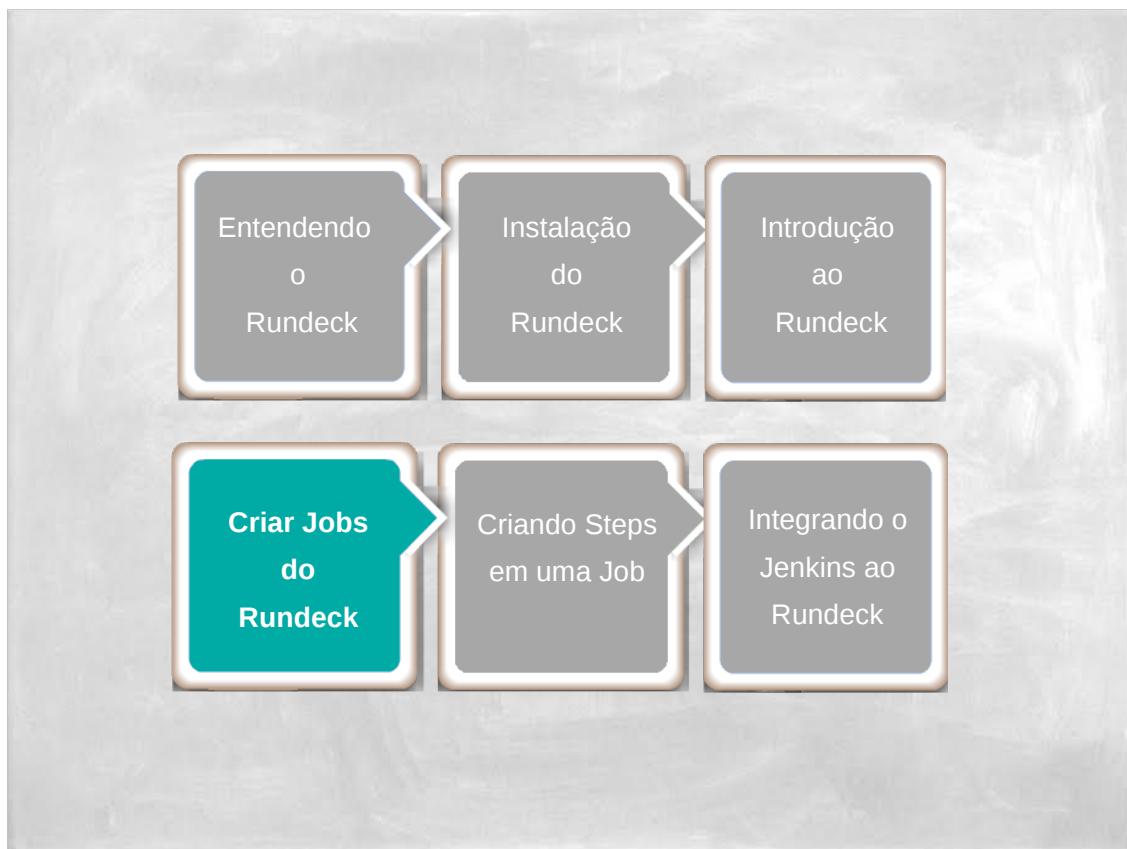
- A página Comandos permite executar comandos abitários nos nós que correspondem ao filtro de nó.

### **Activity**

- A partir da página Atividade, é possível visualizar os comandos em execução no momento, e também Trabalhos ou ainda procurar o histórico de execução. O histórico de execução pode ser filtrado com base nos parâmetros selecionados pelo usuário. Uma vez definido o filtro, o histórico correspondente é exibido.

### **Configure**

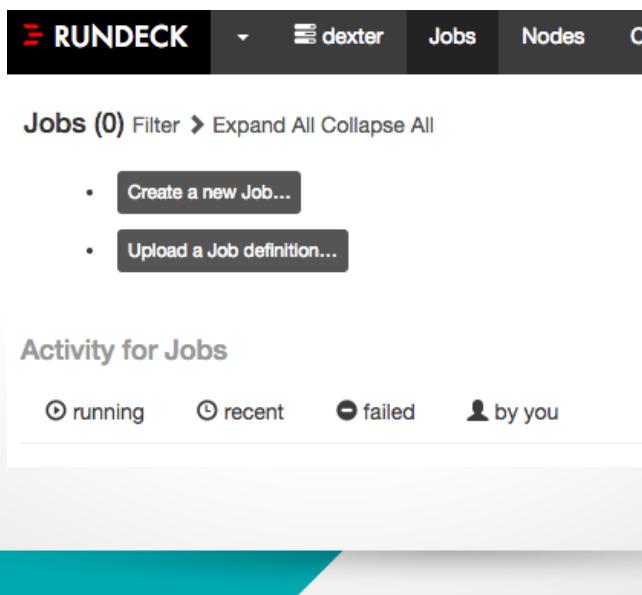
- Se seu login pertence ao grupo "admin" e, portanto, concedeu privilégios "admin", um ícone "Configurar" será exibido na barra de navegação superior. Na página Configurar, é possível editar a configuração do projeto, exportar e importar arquivos do projeto, exibir informações do sistema e ver quais plugins estão instalados.



## Anotações:

## Criando Jobs com Rundeck

Vamos criar uma job em “create a new job”.



The screenshot shows the Rundeck dashboard. At the top, there's a teal header bar with the title "Criando Jobs com Rundeck". Below it is a dark navigation bar with the Rundeck logo, a dropdown menu, and links for "dexter", "Jobs", "Nodes", and "Cloud". The main content area has a light gray background. It displays a section titled "Jobs (0)" with a "Filter" dropdown and buttons for "Expand All" and "Collapse All". Below this is a list with two items: "Create a new Job..." and "Upload a Job definition...". Further down, there's a section titled "Activity for Jobs" with four status filters: "running", "recent", "failed", and "by you". On the left side of this section is a decorative graphic consisting of overlapping triangles in shades of gray and teal, with the number "250" in white. In the bottom right corner of the main content area, there's a logo for "4LINUX OPEN SOFTWARE SPECIALISTS".

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Criando Jobs com Rundeck

Na parte superior, o nome da job e o grupo são muito importantes porque será usado pelo plugin jenkins na consulta da API. Vamos definir essa job como “teste” e também o grupo “dexter”.

The screenshot shows the 'Create New Job' interface. The 'Job Name' field contains 'teste'. The 'Group' dropdown is set to 'dexter'. The 'Description' field contains 'Job de teste no rundeck.'. A note below the description says: 'The first line of the description will be shown in plain text, the rest will be rendered with Markdown. More'.

251

4LINUX  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Criando Jobs com Rundeck

Nessa job de teste, teremos um step “command” que executará um comando na local.

### Add a Step

Click on a Step type to add.

Node Steps     Workflow Steps

**Runs once for each node in the workflow.**

Command - Execute a remote command

Script - Execute an inline script

Script file or URL - Execute a local script file or a script from a URL

Job Reference - Execute another Job for each Node

2 Node Step Plugins

Copy File - Copy a file to a destination on a remote node.

Local Command - Run a command locally on the server

252



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

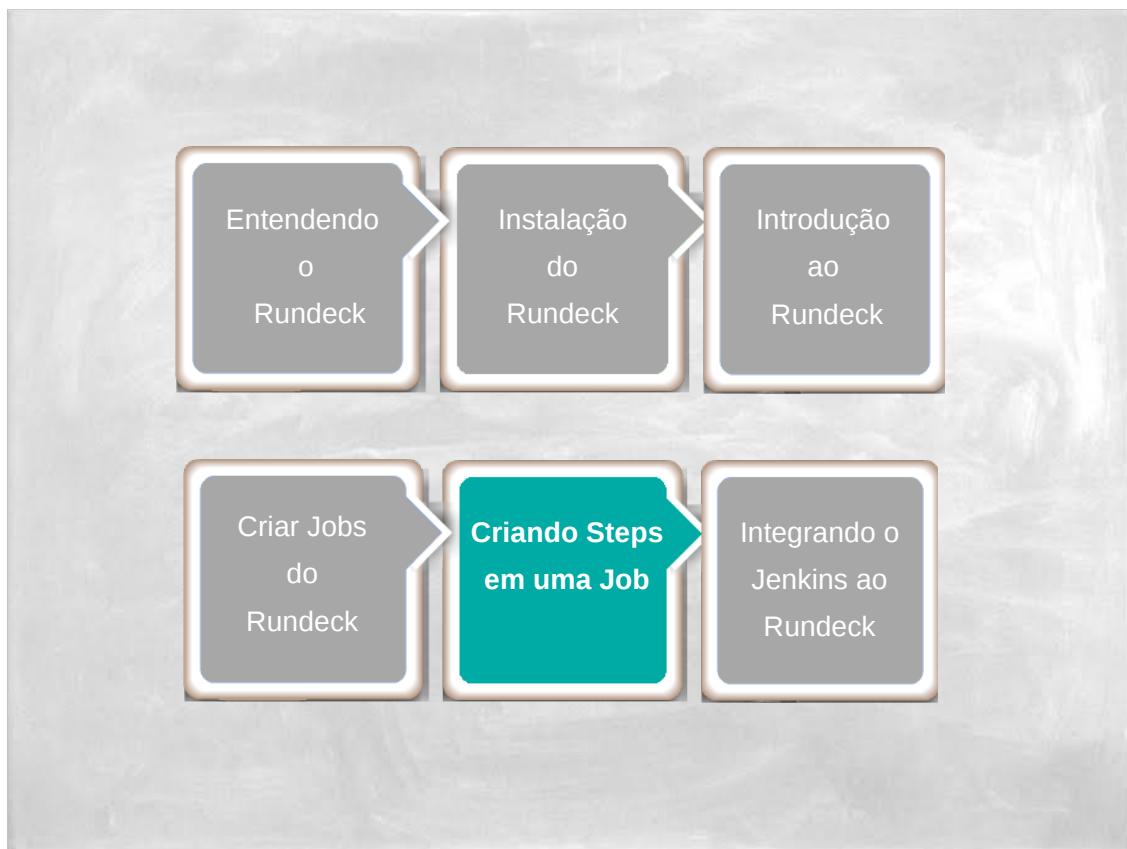
---

---

---

---

---



## Anotações:

## Criando Steps com Rundeck

Vamos adicionar o comando “mkdir” como teste, após isso podemos salvar.

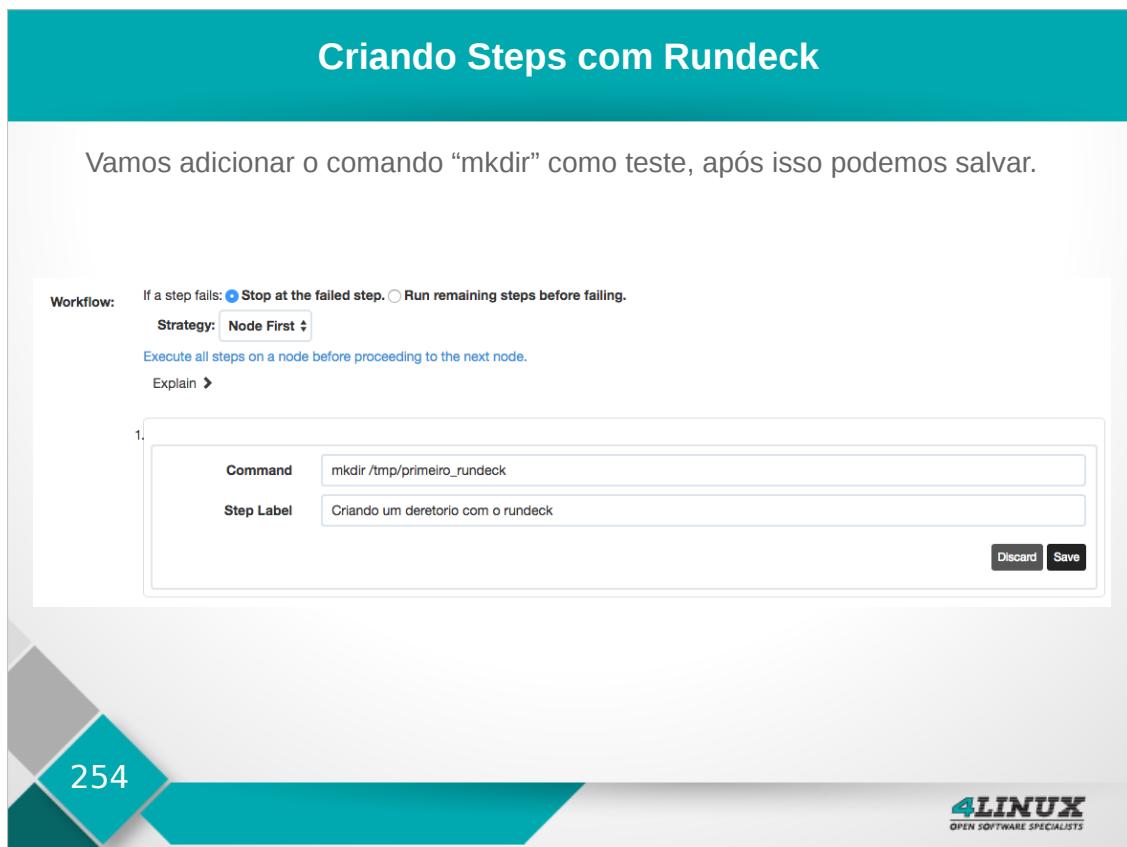
Workflow: If a step fails:  Stop at the failed step.  Run remaining steps before failing.  
Strategy: Node First ↗  
Execute all steps on a node before proceeding to the next node.  
Explain ➔

1.

Command	mkdir /tmp/primeiro_rundeck
Step Label	Criando um diretório com o rundeck

**254**

**4LINUX**  
OPEN SOFTWARE SPECIALISTS



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Criando Steps com Rundeck

Vamos finalizar nossa job.

Retry will occur if the job fails or times out, but not if it is manually killed. Can use an option value reference like "\${option.retry}".

<b>Log Output Limit</b>	E.g as '100', '100/node' or '100MB'	<b>Log Limit Action</b>	<input checked="" type="radio"/> Halt with status: ? 'failed', 'aborted', or any string
Enter either maximum total line-count (e.g. "100"), maximum per-node line-count ("100/node"), or maximum log file size ("100MB", "100KB", etc.), using "GB", "MB", "KB", "B" as Giga- Mega- Kilo- and bytes.		<input type="radio"/> Truncate and continue	Action to perform if the output limit is reached.
UUID			
<b>Cancel</b>	<b>Create</b>		

## Anotações:



## Criando Steps com Rundeck

Quando criamos em uma job do rundeck por default, ela só executará através de um agendamento, ou se nós clicarmos em "Run Job Now".

The screenshot shows the Rundeck interface for a job named 'teste'. At the top, there's a navigation bar with 'dexter' and a 'Job' icon. Below it, the job name 'teste' is displayed with an 'Action' dropdown arrow. A sub-header says 'Job de teste no rundeck.' There are two tabs: 'Prepare and Run...' (selected) and 'Definition'. Under 'Input Options', it says 'None for this Job.' On the right, there are three buttons: 'Run Job Now' (green), 'Run Job Later' (grey), and 'Follow execution' (checkbox checked). Below these are 'Log level' options ('Normal' is selected, 'Debug' is unselected) with a note: 'Debug level produces more output'. A 'Statistics' section shows 'EXECUTIONS' with a value of '0'. At the bottom, there's a teal decorative bar with the number '256' and the '4LINUX OPEN SOFTWARE SPECIALISTS' logo.

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Criando Steps com Rundeck

Vamos executar essa job clicando em “Run Job Now”.

**Run Job Now ►** **Run Job Later ⏱**  
 Follow execution



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

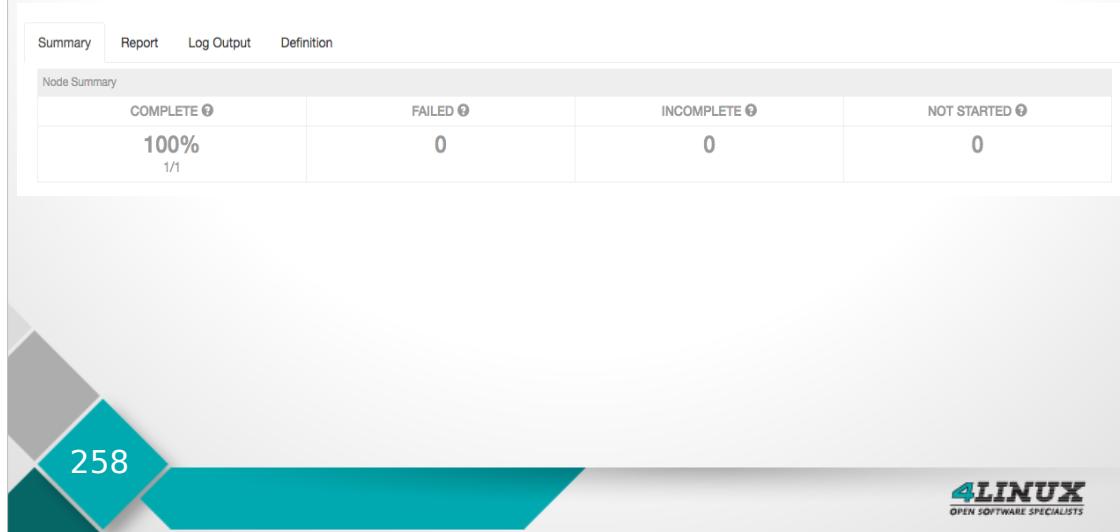
---

---

---

## Criando Steps com Rundeck

Após a execução, ele retornará 100% na Tag “complete”, e caso o step executado tenha uma saída na tela, ele mostraria em “Log Output”.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

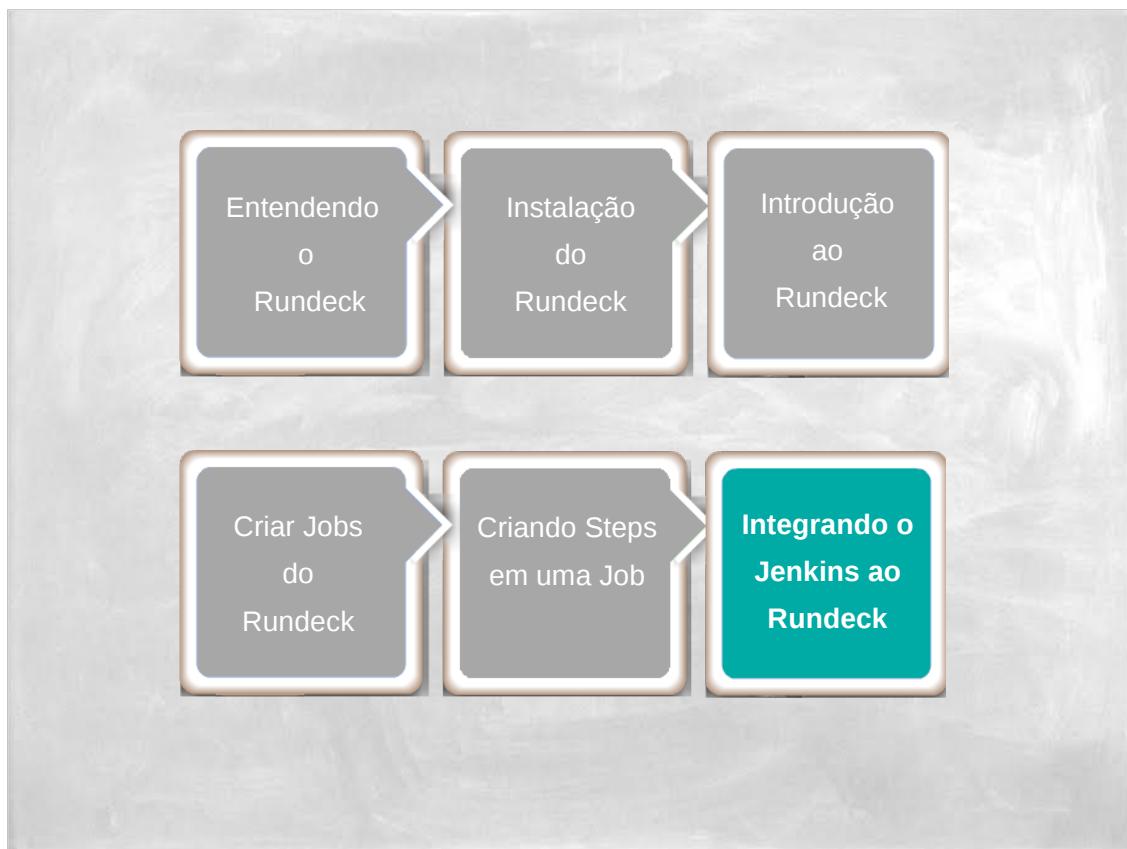
---

---

---

---

---



## Anotações:

## Integrando o Rundeck com o Jenkins

1

Vamos criar o projeto que será usado no deploy da nossa aplicação.

To get started, create a new project.

New Project +



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Integrando o Rundeck com o Jenkins

2

Após criarmos o projeto, devemos definir um nome e uma descrição.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Integrando o Rundeck com o Jenkins

3

Para esse projeto, vamos usar as configurações padrões.

SSH Key File path `/var/lib/rundeck/.ssh/d_rsa`  
File Path to the SSH Key to use

SSH Key Storage Path

SSH Password Storage Path

SSH Authentication

**Script Execution**  
Delegates file copying to an external script. Can be configured project-wide or on a per-node basis.

**Stub**  
Prints information about file copy request instead of copying it. (Useful for mocking processes.)

**262**

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Integrando o Rundeck com o Jenkins

Após a criação, já estaremos dentro do projeto.

The screenshot shows the Rundeck web interface. At the top, there is a dark header bar with the Rundeck logo and navigation links: deploy, Jobs, Nodes, Commands, and Activity. Below the header, the main title is "deploy". To the right of the title, it says "0 Executions In the last day". Underneath the title, there is a small footer with the text "Rundeck 2.7.3-1" followed by a small icon and the text "'cafecito tomato paperclip' 2017-03-11", and "© Copyright 2017 #SimplifyOps. All rights reserved. [Licenses](#)". On the left side of the main area, there is a decorative graphic consisting of overlapping triangles in shades of grey and teal, with the number "263" displayed prominently. On the right side, there is a logo for "4LINUX OPEN SOFTWARE SPECIALISTS".

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

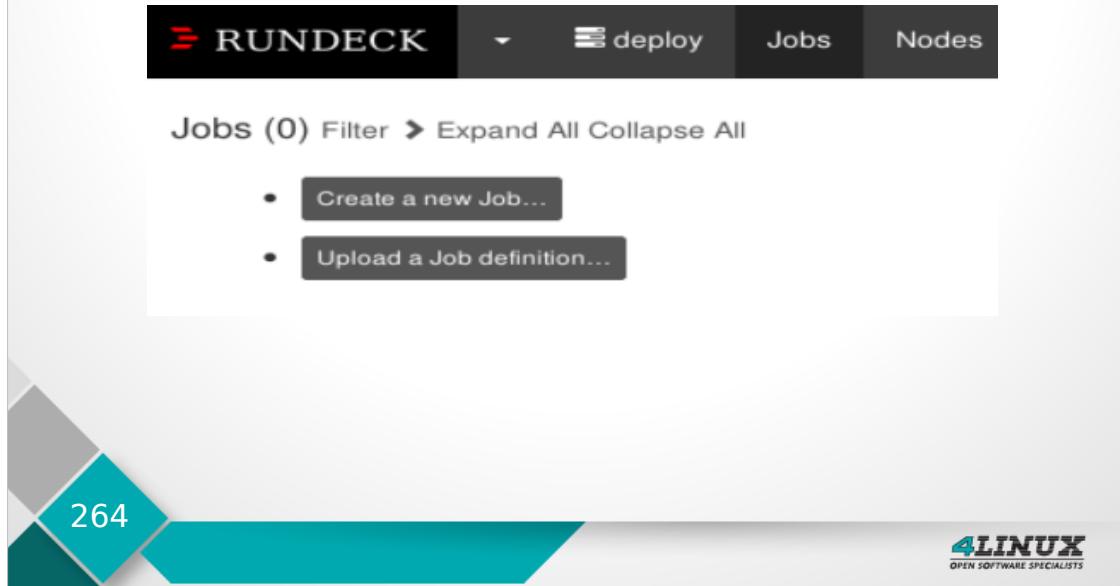
---

---

---

## Integrando o Rundeck com o Jenkins

Vamos criar uma job na aba Jobs, em “create a new job”.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

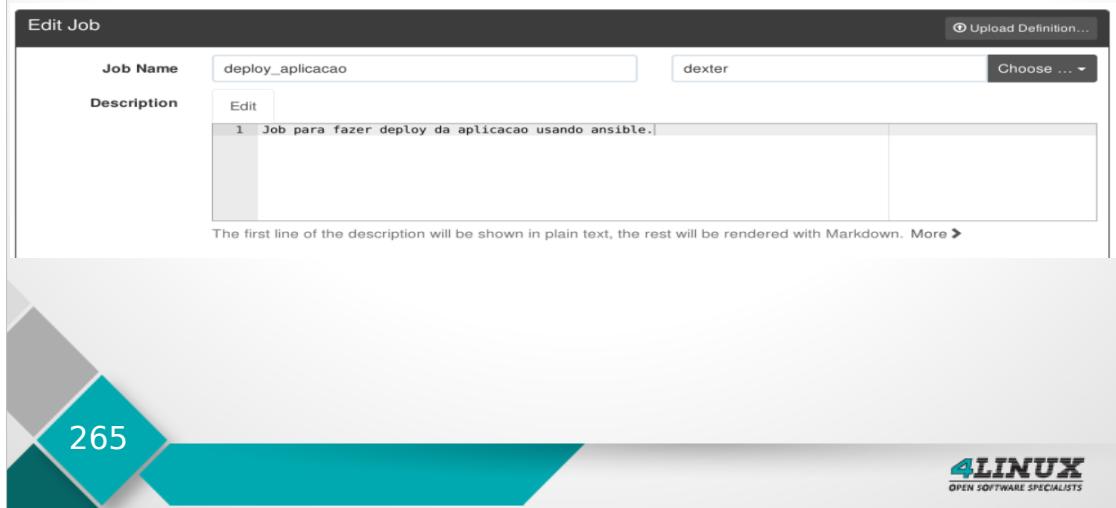
---

---

---

## Integrando o Rundeck com o Jenkins

Na parte superior, o nome da job e o grupo são muito importantes, porque será usado pelo plugin jenkins na consulta da API. Vamos definir essa job como “deploy\_aplicacao” e também o grupo “dexter”.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

# Integrando o Rundeck com o Jenkins

Nessa job de teste, teremos um step “command” que executará um comando na local.

## Add a Step

**Click on a Step type to add.**

## Node Steps      Workflow Steps

**Runs once for each node in the workflow.**

## > Command - Execute a remote command

## #! Script - Execute an inline script

## # Script file or URL - Execute a local script file or a script from a URL

### ■ Job Reference - Execute another Job for each Node

## 2 Node Step Plugins

◆ Copy File - Copy a file to a destination on a remote node.

◆ Local Command - Run a command locally on the server

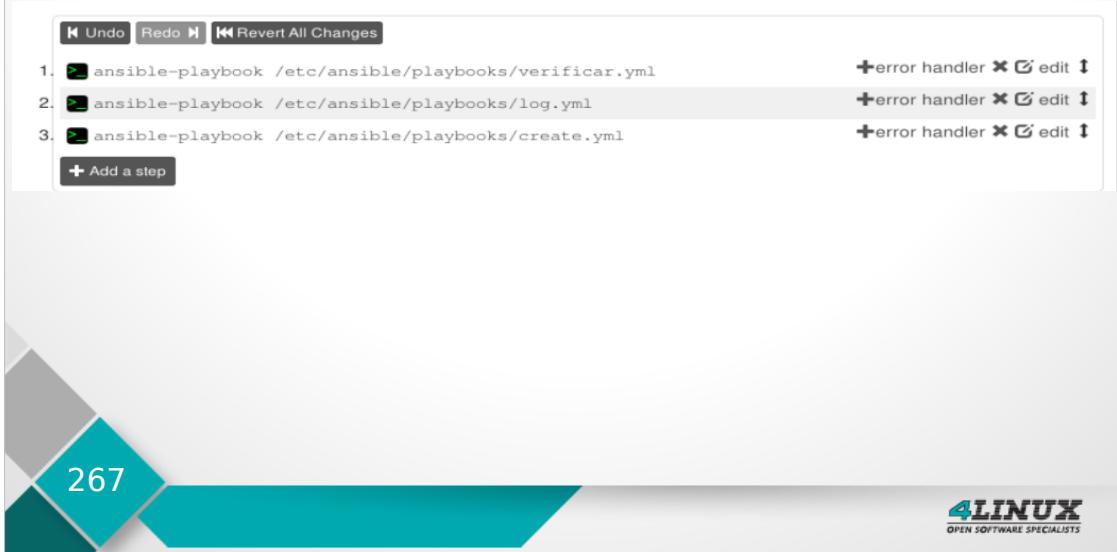
266



## Anotações:

# Integrando o Rundeck com o Jenkins

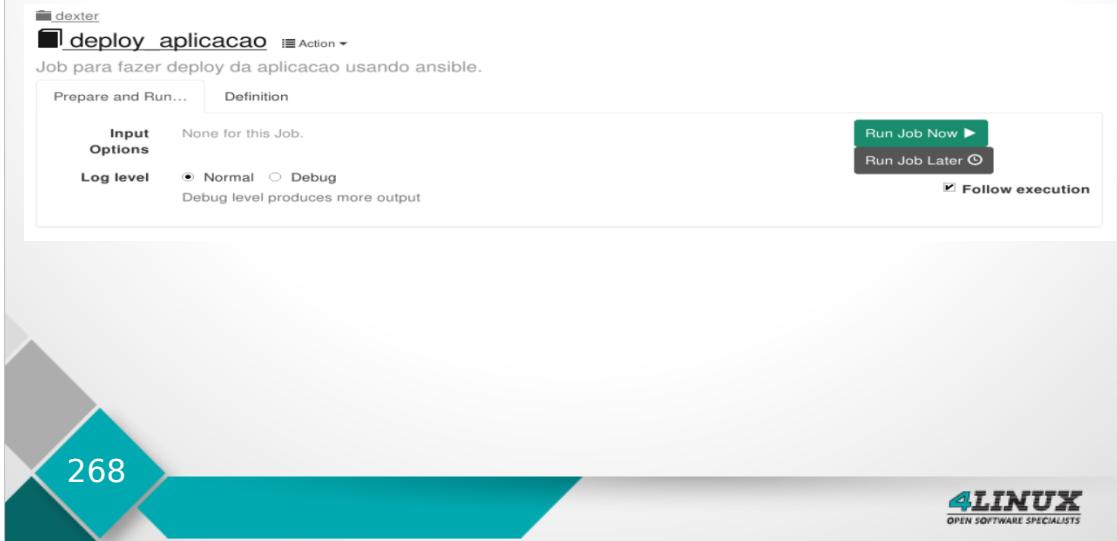
Nessa job teremos três steps que futuramente executarão os playbooks ansible para deploy da aplicação.



## Anotações:

## Integrando o Rundeck com o Jenkins

Feito isso, teremos a Job pronta para ser executada pelo jenkins no laboratório Dexter.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Projeto Dexter

No laboratório final **Dexter**, todos os nossos repositórios serão integrados com as Jobs do Jenkins e do Rundeck para cada deploy.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Integração com Jenkins

### Objetivos da Aula

- Fazer uma Introdução ao Jenkins;
- Dar os Primeiros Passos com Jenkins;
- Gerenciar Plugins no Jenkins;
- Trabalhar com dependências entre Jobs;
- Integrar o Jenkins ao Gitlab.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

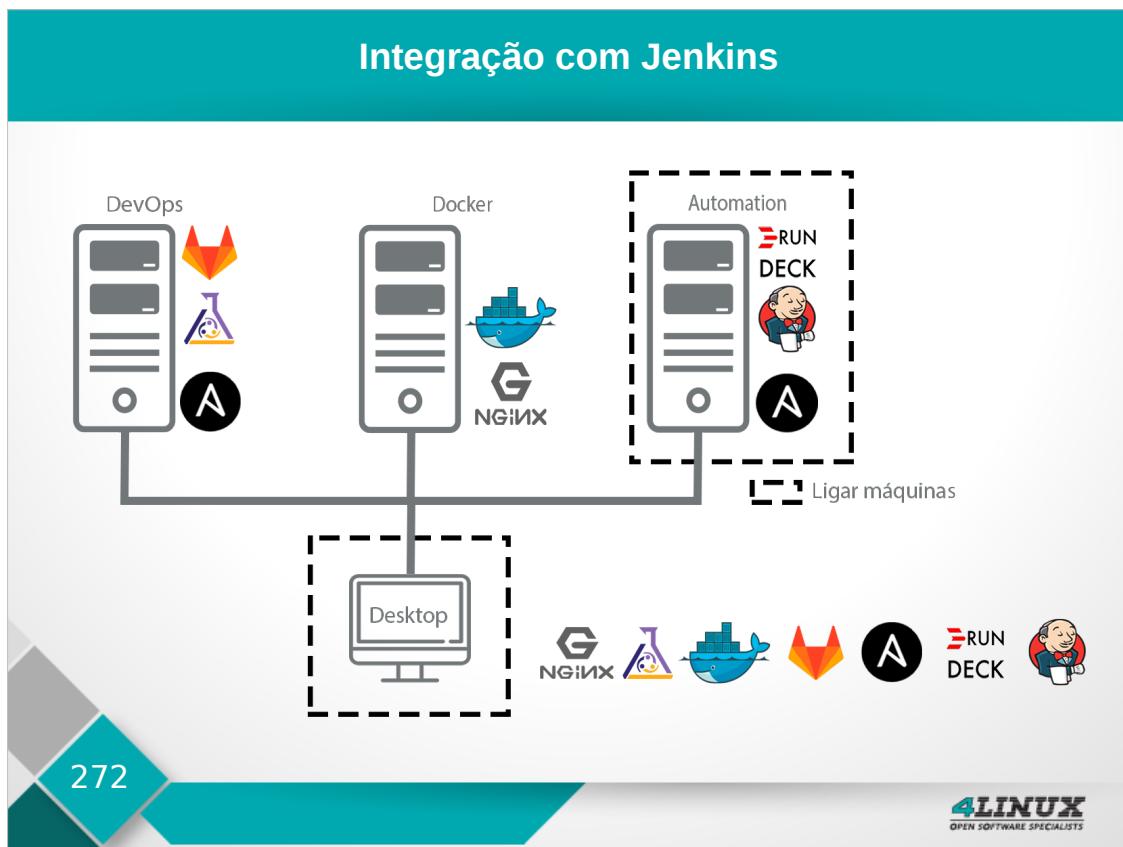
---

---

---

---

---



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

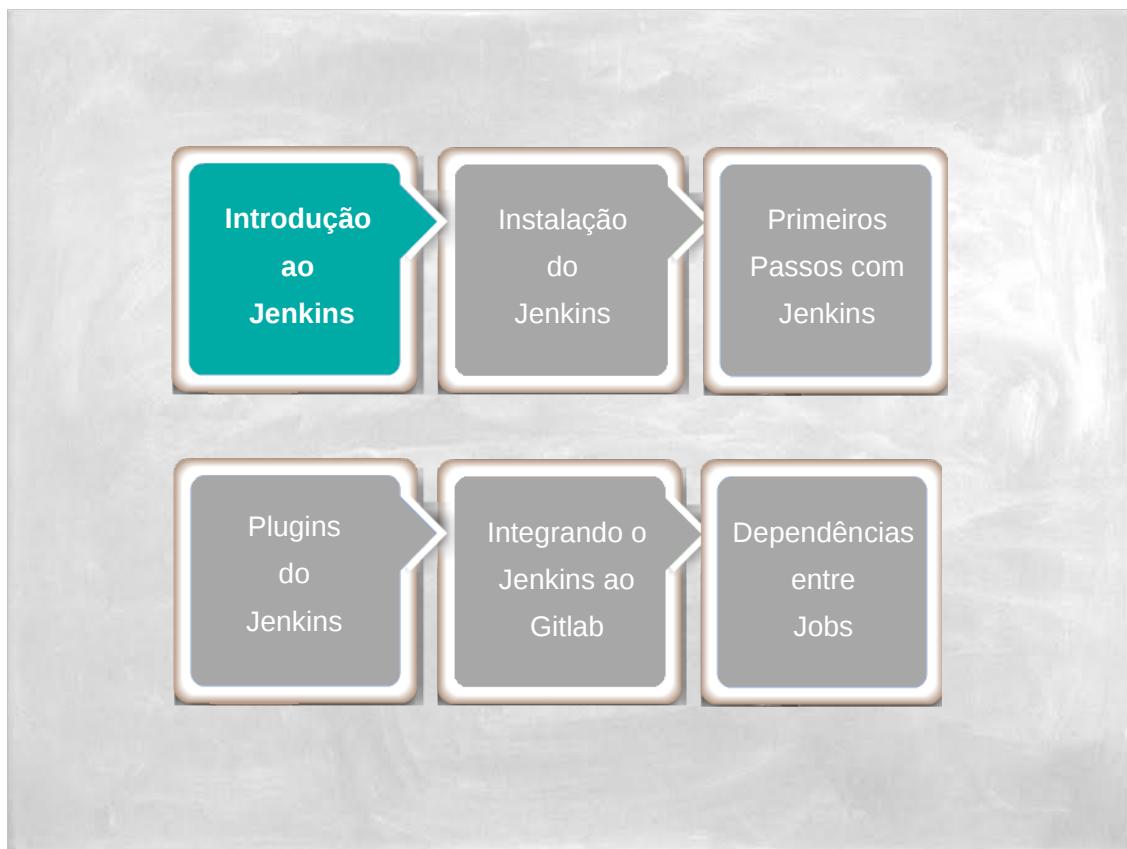
---

---

---

---

---



## Anotações:

## Introdução ao Jenkins

O **Jenkins** é uma aplicação para Continuous Integration (CI) e Continuous Delivery (CD).

Ele possui integração com várias ferramentas, inclusive com o Gitlab e Rundeck, nos quais iremos utilizar para integrar o ambiente da Dexter.



### Continuous Integration

**Continuous Integration** ou **Integração Contínua** é uma prática de desenvolvimento em que os membros de um time integram seu trabalho frequentemente, normalmente essa integração é realizada diariamente. Cada integração é verificada por uma tarefa ou build automatizada para detectar erros de integração da forma mais rápida possível. A integração contínua diminui a probabilidade de erros dentro de um projeto.

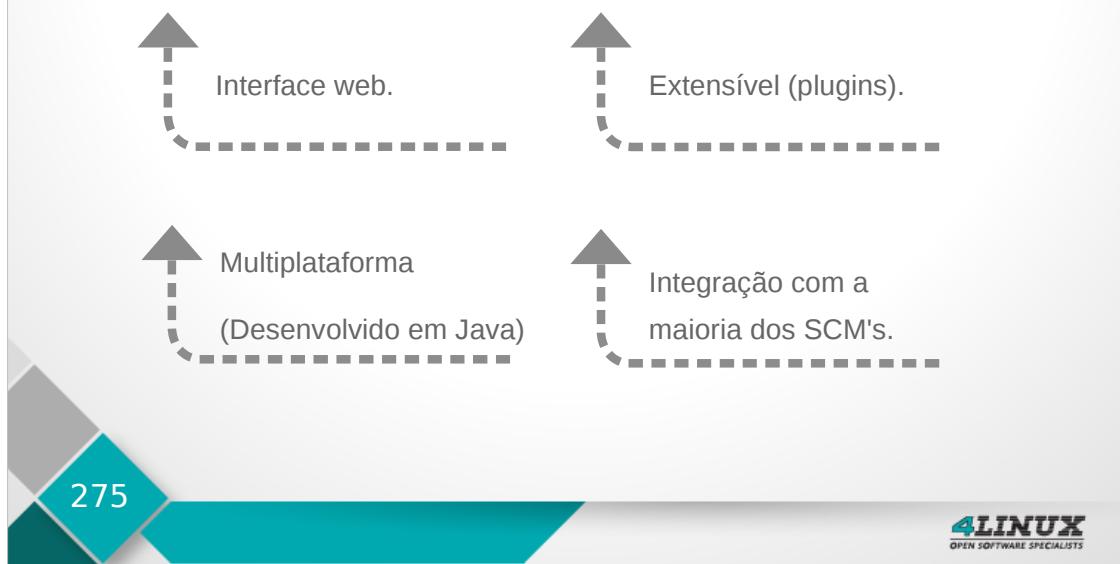
### Continuous Delivery

**Continuous Delivery** ou **Entrega Contínua** é uma prática de desenvolvimento usada na construção de software, de uma maneira que você possa disponibilizar mudanças em seu ambiente de produção a qualquer momento.

Em um ambiente de entrega contínua, são realizadas mudanças pequenas e com maior frequência de Deploy, o que diminui o risco de erros em ambientes de produção, simplesmente pelo fato de serem pequenas mudanças.

## Introdução ao Jenkins

### Características do Jenkins



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

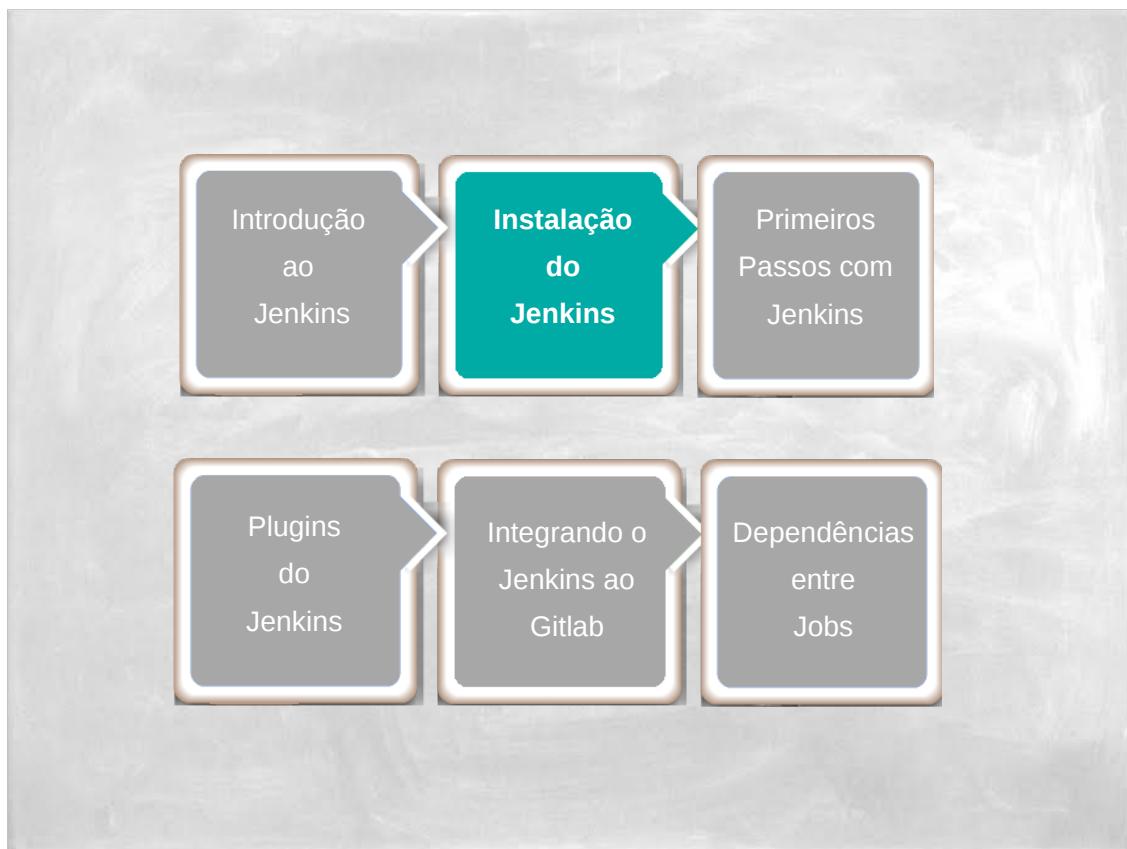
---

---

---

---

---



## Anotações:

## Instalação do Jenkins

Máquina: Automation

Agora, vamos executar o playbook do ansible para instalação do Jenkins.

1

Abra o arquivo /etc/ansible/roles/jenkins/tasks/main.yml para que possamos conferir os passos de instalação do Jenkins.

```
1# vim /etc/ansible/roles/jenkins/tasks/main.yml
```

2

Para executar o playbook, podemos usar o comando:

```
1# ansible-playbook  
/etc/ansible/playbooks/ambiente/automation.yml
```

277



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Instalação do Jenkins

Máquina: Automation

Após a instalação, vamos alterar a porta e iniciar o serviço.

- Vamos alterar a linha do arquivo "/etc/default/jenkins".

```
1# vim /etc/sysconfig/jenkins
```

```
JENKINS_PORT=8080
```

- Após alterar a porta, podemos iniciar o serviço.

```
1# systemctl restart jenkins
```

278



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Instalação do Jenkins

Ao instalar o jenkins, ele pede para desbloquear usando uma senha. Por padrão, a mesma fica no caminho informado.

Getting Started

### Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

279

Continue

4LINUX  
OPEN SOFTWARE SPECIALISTS



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Instalação do Jenkins

Máquina: Automation

1

Vamos abrir o arquivo com o vim para pegar a senha.

```
1# vim /var/lib/jenkins/secrets/initialAdminPassword
```

2

Após colocarmos a senha, o jenkins nos dará a opção de instalar os plugins sugeridos ou selecionar quais instalar.

### Customize Jenkins

Plugins extend Jenkins with additional features to support many different r

#### Install suggested plugins

Install plugins the Jenkins community finds most useful.

#### Select plugins to install

Select and install plugins most suitable for your needs.

280



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Instalação do Jenkins

Máquina: Automation

3

Para o laboratório da Dexter, usaremos o plugin do gitlab e o do Rundeck, vamos instalar os sugeridos.

### Install suggested plugins

Install plugins the Jenkins community finds most useful.

281

4LINUX  
OPEN SOFTWARE SPECIALISTS

## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Instalação do Jenkins

Máquina: Automation

4

Após a instalação, teremos a opção de criar o usuário com privilégios de administrador, vamos criar o usuário devops com a senha 4linux.

### Create First Admin User

Nome de usuário:	<input type="text" value="devops"/>
Senha:	<input type="password" value="....."/>
Confirmar a senha:	<input type="password" value="....."/>
Nome completo:	<input type="text" value="devops"/>
Endereço de e-mail:	<input type="text" value="devops@dexter.com.br"/>

282

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Instalação do Jenkins

5

Com isso, podemos começar a utilizar o jenkins.

Getting Started

# Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)

283

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

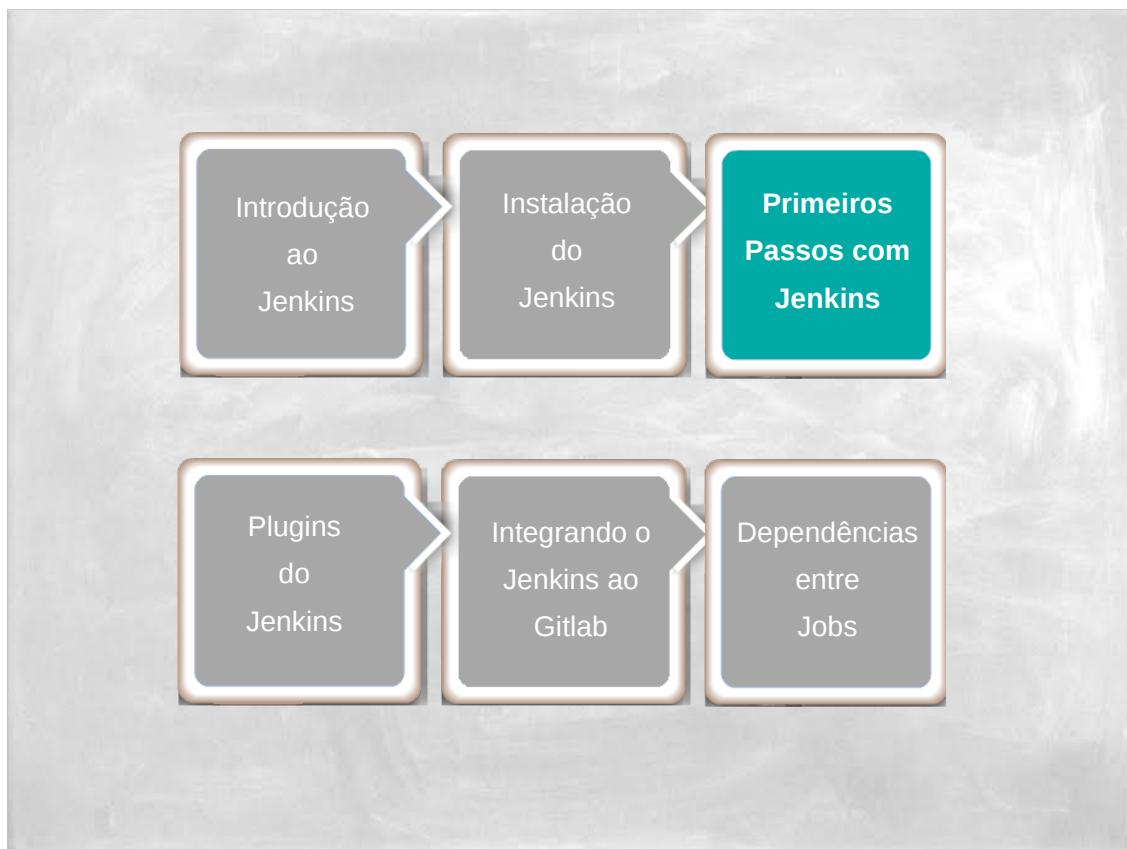
---

---

---

---

---



## Anotações:

## Primeiros Passos com Jenkins

Tela inicial do Jenkins:

The screenshot shows the Jenkins dashboard with a teal header. The main content area has a large "Bem-vindo ao Jenkins!" message with a call to action: "Por favor clique em: **Novo job** para iniciar." On the left sidebar, there are links for "Novo job", "Usuários", "Histórico de compilações", "Gerenciar Jenkins", and "Credenciais". Below the sidebar, there are two sections: "Fila de builds" (which says "Nenhum build na fila.") and "Estado do executor de builds" (which lists "1 Parado" and "2 Parado"). A watermark for "4LINUX OPEN SOFTWARE SPECIALISTS" is visible at the bottom right.

Novo Job – Criação de novas jobs do jenkins,  
Usuários – Gerenciamento de usuários,  
Historico de compilações – Histórico de execução  
das jobs.  
Credenciais – tipos de autenticação entre o jenkins e  
as API's.

### Fila de build

Todo build que inicia, entra nessa fila e vai sendo incrementado com #1, #2 e assim por diante, possibilitando a análise de cada um deles.

## Primeiros Passos com Jenkins

Vamos criar uma Job no Jenkins.

**Jobs** ou **Projects** são tarefas gerenciadas e monitoradas pelo Jenkins.

The screenshot shows the Jenkins dashboard. At the top is a logo of a cartoon character and the word "Jenkins". Below it is a navigation menu with the following items: "Novo job" (highlighted with a cursor icon), "Usuários", "Histórico de compilações", "Gerenciar Jenkins", and "Credentials". A section titled "Fila de builds" contains the message "Nenhum build na fila.". In the bottom right corner, there is a logo for "4LINUX OPEN SOFTWARE SPECIALISTS".

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

286

## Primeiros Passos com Jenkins

Defina o Nome da Job: **Hello Dexter.**

Selecione a opção **Construir um projeto de software free-style** e clique no botão no **OK**.

**Enter an item name**

» Required field

**Construir um projeto de software free-style**

Esta é a central de funcionalidades do Jenkins. Ele pode ser usado para construir projetos que podem ser executados em diferentes ambientes ou que precisam de configurações específicas.

**Pipeline**

Orquestra long-running activities that can span multiple build steps and nodes, organizing complex activities that do not easily fit in free-style projects.

**External Job**

Este tipo de trabalho permite que você grave a execução de um projeto externo para que você possa usar Jenkins como um monitor de execuções.

**Construir projeto de múltiplas configurações**

Apropriado para projetos que necessitam de grande número de configurações diferentes, mas específicas, etc.



### Tipos de Jobs

**Projetos de software Free-Style:** São projetos para um propósito geral, que disponibiliza maior flexibilidade de ferramentas e tarefas no Jenkins.

**Pipeline:** Acompanhamento de execuções longas, mostrando graficamente cada saída de erro ou sucesso .

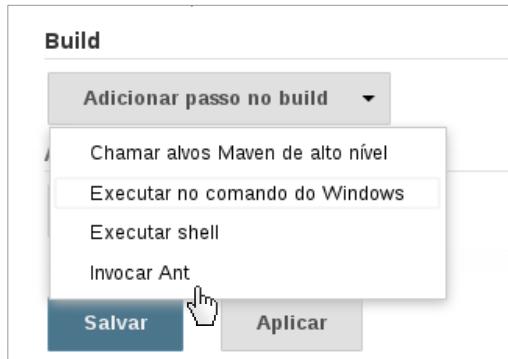
**Projetos Maven:** É uma Job especializada para construção de projetos utilizando o Maven. O Jenkins reconhece os arquivos de configuração do Maven para construir os projetos. Maven é um software para automatizar a construção e dependências de projetos de software, especialmente utilizado para projetos Java.

**Construir um Projeto de Múltipla Configuração:** Define um projeto que pode ser executado para várias outras configurações. Normalmente é utilizado para ambientes de teste de uma aplicação em ambientes diferentes, com banco de dados diferentes e máquinas diferentes.

**External Jobs:** Usado para monitorar a execução de processos não interativos, como tarefas da crontab do sistema e procmail.

## Primeiros Passos com Jenkins

Adicione um passo ao projeto para executar uma **Shell**.



288

4LINUX  
OPEN SOFTWARE SPECIALISTS

- Chamar alvos maven de alto nível

Caso o build seja de uma aplicação Java, o jenkins aceita um arquivo Maven.

- Executar comandos no windows

Caso queria executar alguma job em ambientes microsoft.

- Executar Shell

Utilizar um interpretador de comandos padrão Unix.

- Invocar Ant

Assim como o Maven o Ant é usado para aplicações java.

## Primeiros Passos com Jenkins

Defina o comando a ser executado.

Clique no botão **Salvar** para finalizar a criação da nossa primeira Job no Jenkins.

Build

Executar shell

Comando `mkdir /tmp/primeiro`

Veja [a lista de variáveis de ambiente disponíveis](#)

Adicionar passo no build ▾

Ações de pós-build

Adicionar ação de pós-build ▾

Salvar Apply

289

4LINUX  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Primeiros Passos com Jenkins

### Menu do Projeto

The screenshot shows the Jenkins Project menu. The menu items are:

- Voltar para o Dashboard
- Situação
- Alterações
- Workspace
- Construir agora
- Excluir Projeto
- Configurar

Annotations in Portuguese point to specific items:

- Arquivos criados pelas Builds. (points to 'Workspace')
- Reconfigurar o projeto. (points to 'Configurar')
- Histórico de mudanças. (points to 'Alterações')
- Executar uma tarefa manualmente. (points to 'Construir agora')
- Deletar um projeto. (points to 'Excluir Projeto')

A small teal triangle at the bottom left contains the number 290.

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

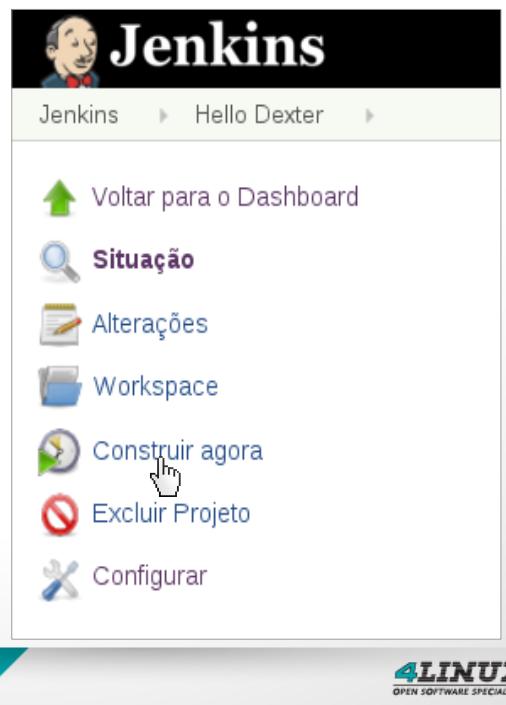
---

---

## Primeiros Passos com Jenkins

Vamos executar manualmente a nossa primeira Tarefa no Jenkins.

Clique no botão **Construir Agora** para executar o projeto.



291

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Primeiros Passos com Jenkins

Podemos visualizar o histórico de Builds do projeto. Uma **Build** representa a construção de uma Job do Jenkins. Nela você pode visualizar a saída dos comandos utilizados, basicamente um log das Builds.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

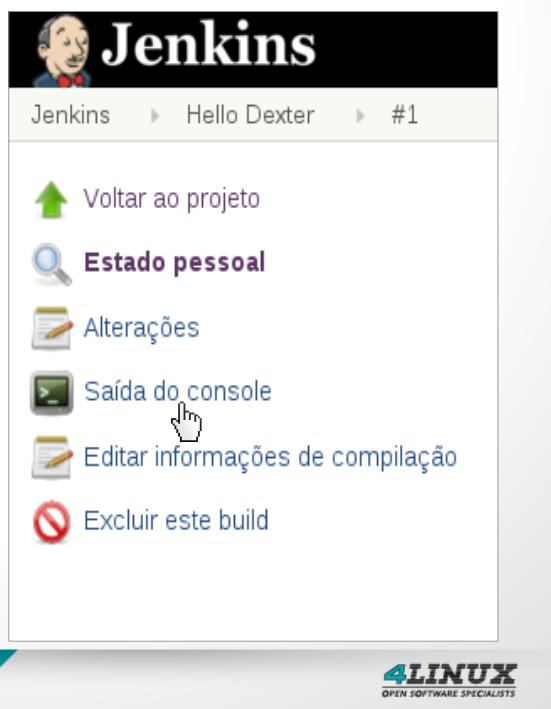
---

---

## Primeiros Passos com Jenkins

Podemos verificar na opção **Saída do Console** a saída dos comandos executados pelo Jenkins.

Clique no link **Saída do Console** para visualizar qual foi a saída da Build em questão.



## Anotações:

## Primeiros Passos com Jenkins

Saída do Console Armazenado pelo Jenkins.



### Saída do console

```
Started by user devops
Building in workspace /var/lib/jenkins/workspace/hello_dexter
[hello_dexter] $ /bin/sh -xe /tmp/jenkins4256604733931399277.sh
+ mkdir /tmp/primeiro
Finished: SUCCESS
```

294



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

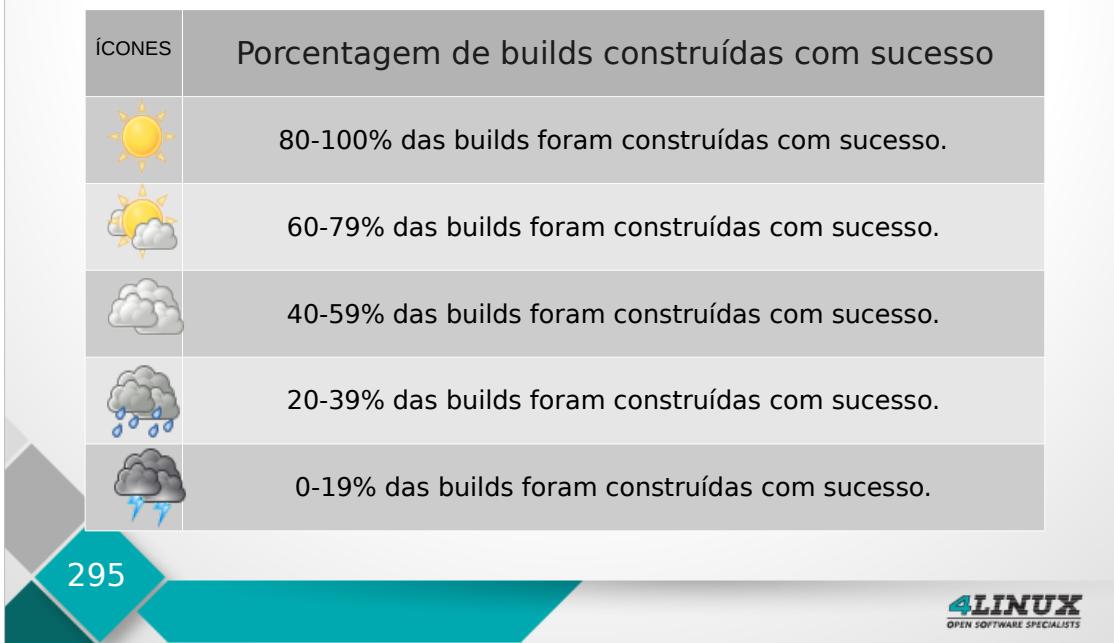
---

---

---

## Primeiros Passos com Jenkins

### Indicadores de Estabilidade



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

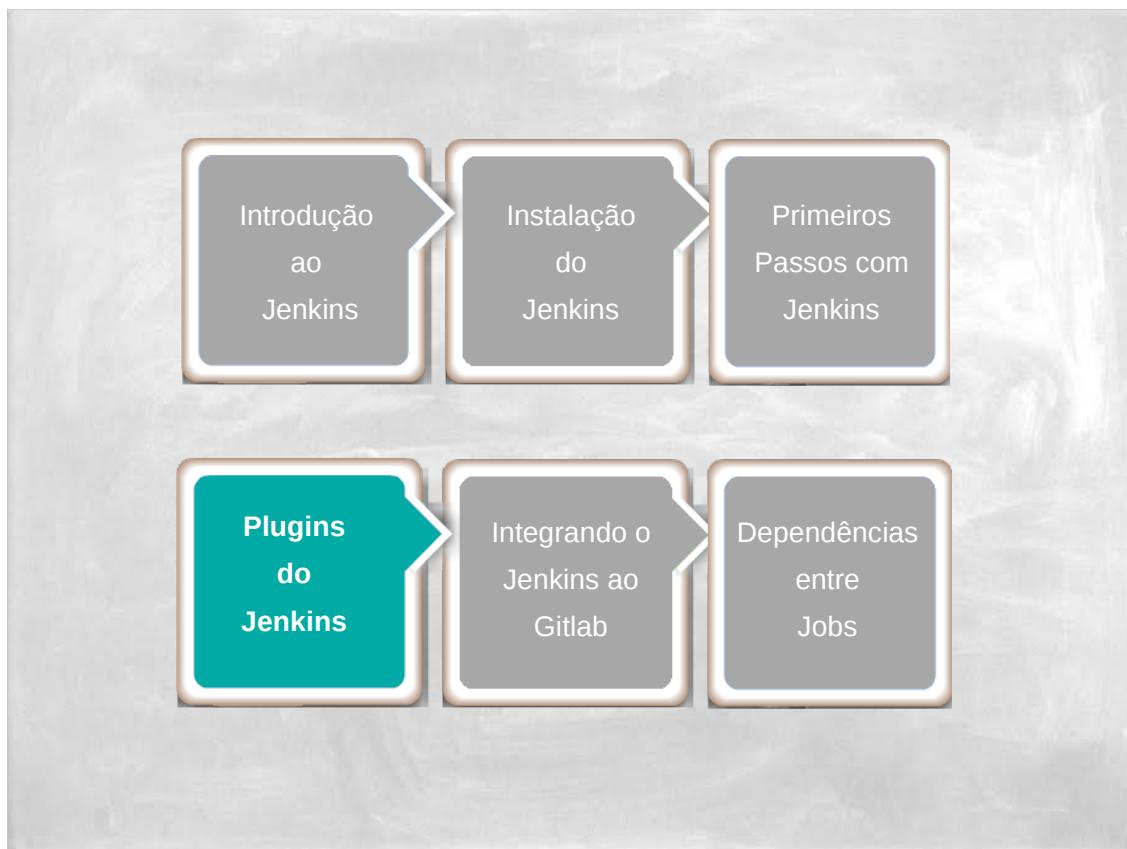
---

---

---

---

---



## Anotações:

## Plugins do Jenkins

**Plugins:** O Jenkins é um software extensível, o que significa que podemos facilmente adicionar novas funcionalidades para ele através dos Plugins.

Qualquer um pode desenvolver plugins para o Jenkins, o que o torna uma ferramenta com grandes possibilidades de configurações e alterações.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

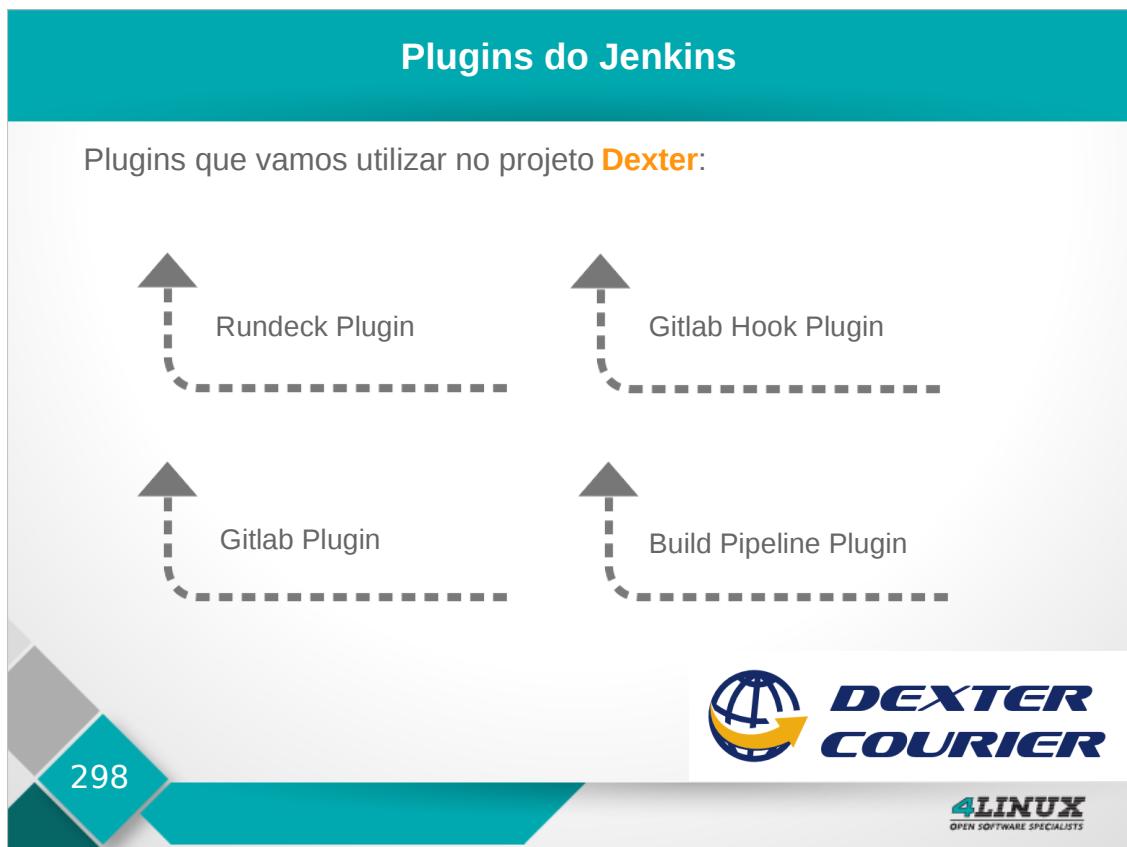
---

---

---

---

---



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Plugins do Jenkins

Clique no botão **Gerenciar Jenkins** para entrar nas páginas de configurações do Jenkins.



299

4LINUX  
OPEN SOFTWARE SPECIALISTS

## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

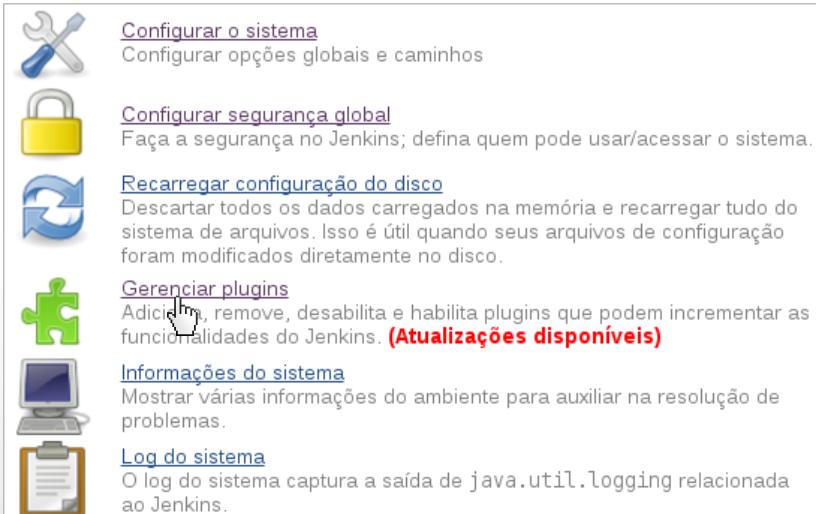
---

---

---

## Plugins do Jenkins

Procure a opção para **Gerenciar Plugins**.



The screenshot shows the Jenkins Global Configuration interface. On the left, there's a sidebar with icons for system configuration, security, disk reload, plugin management, system information, and system logs. The 'Manage Plugins' icon (a green puzzle piece) is highlighted with a red box and a cursor. The right panel contains descriptive text for each option.

- Configurar o sistema**  
Configurar opções globais e caminhos
- Configurar segurança global**  
Faça a segurança no Jenkins; defina quem pode usar/acessar o sistema.
- Recarregar configuração do disco**  
Descartar todos os dados carregados na memória e recarregar tudo do sistema de arquivos. Isso é útil quando seus arquivos de configuração foram modificados diretamente no disco.
- Gerenciar plugins**  
Adicione, remove, desabilita e habilita plugins que podem incrementar as funcionalidades do Jenkins. **(Atualizações disponíveis)**
- Informações do sistema**  
Mostrar várias informações do ambiente para auxiliar na resolução de problemas.
- Log do sistema**  
O log do sistema captura a saída de `java.util.logging` relacionada ao Jenkins.

300

4LINUX  
OPEN SOFTWARE SPECIALISTS

## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Plugins do Jenkins

Selecione a aba **Disponíveis** para instalar o plugins.

Atualizações Disponíveis Instalados Avançado

Instalar ↗ Nome

<input type="checkbox"/>	.NET Development <a href="#">CCM Plug-in</a> This plug-in generates reports on cyclomatic complexity
<input type="checkbox"/>	<a href="#">ExCop Runner plugin</a> FxCopCmd.exe execute plugin.
MSBuild Plugin	

301

4LINUX  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Plugins do Jenkins

Utilize o filtro para filtrar a palavra ansible e selecione o **Rundeck Plugin**.  
Em seguida, repita o processo para os plugins: Gitlab, Gitlab Hook e Build Pipeline: **Gitlab Plugin, Gitlab Hook Plugin e Build Pipeline Plugin**.

The screenshot shows the Jenkins Plugins page with a search bar at the top containing 'rundeck'. Below the search bar, there are tabs: Atualizações, Disponíveis (selected), Instalados, and Avançado. Under the 'Instalar' section, the 'Rundeck plugin' is listed. It includes a checkbox, a description: 'Jenkins plugin for Rundeck integration : trigger, notifier and option provider.', and its version: 3.6.1. Buttons below the table include 'Instalar sem reiniciar', 'Baixar agora, instalar e depois reiniciar', and 'Verificar agora'. A note says 'Update information obtained: 33 minutos ago'. The bottom right corner features the 4LINUX logo.

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

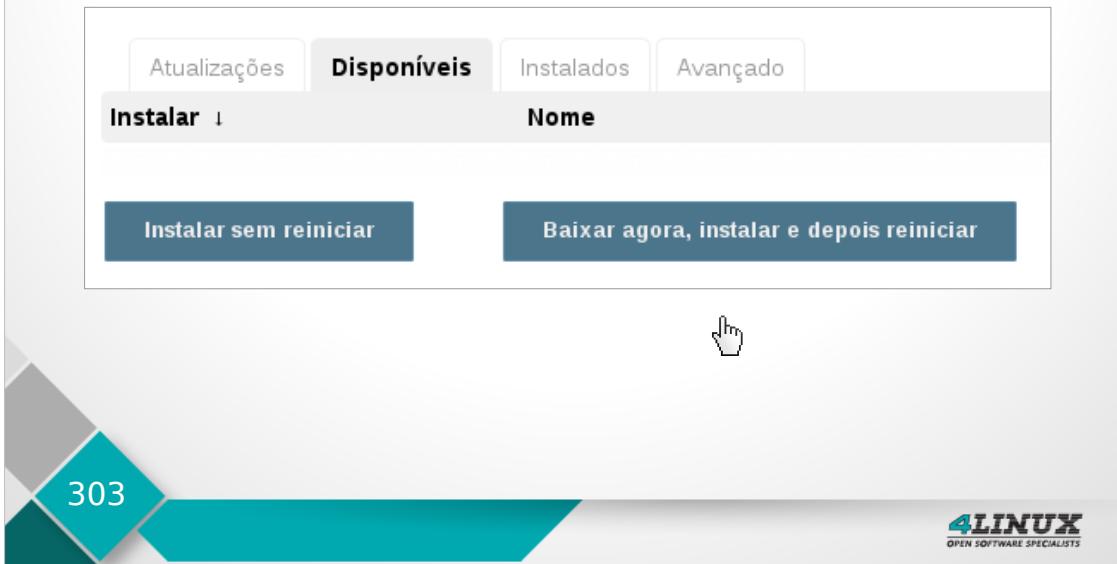
---

---

---

## Plugins do Jenkins

Após selecionar todos os plugins, clique no botão **Baixar Agora** e aguarde a instalação do mesmos.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Projeto Dexter

Concluímos a instalação do Jenkins com os plugins que utilizaremos no projeto Dexter.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

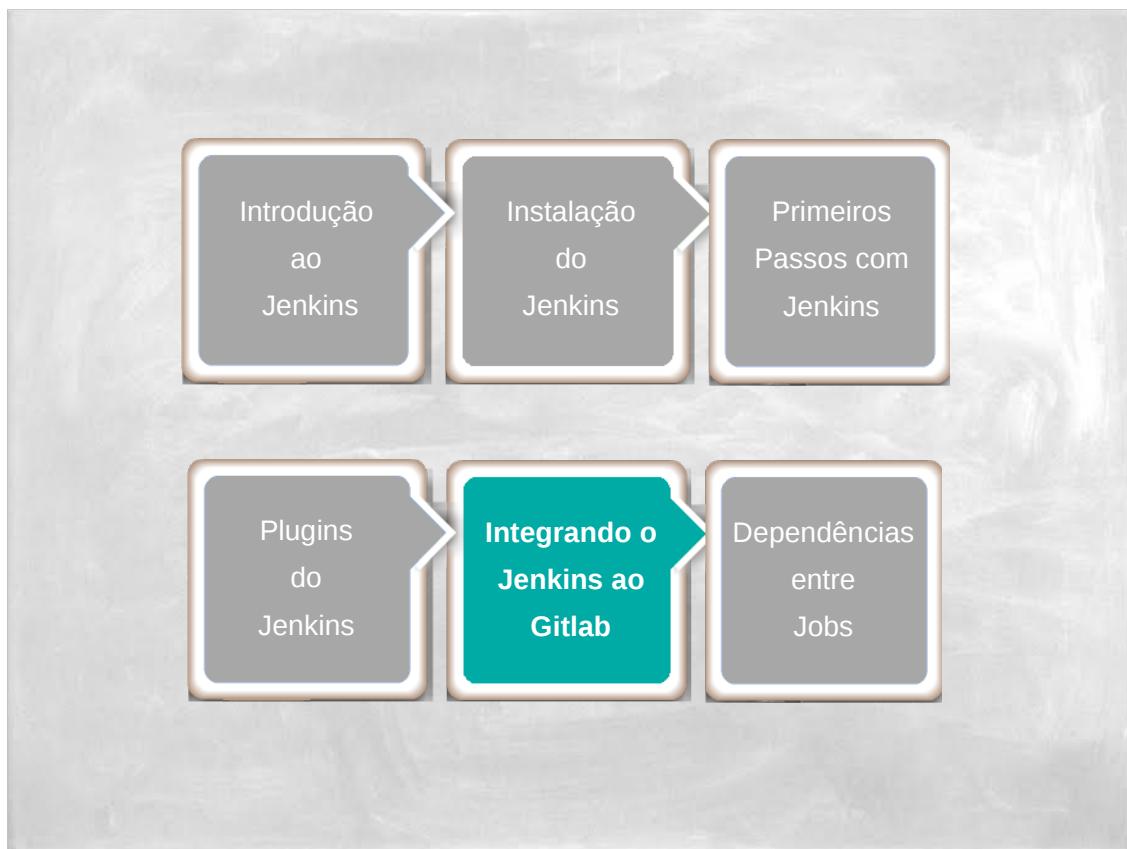
---

---

---

---

---



## Anotações:

## Integração do Jenkins ao Gitlab

Vamos configurar a nossa Job **Deploy aplicacao** para ser executada toda vez que alguém modificar o repositório do Gitlab.

Assim, teremos o Gitlab disparando uma trigger para o Jenkins avisando que houve mudanças no repositório.



306



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Integração do Jenkins ao Gitlab

Crie um repositório chamado **web** no nosso Gitlab.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Integração do Jenkins ao Gitlab

Servidor: Devops

Vamos inserir um conteúdo no repositório.

- 1 Faça o clone do repositório criado.

```
1# git clone git@devops.dexter.com.br:devops/web.git
```

```
2# cd web
```

- 2 Adicione o arquivo.

```
3# vim index.html
```

```
<p>Dexter Courier</p>
```

- 3 Salve as alterações e atualize o repositório.

```
4# git add --all
```

```
5# git commit -m "Criação do Projeto"
```

```
6# git push origin master
```

308



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Integração do Jenkins ao Gitlab

Vamos configurar uma credencial no Jenkins com a chave do SSH configurada no Gitlab.

Clique no link **Credentials** no menu da página inicial do Jenkins.



309

4LINUX  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

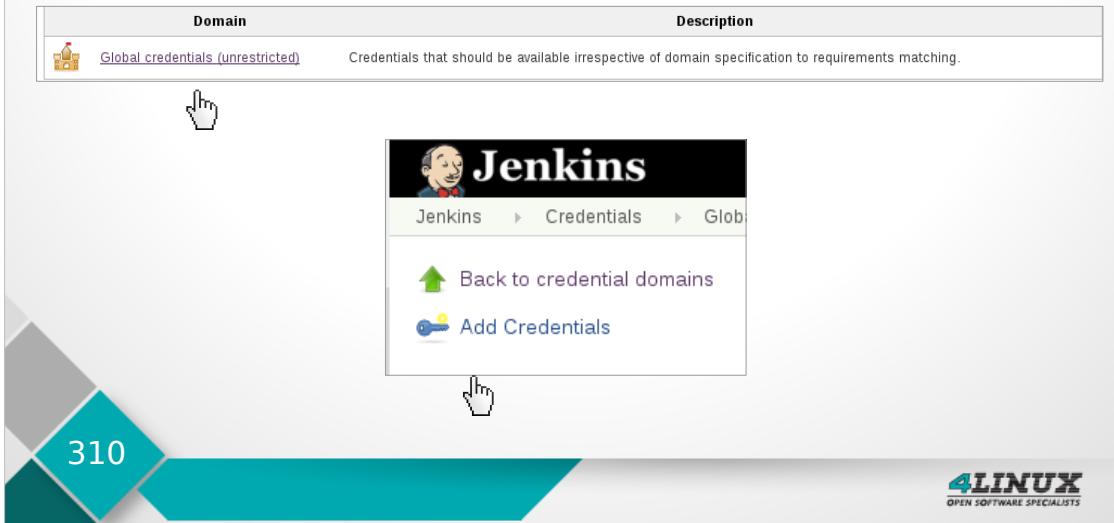
---

---

---

## Integração do Jenkins ao Gitlab

Clique em **Global Credentials (unrestricted)** para cadastrar a chave para qualquer projeto do Jenkins. Em seguida, clique em **Add Credentials** para adicionar uma nova credencial.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

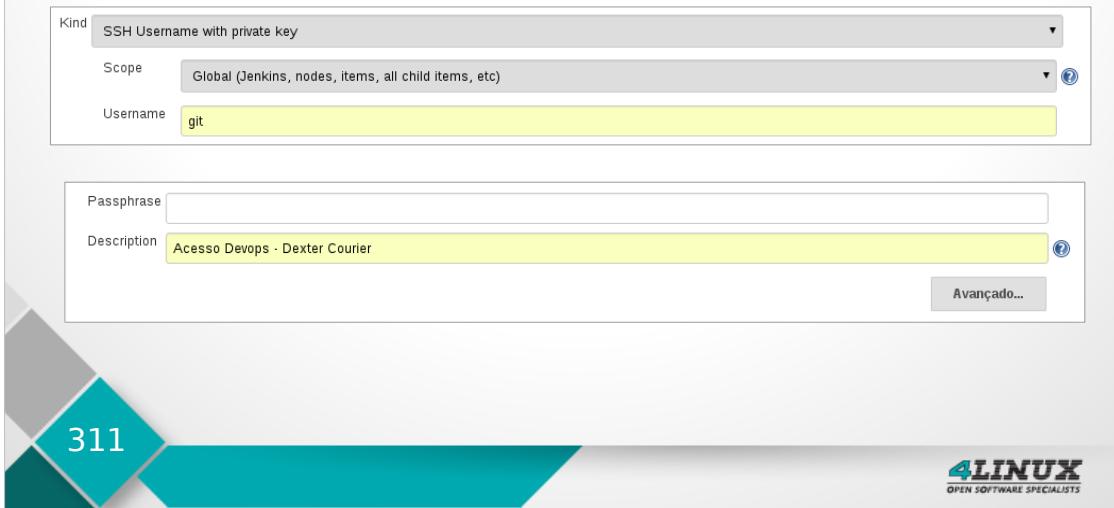
---

## Integração do Jenkins ao Gitlab

Escolha o tipo **SSH Username with private key** e defina as opções.

**Username:** git

**Description:** Acesso Devops – Dexter Courier



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Integração do Jenkins ao Gitlab

Servidor: Devops

Precisamos copiar a chave privada no servidor para inserir no formulário de cadastro da credencial.

1

Pelo terminar, mostre o conteúdo do arquivo utilizando o comando cat.

```
1# cat /etc/keys/key.pem
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAsHOpIpoNbNu7ZPf93ndMMAhQR84QffifWkcPJh9ZC7rkSPTc
AOTONTacicdu6cY3b9Jx5tnzBK5UNjHVKEmmRfov3KZpTyKHX4VCW1EM0r/JsxkJ
crVjdzwFThMemGDlN3nswgRkdIZ761zyhSE/gQ4b4hOT8XKZkAgQeE3r+b8Vyxm3
bowfBkb1GrNqrluPqjlgKv+86o86CuctgAf7RQngNZThfri+oeC1HDTGSHUG+YJJ
Flzh/KWLnk2CYFe34KYz9TEo+8tD4IHPNNh1jaFOqHanqXD89Or+3mG
...
...
```

312



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

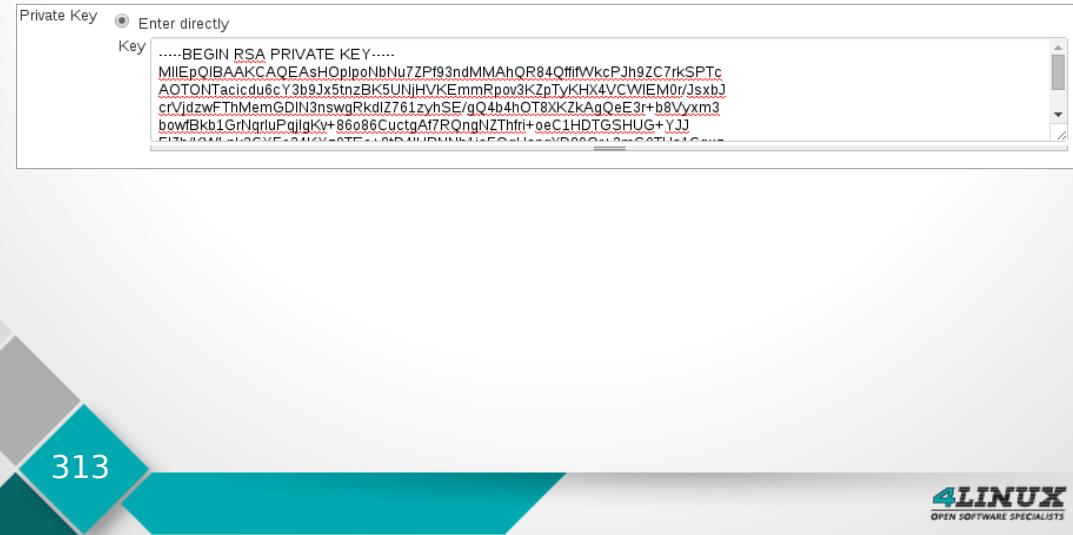
---

---

---

## Integração do Jenkins ao Gitlab

Selecione a opção **Enter Directly** e cole o conteúdo da chave privada dentro da entrada de texto.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Integração do Jenkins ao Gitlab

Clique em **OK** para finalizar a configuração da nossa Credencial.

Kind **SSH Username with private key**

Scope **Global (Jenkins, nodes, items, all child items, etc)**

Username **git**

Private Key  Enter directly  
**Key**  
`-----BEGIN RSA PRIVATE KEY-----  
MIIEpQIBAAKCAQEAsHOpIpohblu7ZPf93ndMMAhQR84OfifWkcPJh9ZC7rkSPTc  
AOTONTaciduc6cY3b9Jx5tnzBK5UjhHVKEmmRpov3KzpTyKH4VCWEIM0rJxbJ  
crVjdzwFThMencGDNjNswgRkdIz761zyhSE/gQ4b4hOT8XKZkAgQeE3r+b8Vyxm3  
bowfBkb1GrNqrUpqjgkV+86o86CuctgAf7RQnghZThfr+oeC1HDTSHGUG+YJJ  
-----END RSA PRIVATE KEY-----`

From a file on Jenkins master  
 From the Jenkins master ~/ssh

Passphrase

Description **Acesso Devops - Dexter Courier**

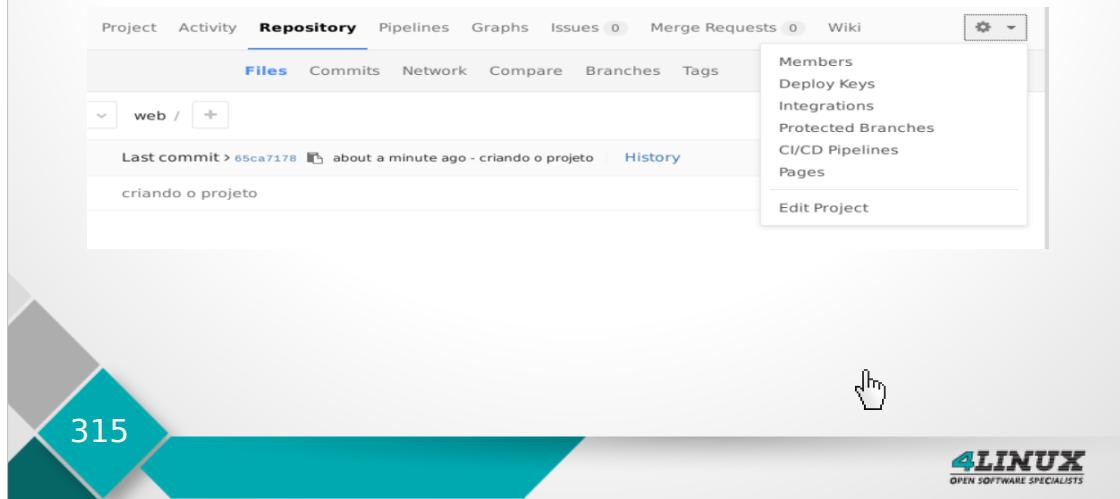
**Avançado...**



## Anotações:

## Integração do Jenkins ao Gitlab

Agora vamos ao Gitlab para configurar a **Web Hook** para as chamadas da Job do Jenkins. Clique em **integrations** na página de configurações de repositório do projeto **web**.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Integração do Jenkins ao Gitlab

Na página de Interações do Projeto, no campo **URL** vamos inserir a API do jenkins.

### Integrations

[Webhooks](#) can be used for binding events when something is happening within the project.

#### URL

`http://automation.dexter.com.br:8080/gitlab/build_now`

#### Secret Token

Use this token to validate received payloads. It will be sent with the request in the X-Gitlab-Token HTTP header.

#### Trigger

##### Push events

This URL will be triggered by a push to the repository



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

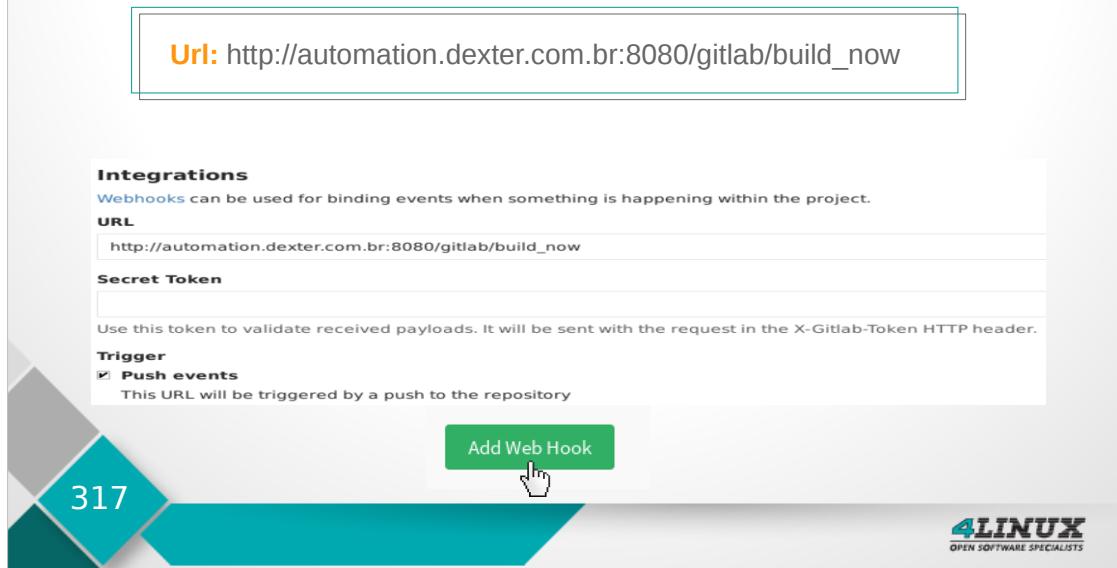
---

---

---

## Integração do Jenkins ao Gitlab

Insira a URL do plugin do Gitlab dentro do Jenkins e clique no botão **Add Web Hook**.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

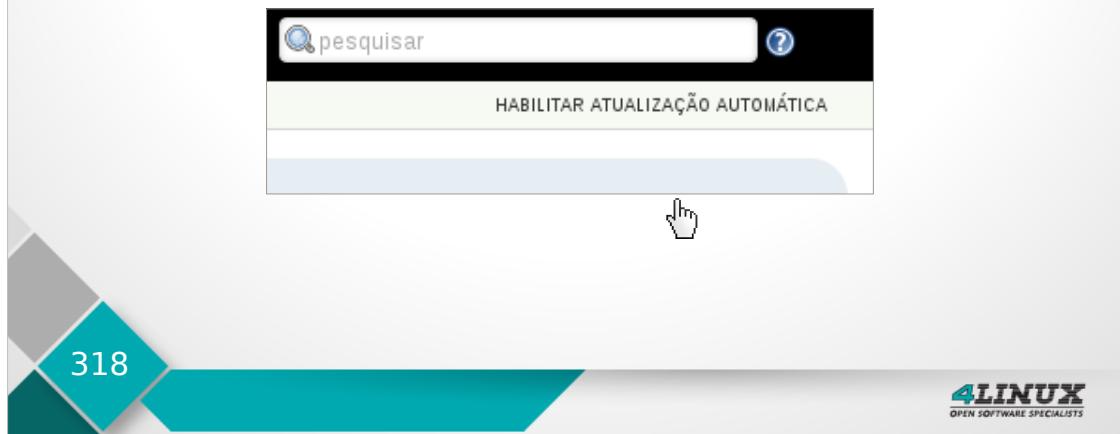
---

---

---

## Integração do Jenkins ao Gitlab

Podemos utilizar a função **Habilitar Atualização Automática**, assim o Jenkins atualizará a página automaticamente. Visualizaremos a Job sendo executada automaticamente sem precisar recarregar manualmente a página do Jenkins.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

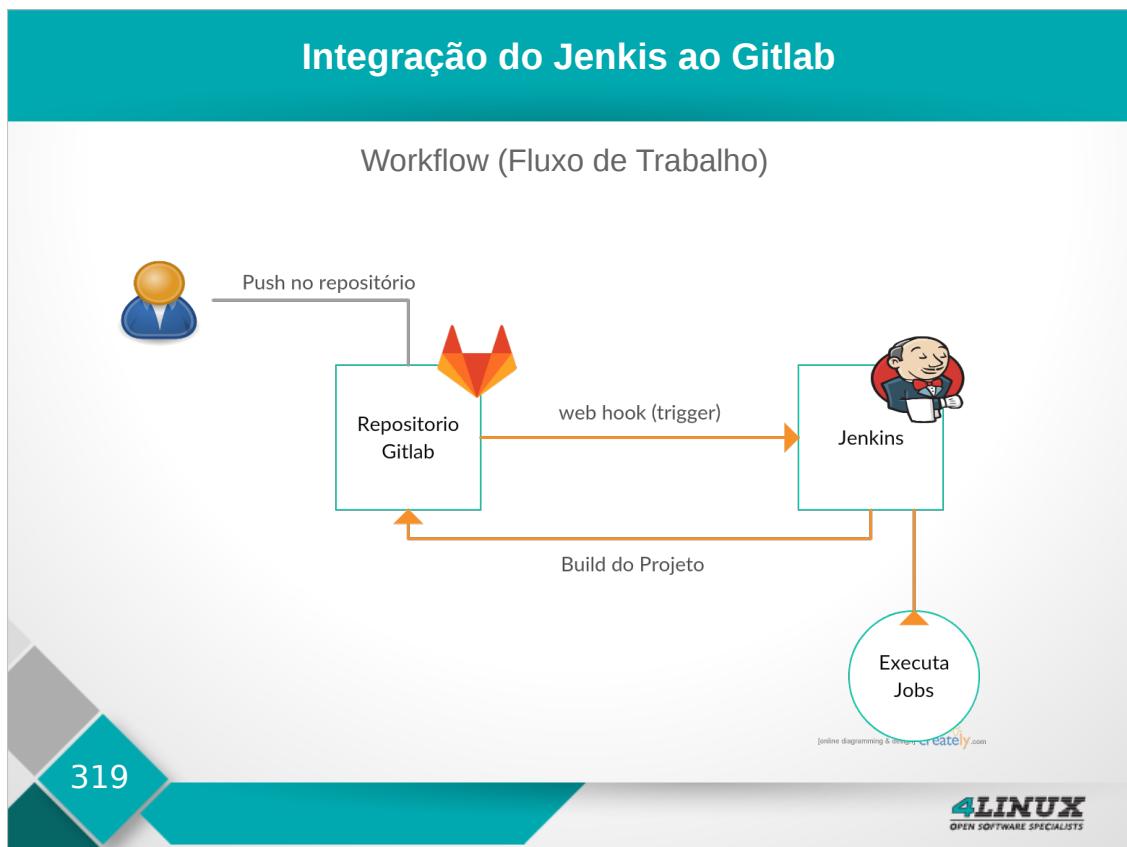
---

---

---

---

---



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

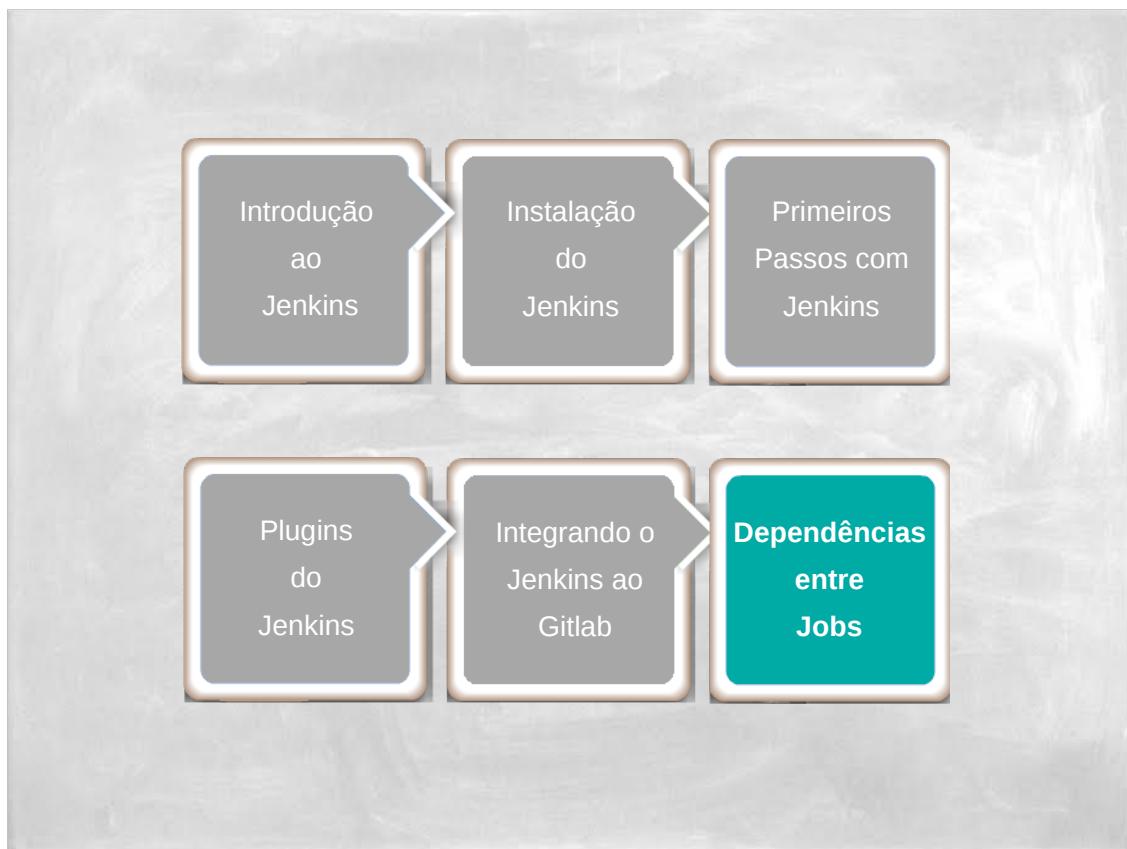
---

---

---

---

---



## Anotações:

## Dependências entre Jobs

No **Projeto Dexter** teremos uma Job que será responsável por executar uma job do rundeck.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Dependências entre Jobs

Para que o Jenkins acione o rundeck no pós-build, é necessário que exista uma instância do rundeck configurada. Na Página inicial do jenkins, vamos clicar em “**Manage Jenkins**” e depois “**Configure System**”.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Dependências entre Jobs

Nessa página, podemos gerenciar todas as configurações do jenkins, nós vamos adicionar uma nova instância do Rundeck em “**Add Rundeck**”.

**Rundeck**

Job cache  Enable Rundeck job cache  
Rundeck job cache configuration

Instances **Add Rundeck**

List of Rundeck instances:  
2

# of executors

Labels

Usage Use this node as much as possible

Quiet period 5

SCM checkout retry count 0

Restrict project naming

323

4LINUX  
OPEN SOFTWARE SPECIALISTS

## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Dependências entre Jobs

Nessa etapa, passaremos as opções de autenticação do jenkins ao rundeck, em seguida podemos clicar em “Salvar”.

324

Add Rundeck

List of Rundeck Instances

Name	Value
Name	rundeck
URL	<a href="http://automation.dexter.com">http://automation.dexter.com</a>
Login	admin
Password	*****
Auth Token	
API Version	

[Test Connection](#)

[Delete Rundeck](#)

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

## Anotações:

## Dependências entre Jobs

No **Projeto Dexter** teremos apenas uma instância do Rundeck. Agora que já configuramos, podemos criar a nossa Job.



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Dependências entre Jobs

Vamos criar uma Job e adicionar um passo Pós-Build.

Nome: **Deploy aplicacao;**

Selecionar a opção: **Construir Outros Projetos.**

Enter an item name

Deploy aplicacao

» Required field

**Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your code and run tests. It can even be used for something other than software development.

**Pipeline**  
Orchestrates long-running activities that can span multiple stages (known as workflows) and/or organizing complex activities.

**External Job**  
This type of job allows you to record the execution of a pipeline or script. This is designed so that you can use Jenkins as a dashboard.

**Multi-configuration project**  
Suitable for projects that need a large number of different configurations (e.g., specific builds, etc.)

OK

Folder

326

4LINUX  
OPEN SOFTWARE SPECIALISTS

## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Dependências entre Jobs

Na nossa Job, no campo “**Source Code Management**”, faremos integração com o gitlab.

The screenshot shows the Jenkins job configuration interface for the 'Source Code Management' section. It is set to 'Git'. A repository is defined with the URL 'git@devops.4linux.com.br:devops/web.git' and credentials 'git'. The 'Branch Specifier' is set to '\*/\*master'. The 'Repository browser' is configured to '(Auto)'. At the bottom, there are 'Save', 'Apply', and 'Add' buttons. A watermark for '4LINUX OPEN SOFTWARE SPECIALISTS' is present in the bottom right corner. A large teal arrow graphic with the number '327' is overlaid on the left side of the screenshot.

## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Dependências entre Jobs

Vamos criar uma Job e adicionar um passo Pós-Build.

Nome: **Deploy aplicacao;**

Selecionar a opção: **Construir Outros Projetos.**

Enter an item name  
Deploy aplicacao  
» Required field

---

**Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your code and run tests. It can be even used for something other than software development.

**Pipeline**  
Orchestrates long-running activities that can span multiple stages (known as workflows) and/or organizing complex activities.

**External Job**  
This type of job allows you to record the execution of a pipeline or script. This is designed so that you can use Jenkins as a dashboard for external systems.

**Multi-configuration project**  
Suitable for projects that need a large number of different configurations (informations) for specific builds, etc.

**OK**  Folder



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

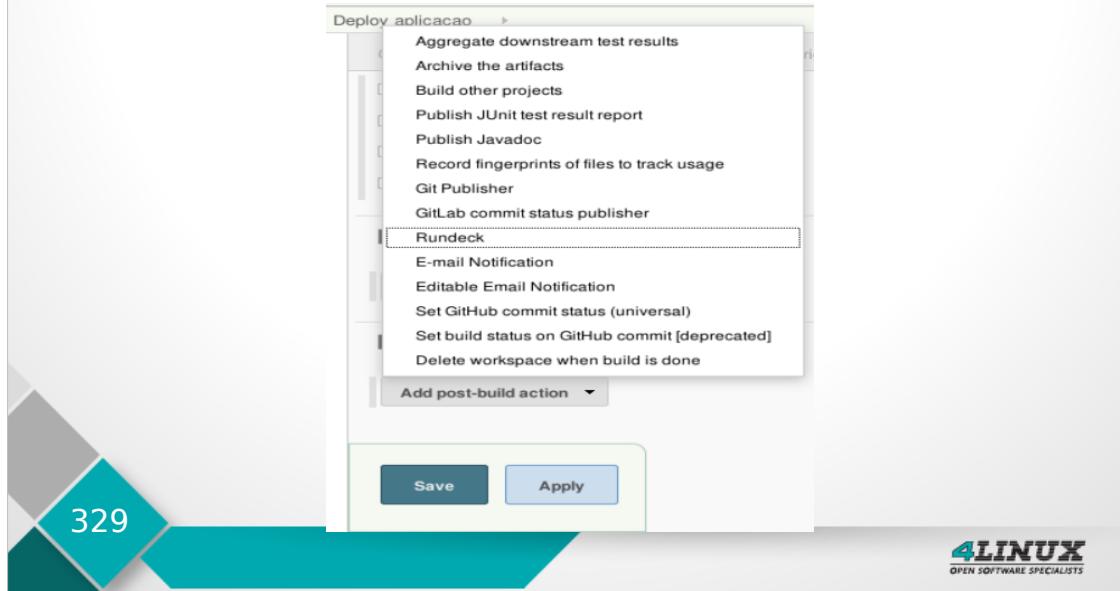
---

---

---

## Dependências entre Jobs

Após adicionar o “**Source Code Management**”, faremos um pós-build que chamará a job do Rundeck criada anteriormente.



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

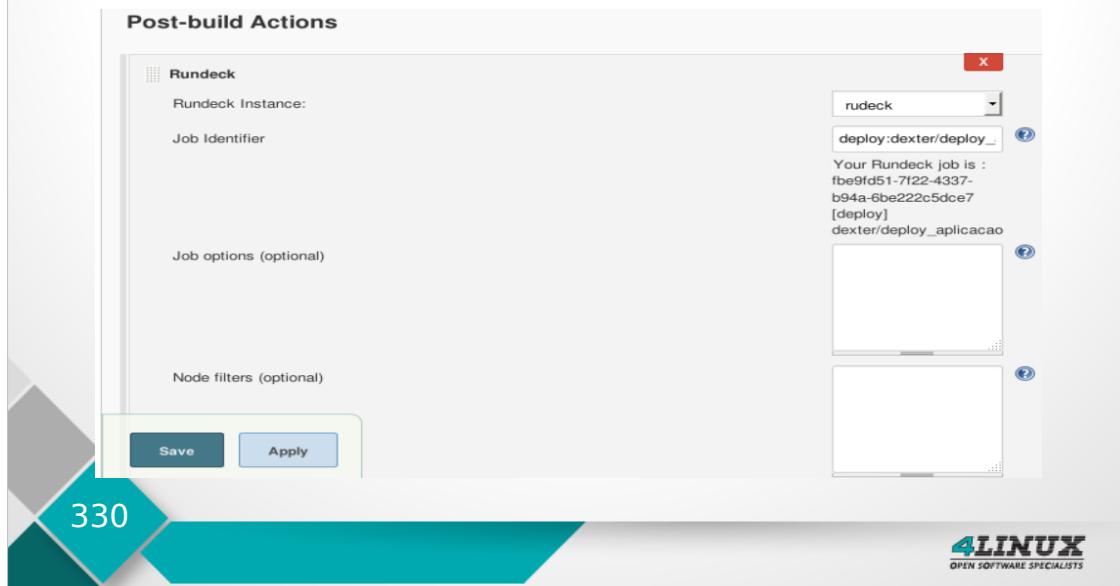
---

---

---

## Dependências entre Jobs

No pós-build, vamos especificar a instância do Rundeck que havíamos configurado e informar “**PROJETO:GRUPO/JOB**”.



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Projeto Dexter

No laboratório final **Dexter**, todos os nossos repositórios serão integrados com as Jobs do Jenkins para cada serviço.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Laboratório Dexter – Parte 04: Automação do Site Dexter

### Objetivos da Aula

- Ajustes do Nginx;
- Apresentação do Laboratório;
- Topologia Nginx como proxy\_pass.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

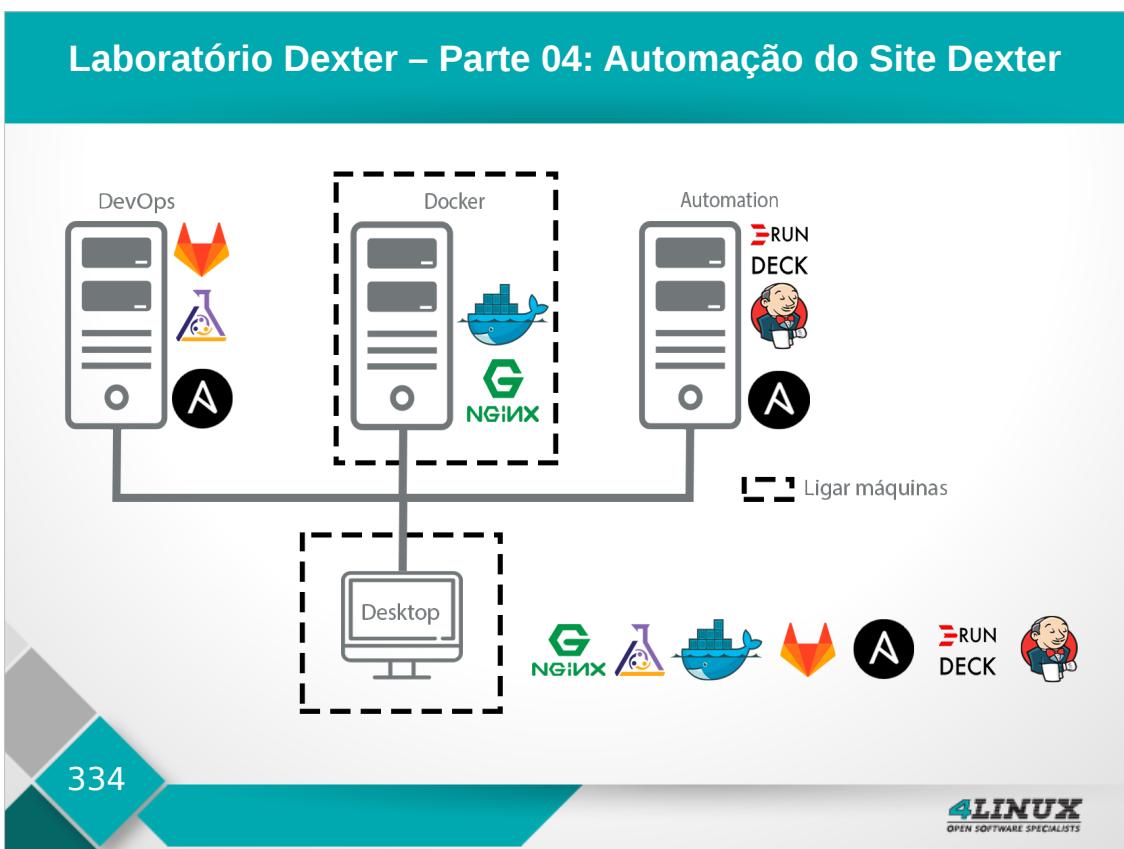
---

---

---

---

---



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Laboratório Dexter – Automação do Site Dexter



Vamos instalar o **Nginx** na máquina docker para fazer o papel de **proxy\_pass**, e toda requisição que chegar no host ele encaminhará para o container.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

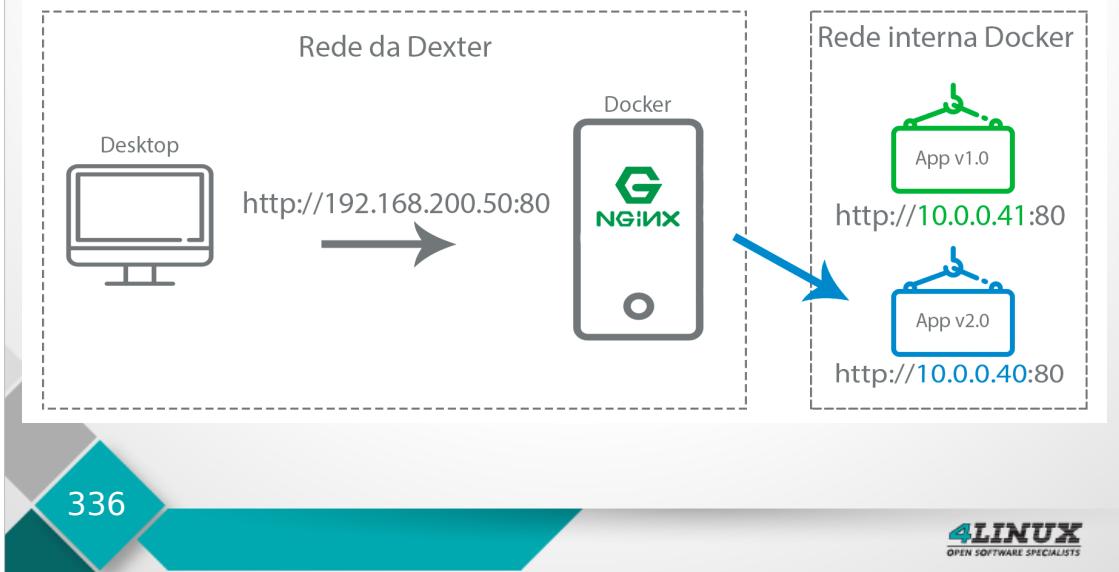
---

---

---

---

## Laboratório Dexter – Automação do Site Dexter



4LINUX  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Laboratório Dexter – Automação do Site Dexter

### Preparação do Ambiente



- Endereços Fixos para os containers de produção;
- Proxy com Nginx precisa compartilhar a porta 80 da máquina Docker Host, com o container que estiver em produção;
- Os nodes com a aplicação da Dexter (Página Web) ficarão atrás do proxy nginx;
- O Rundeck precisa saber qual é o próximo container a ser feito deploy.

337



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Preparação do Ambiente

Máquina:

- 1 Vamos instalar o Nginx.

```
1# apt-get update  
2# apt-get install nginx -y
```

- 2 Agora vamos mover o arquivo padrão para o diretório home.

```
4# cd /etc/nginx/sites-enabled/  
5# mv default /home/default.dist
```

338



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Preparação do Ambiente

Máquina:

3

Na máquina Automation, vamos criar as roles que serão executadas pelo Rundeck.

```
1# cd /etc/ansible/roles/  
2# ansible-galaxy init create  
3# ansible-galaxy init log  
4# ansible-galaxy init verificar
```

4

Vamos criar os arquivos de configuração padrões do Nginx.

```
1# vim verificar/files/green.conf  
server {  
    listen 80;  
    location / {  
        proxy_pass http://10.0.0.41/web/;  
    }  
}
```

339



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Preparação do Ambiente

Máquina:

5

Vamos criar os arquivos de configuração padrões do Nginx.

```
1# vim verificar/files/blue.conf  
  
server {  
  
    listen 80;  
  
    location / {  
  
        proxy_pass http://10.0.0.42/web/;  
  
    }  
  
}
```

340



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Laboratório Dexter – Automação do Site Dexter

O arquivo de configuração irá variar e será copiado para o diretório “`/etc/nginx/sites-enabled`”, através do playbook que será chamado pelo rundeck.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Criando Playbooks de Deploy Blue Green

### Objetivos da Aula

- Criação dos Playbooks de Deploy;
- Apresentação do Fluxo de Trabalho;
- Desenvolvimento do Laboratório.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

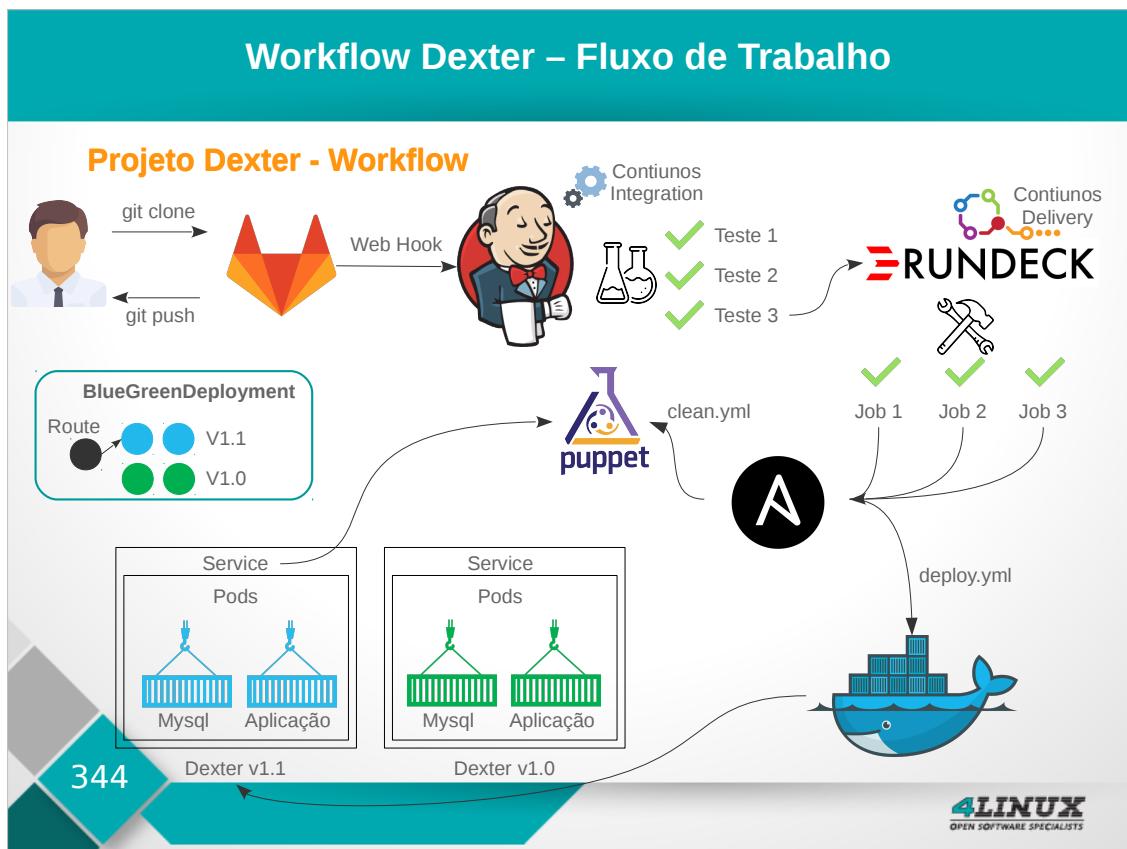
---

---

---

---

---



Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Trabalhando com Roles

1

Dentro do diretório de roles na máquina Automation, editaremos os arquivos das tasks.

```
1# vim /etc/ansible/roles/verificar/tasks/main.yml  
2# vim /etc/ansible/roles/log/tasks/main.yml  
3# vim /etc/ansible/roles/create/tasks/main.yml
```

345



O nosso playbook vai “verificar” no Nginx qual container está em produção atualmente, e trocará o arquivo de configuração do container que será construído.

O próximo container entrará em produção apenas após o restart do serviço.

```
1# vim /etc/ansible/roles/verificar/tasks/main.yml
- name: Validando o Virtual Host
  command: test -f /etc/nginx/sites-enabled/green.conf
  register: vhost
  ignore_errors: yes

- name: Remover green env
  command: rm -f /etc/nginx/sites-enabled/green.conf
  when: vhost.rc == 0

- name: Deploy blue env
  copy:
    src: /etc/ansible/roles/verificar/files/blue.conf
    dest: /etc/nginx/sites-enabled/blue.conf
  when: vhost.rc == 0

- name: Remove blue env
  command: rm -f /etc/nginx/sites-enabled/blue.conf
  when: vhost.rc == 1

- name: Deploy green env
  copy:
    src: /etc/ansible/roles/verificar/files/green.conf
    dest: /etc/nginx/sites-enabled/green.conf
  when: vhost.rc == 1
```

---

```
=====
2# vim /etc/ansible/roles/log/tasks/main.yml
- name: Validando o vhost
  command: test -f /etc/nginx/sites-enabled/green.conf
  register: vhost
  ignore_errors: yes

- name: Gerando log caso seja blue
  shell: echo "blue" > /tmp/log
  when: vhost.rc == 0
```

```
3# vim create/tasks/main.yml

- name: Verificando ambiente
  command: cat /tmp/log
  register: name

- name: Verificar a existencia de um container
  command: docker inspect {{ name.stdout }}
  register: container
  ignore_errors: yes

- name: Remover o container caso exista
  command: docker rm -f {{ name.stdout }}
  when: container.rc == 0
  ignore_errors: yes

- name: Realizar a criacao de um container do docker caso seja green
  command: docker run -dit --name {{ name.stdout }}
    --net=dexterlan
    --ip 10.0.0.42
    --add-host devops.dexter.com.br:192.168.200.100
    deploy /bin/bash
  when: container.rc == 0 and name.stdout == "green"

- name: realizar a criacao de um container do docker caso seja blue
  command: docker run -dit --name {{ name.stdout }}
    --net=dexterlan
    --ip 10.0.0.41
    --add-host devops.dexter.com.br:192.168.200.100
    deploy /bin/bash
  when: container.rc == 0 and name.stdout == "blue"

- name: habilitar o puppet agent no container
  command: docker exec {{ name.stdout }} puppet agent --enable

- name: rodar o puppet agente no container
  command: docker exec {{ name.stdout }} puppet agent -t --server devops.dexter.com.br
  name: Remover o pacote default do apache2
```

```
- name: Baixar a nova versao do codigo
  command: "docker exec {{ name.stdout }}git clone
    git@devops.dexter.com.br:devops/web.git /var/www/web"

- name: restart apache
  command: docker exec {{ name.stdout }} service apache2 restart

- name: restart nginx
  command: service nginx restart
```

## Laboratório Dexter – Automação do Apache e Nginx



Agora, temos os playbooks que o rundeck irá executar toda vez que for acionado pelo jenkins.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Laboratório Dexter – Parte 04: Automação do Site Dexter

### Objetivos da Aula

- Criar Ambientes de Produção;
- Apresentação do Laboratório;
- Desenvolvimento do laboratório.



**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

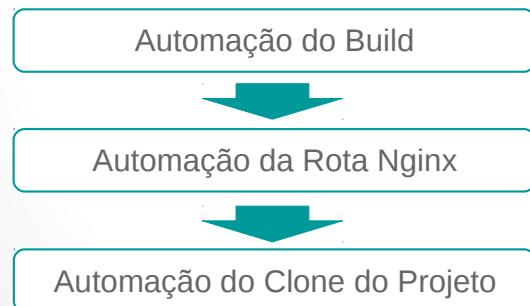
---

---

---

Laboratório Dexter – Automação do Site Dexter

## Etapas do Laboratório:



## Anotações:

## Laboratório Dexter – Automação do Site Dexter



Agora vamos automatizar a Infraestrutura da Dexter Courier.

Nessa parte do laboratório, criaremos a integração de todas as nossas ferramentas, proporcionando o build automático da nossa aplicação.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

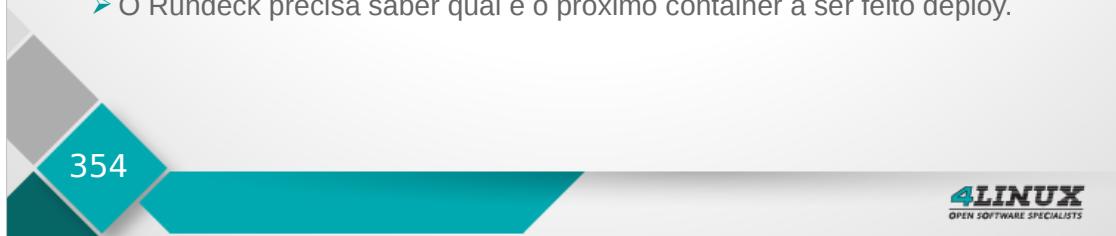
---

## Laboratório Dexter – Automação do Site Dexter

### Preparação do Ambiente:



- Endereços Fixos para os containers de Produção;
- Proxy com Nginx precisa compartilhar a porta 80 da máquina Docker Host, com o container que estiver em produção;
- Os nodes com a aplicação da Dexter (Página Web) ficarão atrás do proxy nginx;
- O Rundeck precisa saber qual é o próximo container a ser feito deploy.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Laboratório Dexter – Automação do Site Dexter

Gitlab terá a seguintes tarefas no laboratório:



- Armazenar o código fonte da Aplicação;
- Ativar o Jenkins quando à mudanças no Gitlab.

Mãos à obra!



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Laboratório Dexter – Automação do Site Dexter

Jenkins terá a seguintes tarefas no laboratório:



- Responsável por realizar testes no código(caso haja);
- Responsável por executar o pós-build que chamará o Rundeck para fazer deploy da aplicação.

Mãos à obra!



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Laboratório Dexter – Automação do Site Dexter

Ansible terá as seguintes tarefas no laboratório:



- Verificar qual container está em produção no momento, elegendo o próximo a ser feito deploy;
- Destruir o container que não está em produção, e recriá-lo com a nova versão da aplicação.

Mãos à obra!



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

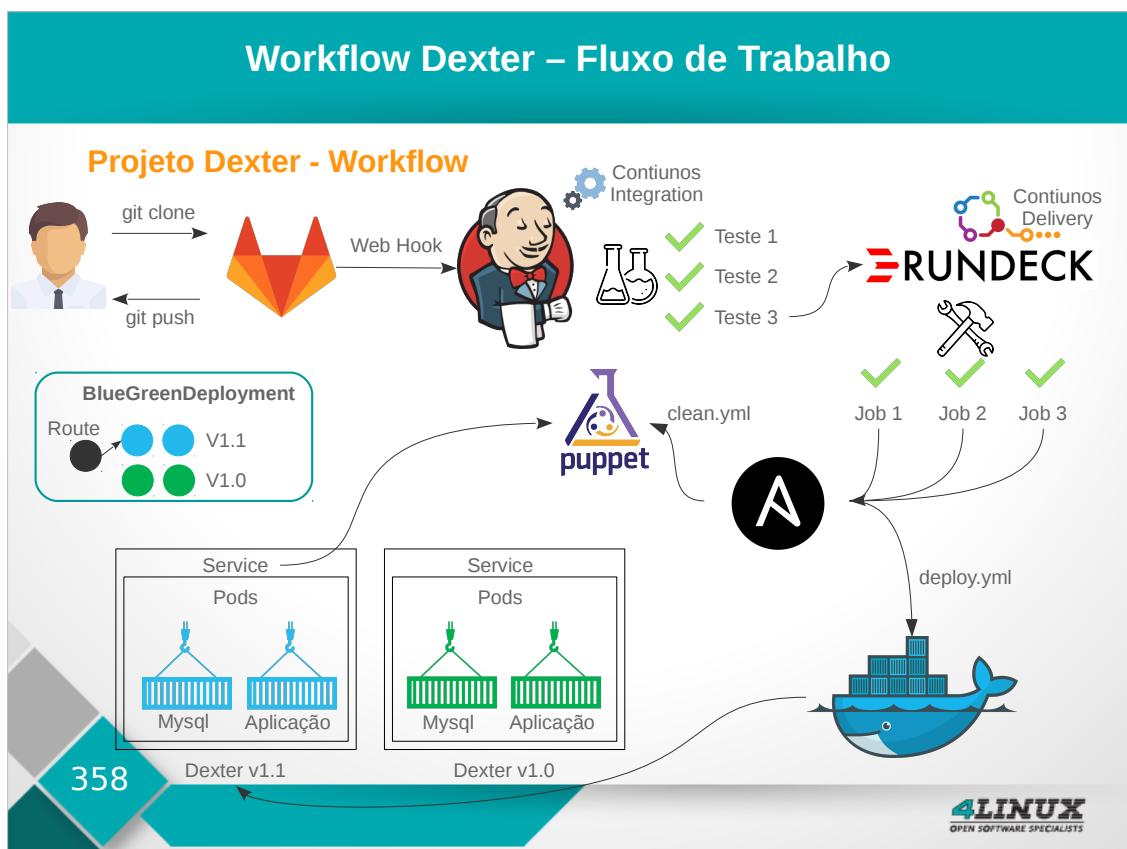
---

---

---

---

---



Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Laboratório Dexter – Automação do Site Dexter

A ultima parte do Laboratório está completa, agora a Dexter possui o seu ambiente totalmente automatizado, uma Infraestrutura Ágil com práticas do Mundo DevOps.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Infraestrutura Cloud AWS

### Objetivos da Aula

- Introduzir a Cloud Computing;
- Conhecer a Amazon Web Services;
- Criar uma conta na AWS;
- Acessar o AWS Management Console;
- Construir a Infraestrutura AWS;
- Configurar o VPC;
- Configurar o EC2;
- Configurar o ELB;
- Configurar o S3;
- Configurar o RDS;
- Configurar o ElasticCache.

361



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

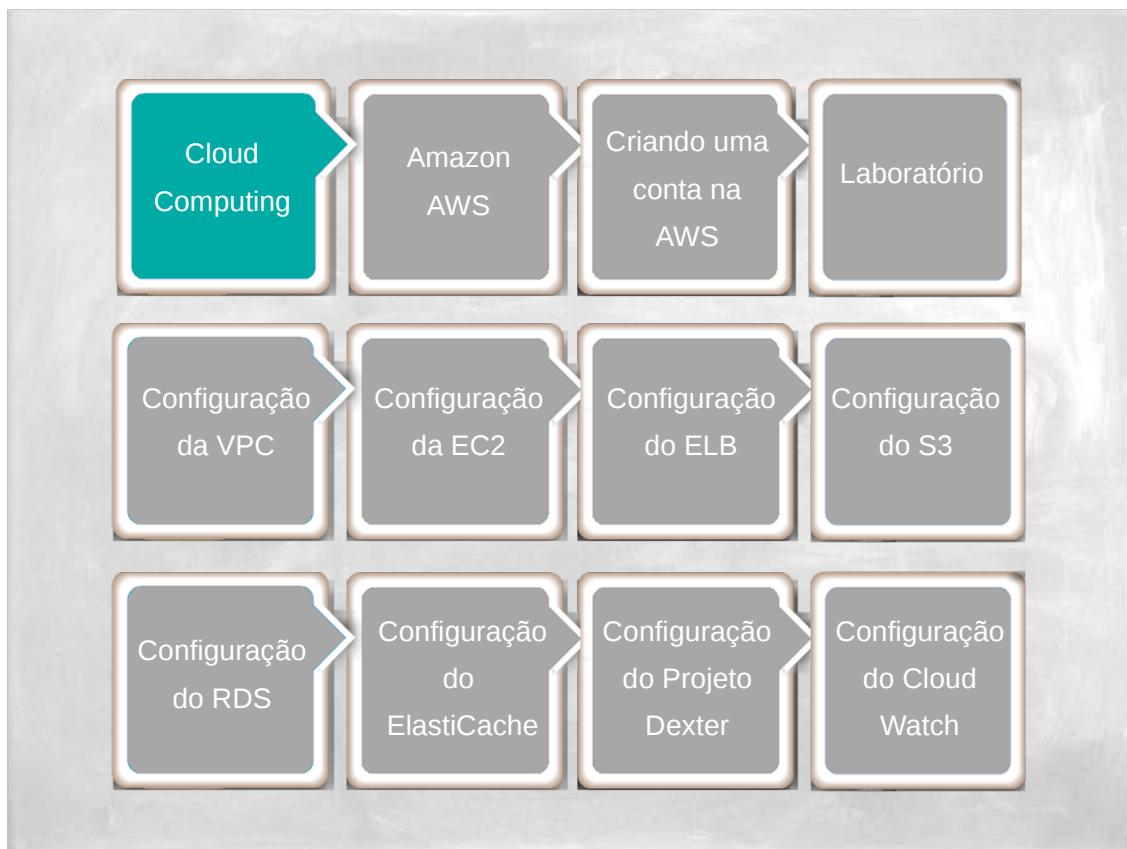
---

---

---

---

---



## Anotações:

## Introdução ao Cloud Computing

**Cloud Computing** ou Computação em Nuvem é um formato computacional, no qual aplicativos, dados e recursos de TI são disponibilizados aos usuários como serviço, por meio da Internet, e pagos de acordo com a utilização dos recursos disponibilizados.

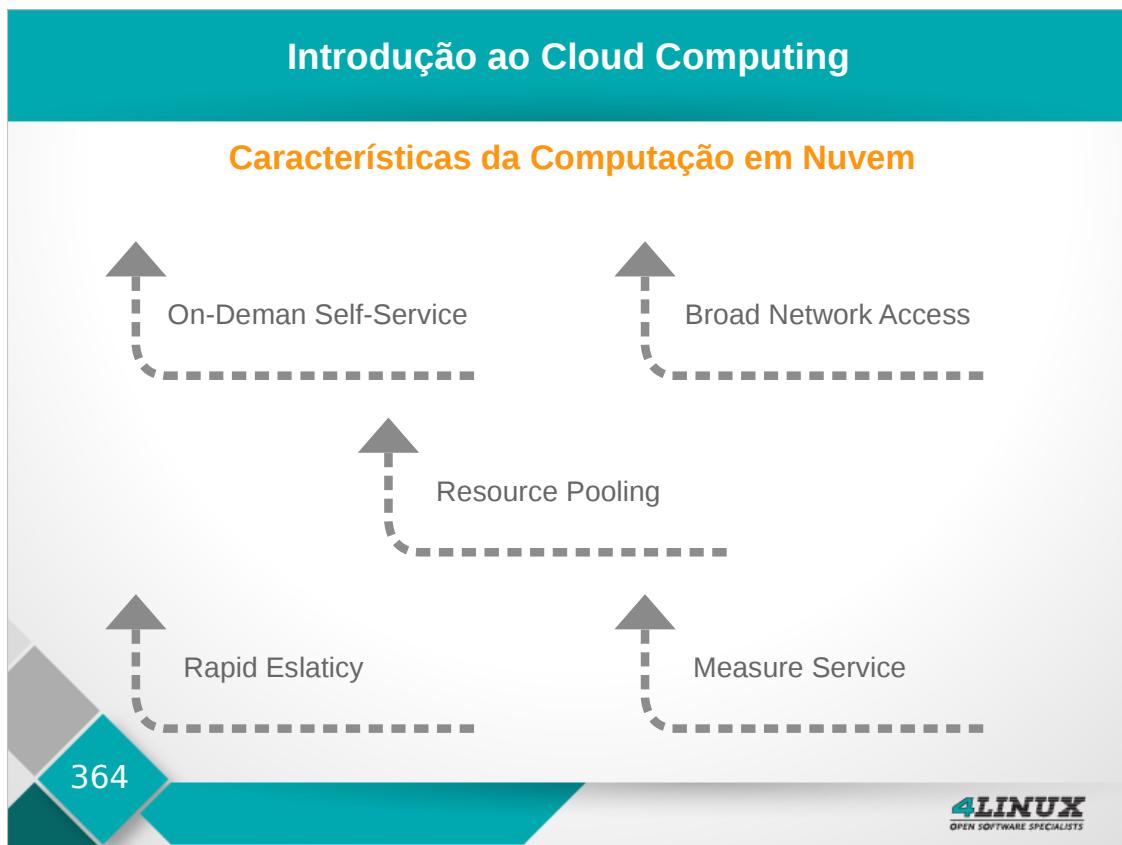


363

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

Computação nas nuvens refere-se à possibilidade de acessar arquivos e executar diferentes tarefas pela internet. Ou seja, você não precisa instalar aplicativos no seu computador para tudo, pois pode acessar diferentes serviços online para fazer o que precisa, já que os dados não se encontram em um computador específico, mas sim em uma rede.

Uma vez devidamente conectado ao serviço online, é possível desfrutar suas ferramentas e salvar todo o trabalho que for feito para acessá-lo depois de qualquer lugar. É justamente por isso que o seu computador estará nas nuvens, pois você poderá acessar os aplicativos a partir de qualquer computador que tenha acesso à internet.



**On-demand Self-Service:** O cliente pode usar os serviços da nuvem e, caso achar necessário, aumentar ou diminuir as capacidades computacionais alocadas, armazenamento de dados, uptime do servidor, tudo isso sem interação humana com o provedor de serviços utilizando API ou softwares.

**Broad Network Access:** Ter um amplo acesso à rede significa que os serviços de nuvem são acessíveis de qualquer plataforma. São utilizadas várias tecnologias e metodologias que possuem uma interoperabilidade do sistema. Assim, o cliente pode acessar os serviços de um smartphone, desktop, laptop ou qualquer outro dispositivo.

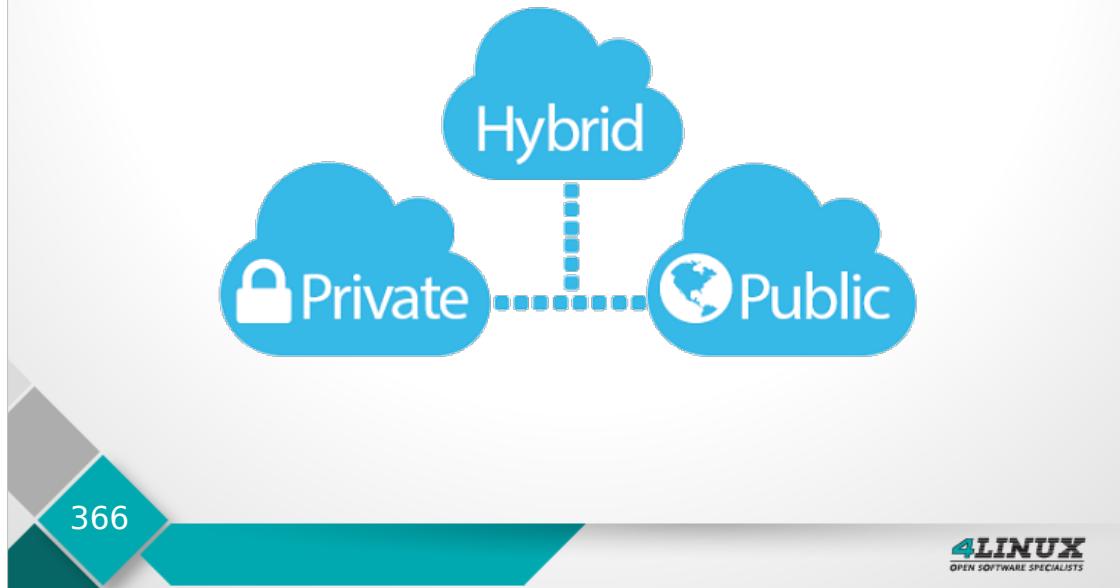
**Resource Pooling:** Os recursos computacionais da nuvem ficam reunidos geograficamente. Seus recursos virtuais são dinamicamente atribuídos ou reatribuídos para que o cliente confirme a sua demanda. O cliente não possui controle sobre a real localização dos recursos que serão utilizados, tendo somente uma informação ampla, como por exemplo, o país em que se encontra o Datacenter. Os tipos de recursos são: armazenamento, processamento, memória, banda e máquina virtuais. Até mesmo nuvens privadas tendem a reunir seus recursos entre partes das organizações.

**Rapid Elasticy:** Elasticidade é definida como a capacidade de alocar mais ou menos recursos no momento em que for necessário, com agilidade. Na ótica do consumidor, a nuvem parece ser infinita, pois ele pode adquirir mais ou menos poder computacional quando for necessário para suas aplicações. Essa é uma das principais características que torna a Cloud Computing um serviço muito atrativo.

**Measured Service:** Todos os serviços são controlados e monitorados automaticamente pela nuvem, de maneira que fica tudo transparente tanto para o consumidor quanto para o fornecedor. Isso ajuda o consumidor a otimizar sua utilização da nuvem de acordo com sua produção e ajuda o provedor na hora da cobrança dos recursos.

## Introdução ao Cloud Computing

**Tipos de Cloud:** Pública, Privada e Híbrida.



### Public Cloud (Nuvem Pública)

Formato de cloud em que qualquer usuário conectado à internet pode utilizar e comprar seus recursos. Esse hardware normalmente é compartilhado entre os clientes o que, em alguns casos, pode gerar problemas de segurança.

### Private Cloud (Nuvem Privada)

Formato de cloud em que apenas uma empresa ou grupo é responsável por utilizar e, na grande maioria dos casos, gerenciar. Os recursos dedicados são de uso dos funcionários ou parceiros de negócios da empresa.

### Community Cloud (Nuvem Comunitária)

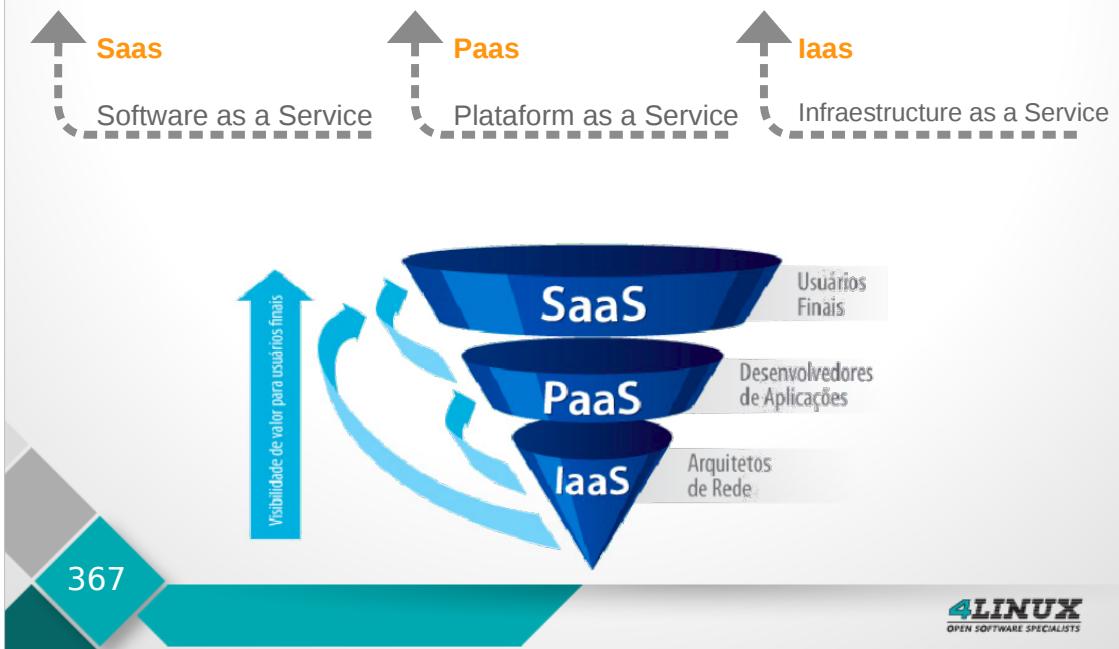
Cloud híbrida que é compartilhada por várias organizações que partilham interesses como a missão, requisitos de segurança, políticas, entre outros. É como uma rede de hospitais, por exemplo. Pode ser administrada pelas próprias empresas ou por terceiros.

### Hybrid Cloud (Nuvem Híbrida)

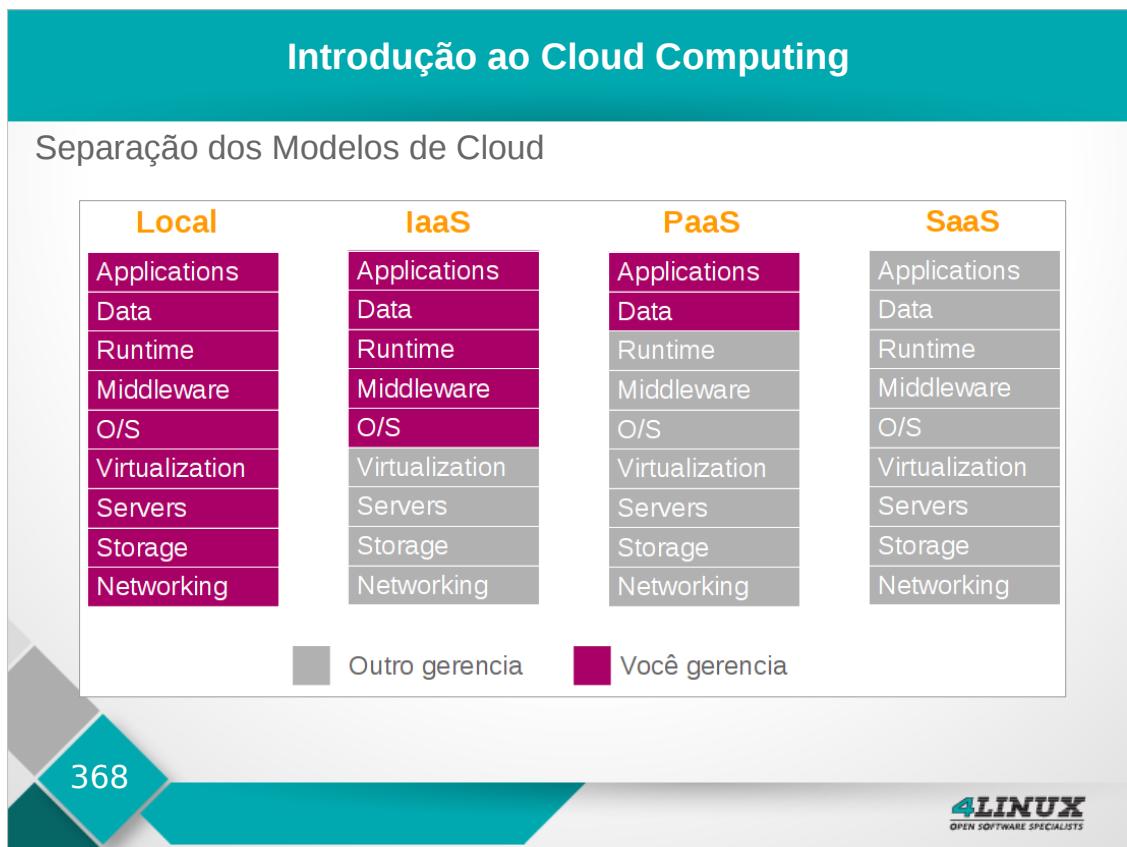
Cloud Hibrida ou Nuvem Híbrida é uma composição de duas ou mais nuvens, sejam elas privadas, públicas ou comunitárias.

## Introdução ao Cloud Computing

## Modelos de Cloud



## Anotações:



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

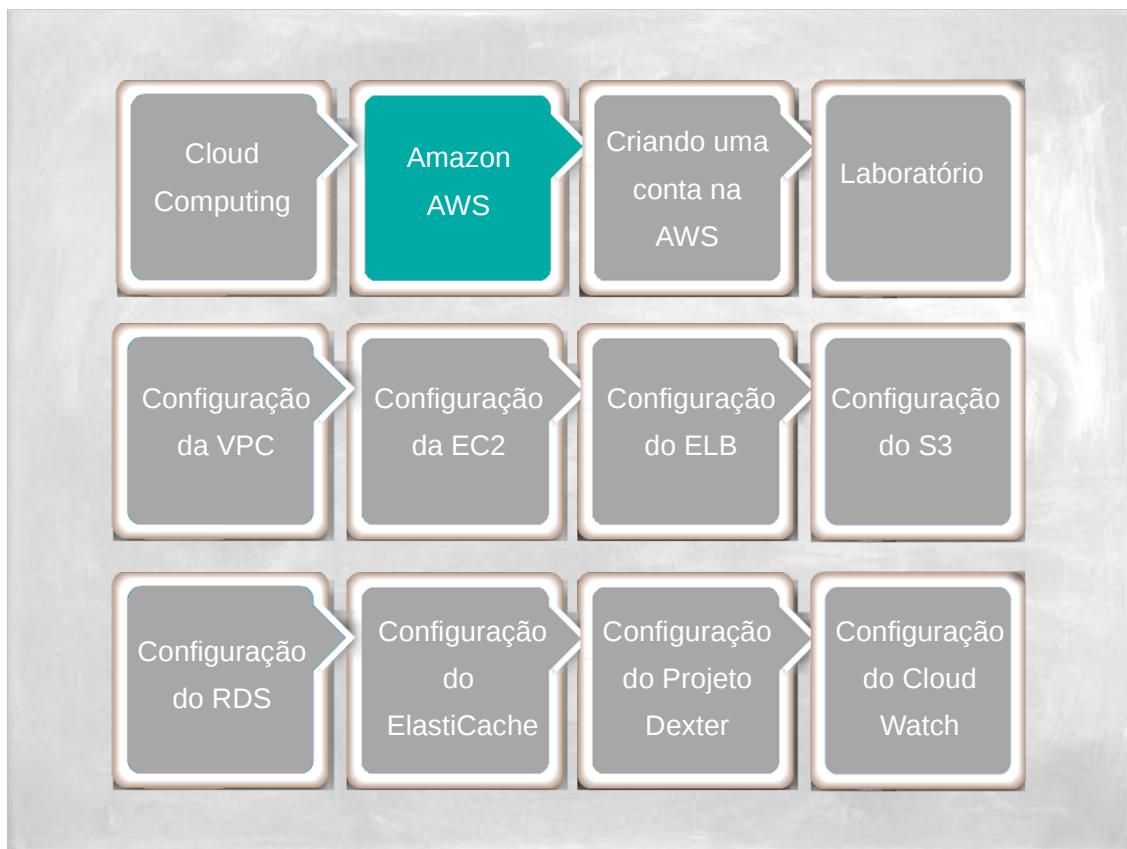
---

---

---

---

---



## Anotações:

## Amazon Web Services

**AWS (Amazon Web Services)** é uma coleção de serviços em nuvem, também conhecido como web services, e oferecidos pela Amazon.



A empresa possui serviços operando em 12 regiões diferentes espalhadas pelo mundo.



**4LINUX**  
OPEN SOFTWARE SPECIALISTS

## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Amazon Web Services



**AMI (Amazon Machine Image)** é um Serviço de imagens utilizado para a criação de máquina (EC2) no ambiente Cloud da Amazon.

É uma appliance virtual que serve como base para o sistema operacional que irá ser usada em instâncias EC2.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Amazon Web Services



**EC2 (Elastic Cloud Compute)** é um serviço web que disponibiliza máquinas virtuais (Instâncias EC2) no ambiente da Amazon.

As instâncias do EC2 são criados a partir das AMI's.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Amazon Web Services



**VPC (Virtual Private Cloud)** é um serviço que disponibiliza um ambiente lógico privado dos Serviços da Amazon.

Utilizando VPC você pode criar uma segurança a nível lógico e gerenciar redes e subredes das instâncias dos serviços.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Amazon Web Services



**S3 (Simple Storage Service)** é um serviço que disponibiliza armazenamento de objetos de forma segura e escalável.

O S3 pode ser usado para armazenar basicamente qualquer coisa.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Amazon Web Services



**ELC (Elastic Cache)** é um serviço que disponibiliza o armazenamento em cache de memória escalável.

Você pode escolher entre dois engines, uma instância do Memcache ou do Redis Server.



**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Amazon Web Services



**RDS (Relational Database Service)** é um serviço que disponibiliza instâncias de Banco de Dados, de forma escalável, backup e Failover automático.

Disponibiliza instâncias de PostgreSQL, MariaDB, MySQL, entre outros bancos.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Amazon Web Services



**Cloud Formation** é um serviço da Amazon que oferece a capacidade para provisionar serviços de forma fácil através de arquivos de configurações.

Você escreve um arquivo no formato JSON, descrevendo os serviços e os parâmetros de configurações e provisiona o ambiente.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Amazon Web Services



**Lambda** é um serviço que permite a você executar códigos sem a necessidade de gerenciar ou criar servidores.

Você pode configurar o lambda para executar códigos a partir de eventos disparados por outros serviços da Amazon.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Amazon Web Services



**Cloud Watch** é um serviço de monitoramento para recursos dos serviços da AWS e os aplicativos que você executa na AWS.

Podemos utilizar o Cloud Watch para armazenar métricas, coletar e monitorar arquivos de log, definir alarmes e agir automaticamente a alterações.

379



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

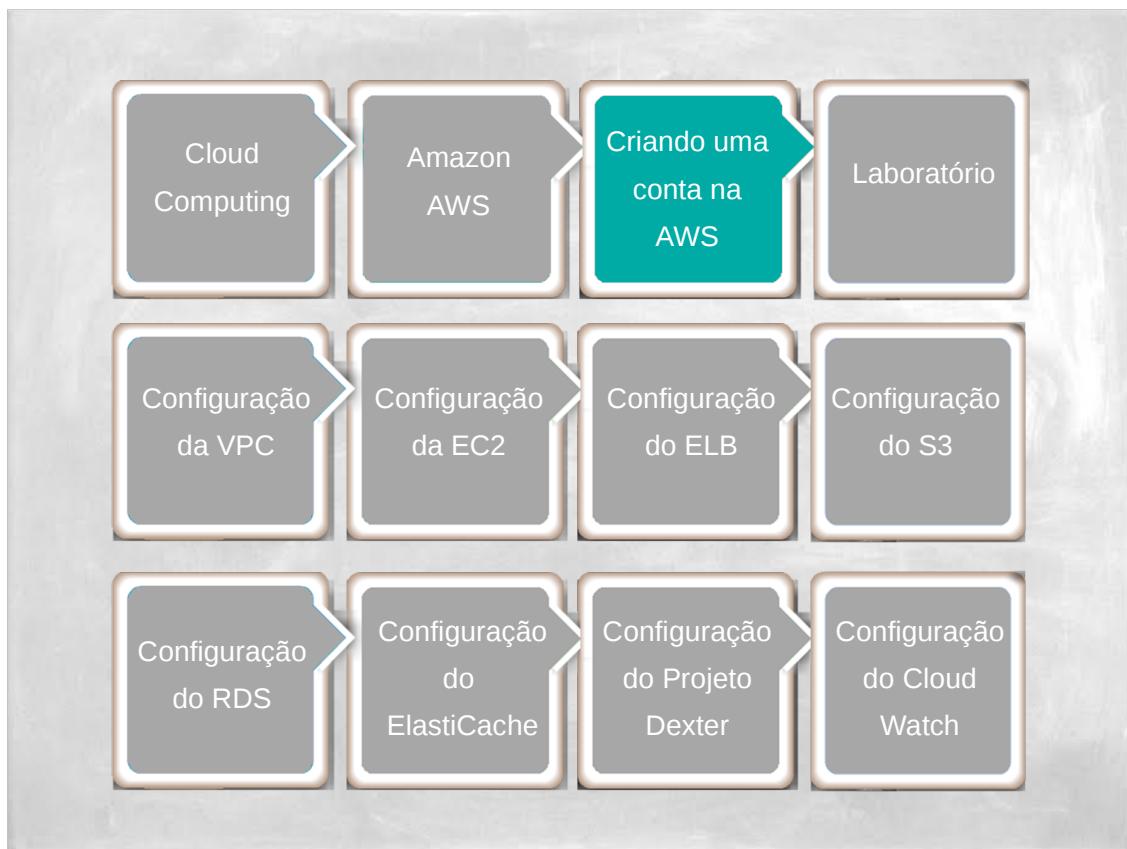
---

---

---

---

---



## Anotações:

## Criando uma Conta na AWS

### Nível Gratuito da AWS:

Serviços disponibilizados gratuitamente por um ano.



#### Nível gratuito da AWS

O nível gratuito da Amazon Web Services (AWS) foi concebido para permitir que você obtenha experiência prática com os serviços da Nuvem AWS. O nível gratuito da AWS inclui serviços com um nível gratuito disponível por 12 meses após a data do seu [cadastro](#) na AWS, além de ofertas de serviço adicionais que não expiram automaticamente ao final do período de 12 meses do nível gratuito da AWS.

Depois de criar a sua conta da AWS, você poderá usar gratuitamente qualquer um dos produtos e serviços listados a seguir, dentro de determinados limites de utilização.

Você pode começar hoje e aproveitar automaticamente o nível gratuito da AWS seguindo as etapas abaixo:

1. [Cadastre-se para abrir uma conta da AWS.](#)
2. Insira o endereço de faturamento e as informações do cartão de crédito. Você não será cobrado, a menos que ultrapasse os níveis de uso gratuito.
3. Comece a usar os serviços em nuvem da AWS escolhendo qualquer um dos produtos a seguir.

<http://aws.amazon.com/pt/free/>



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Criando uma Conta na AWS



### Nível Gratuito EC2:

- **750 Horas** por mês de uso de instância t2.micro Linux RHEL.  
(1 Instância/mês ou 2 Instâncias-duas semanas)
- Expira em 12 meses após o cadastro.



**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Criando uma Conta na AWS



### Nível Gratuito Elastic Load Balancer:

- **750 Horas** por mês.
- **15 GB** de processamento de dados.
- Expira em 12 meses após o cadastro.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Criando uma Conta na AWS



### Nível Gratuito S3:

- **5 GB** de armazenamento padrão.
- **20.000** requisições GET.
- **2.000** requisições PUT.
- Expira em 12 meses após o cadastro.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

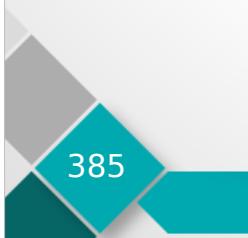
---

## Criando uma Conta na AWS



### Nível Gratuito RDS:

- **750 horas** de uso de instância db.t2.micro Single-AZ.
- **20 GB** de armazenamento de banco de dados: qualquer combinação de propósito geral (SSD) ou magnético.
- **10.000.000 de E/S.**
- Expira em 12 meses após o cadastro.



**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

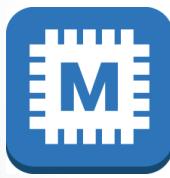
---

---

---

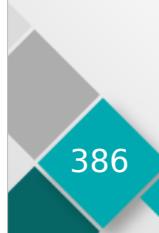
---

## Criando uma Conta na AWS



### Nível Gratuito ElastiCache:

- **750 horas** de uso de instância cache.t2.micro, suficiente para execução contínua a cada mês.
- Expira em 12 meses após o cadastro



**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Criando uma Conta na AWS



### Nível Gratuito CloudWatch:

- **10 métricas** personalizadas e **10 alertas**.
- **1.000.000** de requisição de API.
- **5 GB** de ingestão de dados de log e **5 GB** de arquivos de dados de log.
- **3 painéis** com até **50 métricas** por mês para cada um deles.
- Não expira ao final do período de 12 meses do nível gratuito.



**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



## Anotações:

## Projeto Dexter

A **Dexter** está lançando o sistema de Gerenciamento de Projeto da Dexter Courier.

Esse sistema web precisa ter disponibilidade de 24 horas por dia. Vamos utilizar os serviços da Amazon para armazenar a Infraestrutura deste portal.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

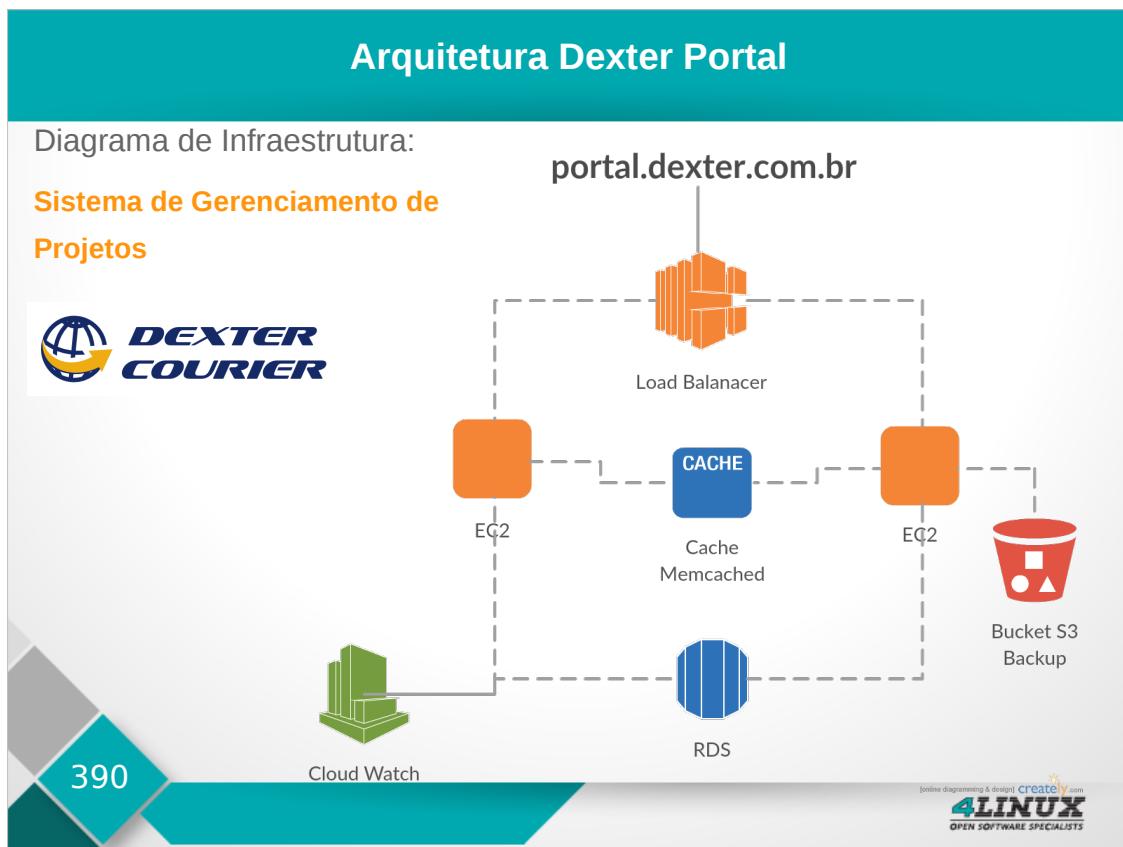
---

---

---

---

---



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

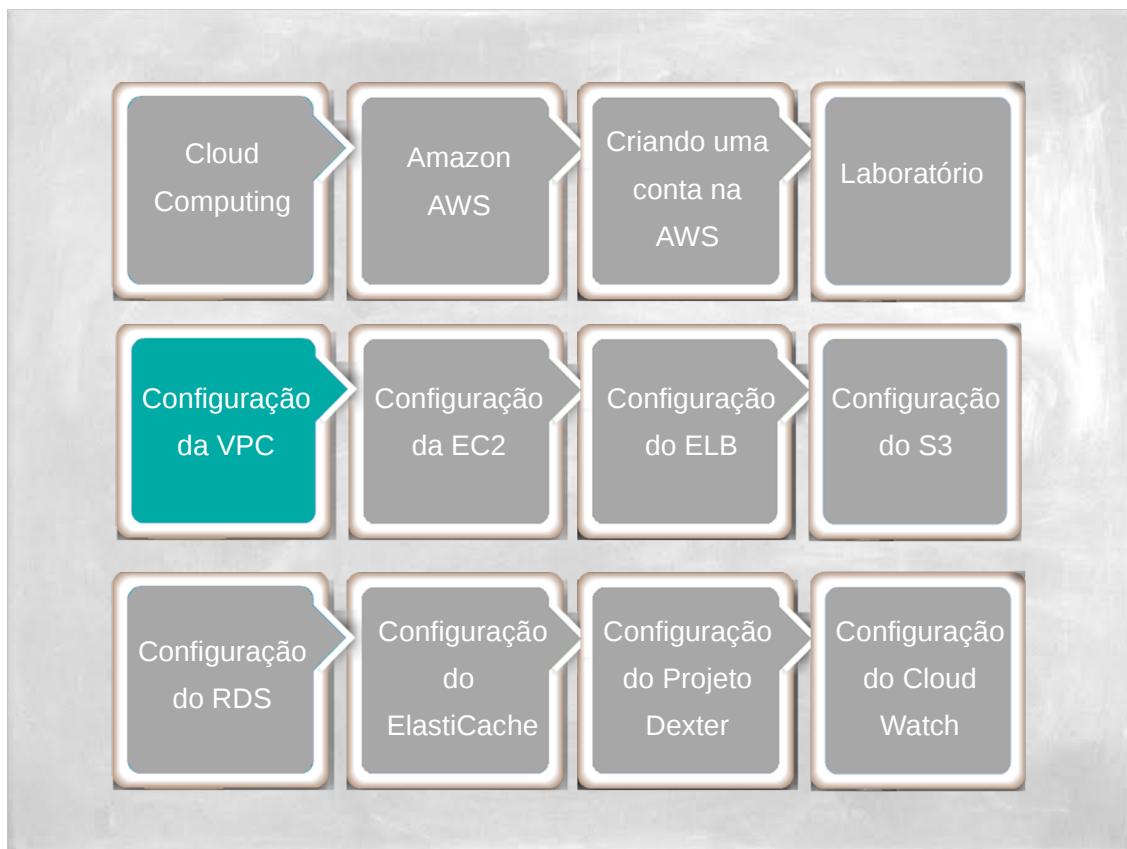
---

---

---

---

---

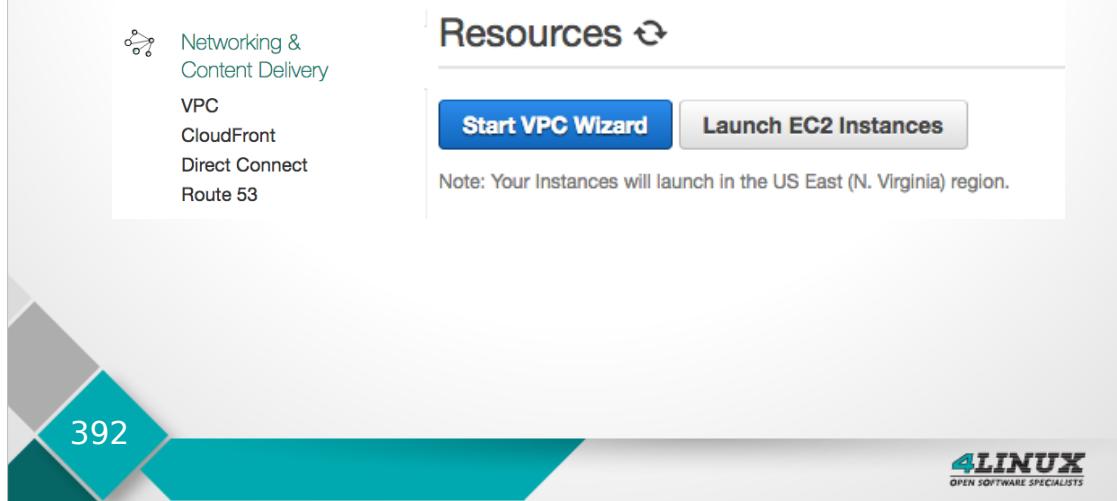


## Anotações:

## Configuração da VPC

No menu de serviços da AWS, selecione a opção **VPC**.

Clique no botão **Start VPC Wizard** para configuração de uma nova VPC.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração da VPC

Selecione o tipo de VPC: **VPC with a Single Public Subnet**.

**VPC with a Single Public Subnet**

Your instances run in a private, isolated section of the AWS cloud with direct access to the Internet. Network access control lists and security groups can be used to provide strict control over inbound and outbound network traffic to your instances.

**Creates:**

A /16 network with a /24 subnet. Public subnet instances use Elastic IPs or Public IPs to access the Internet.

**Select**

393

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração da VPC

Agora, vamos configurar as opções da **VPC** da Dexter.

**Step 2: VPC with a Single Public Subnet**

IPv4 CIDR block\*:  (65531 IP addresses available)

IPv6 CIDR block:  No IPv6 CIDR Block  
 Amazon provided IPv6 CIDR block

VPC name:

Public subnet's IPv4 CIDR\*:  (251 IP addresses available)

Availability Zone\*:

Subnet name:

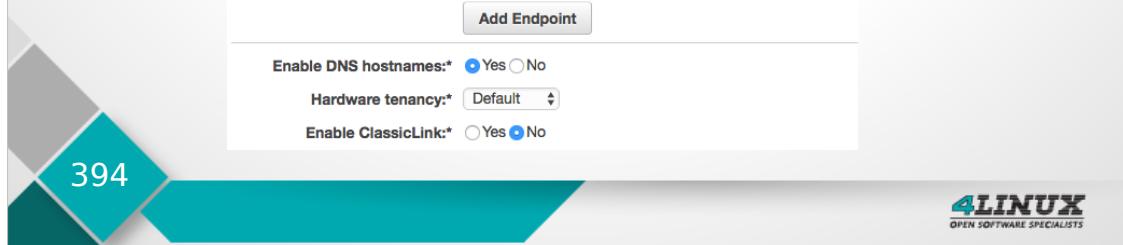
You can add more subnets after AWS creates the VPC.

Service endpoints

Enable DNS hostnames\*:  Yes  No

Hardware tenancy\*:

Enable ClassicLink\*:  Yes  No



**394**

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

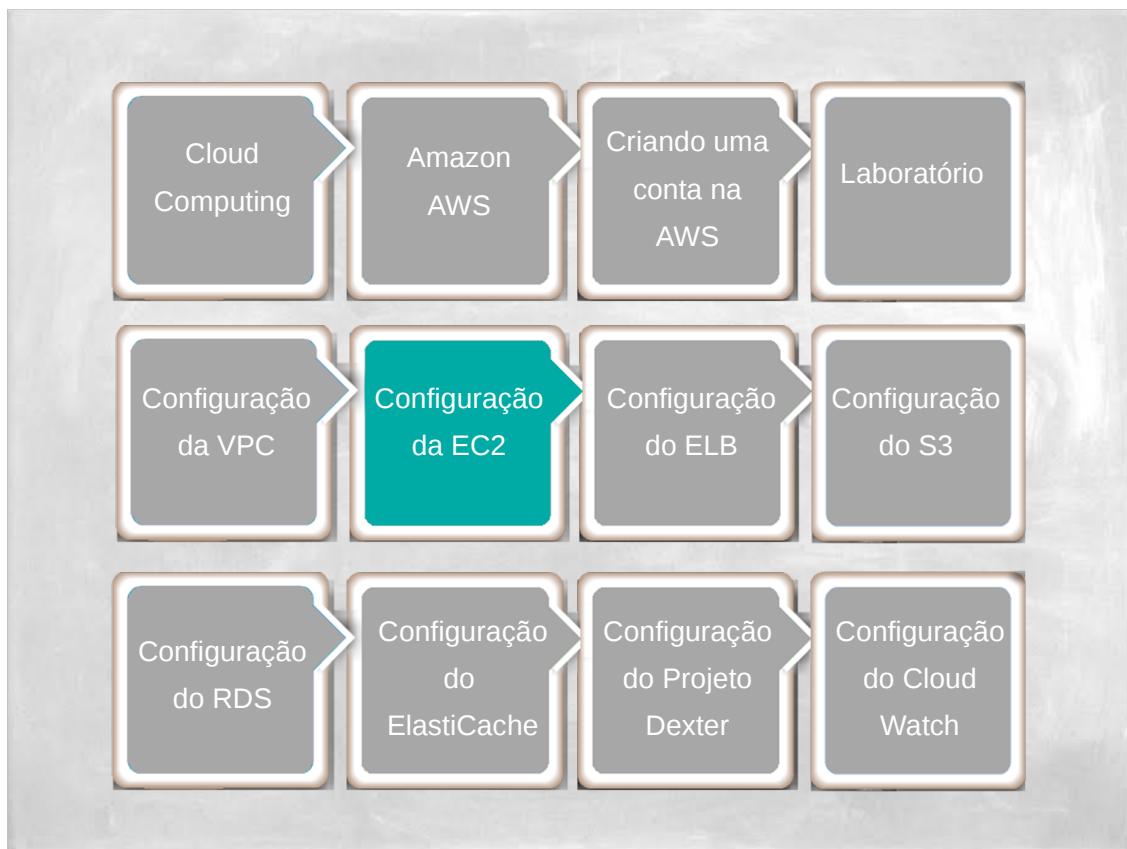
---

---

---

---

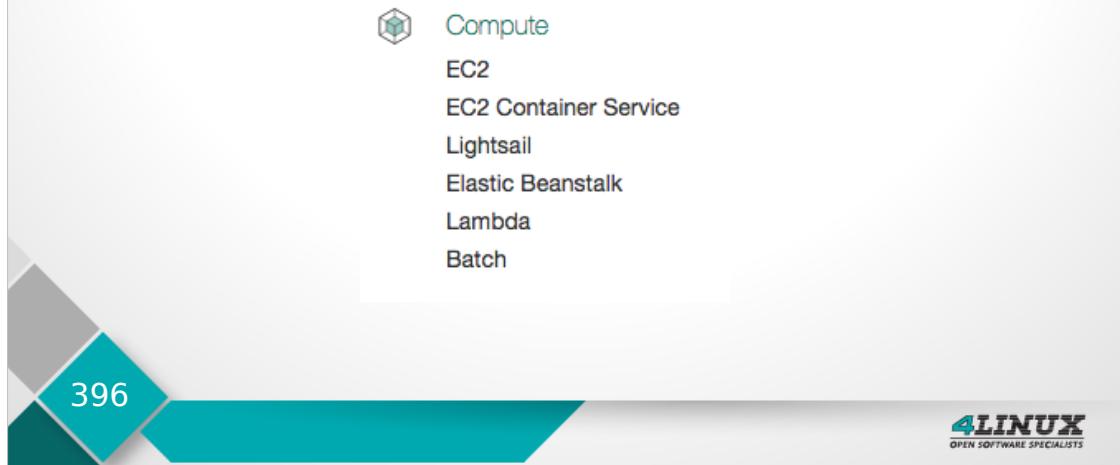
---



## Anotações:

## Configuração da EC2

No menu de serviços da AWS, selecione a opção **EC2** para criar e configurar as instâncias do EC2.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração da EC2

Na página principal do **EC2**, você pode observar um painel com informações sobre as configuração do serviço como, por exemplo, quantas instâncias estão ativas no momento, quantos Elastic Ips, Security Group, entre outras informações.

You are using the following Amazon EC2 resources in the US East (N. Virginia) region:

0 Running Instances	0 Elastic IPs
0 Dedicated Hosts	0 Snapshots
0 Volumes	0 Load Balancers
2 Key Pairs	9 Security Groups
0 Placement Groups	

397



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração da EC2

Vamos criar o par de chaves para acesso **SSH** a instâncias do **EC2**.

You are using the following Amazon EC2 resources in the US East (N. Virginia) region:

0 Running Instances

0 Dedicated Hosts

0 Volumes

2 Key Pairs

0 Replacement Groups

0 Elastic IPs

0 Snapshots

0 Load Balancers

9 Security Groups



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração da EC2

Clique no botão **Create Key Pair** e depois defina o nome da chave.

399

4LINUX  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

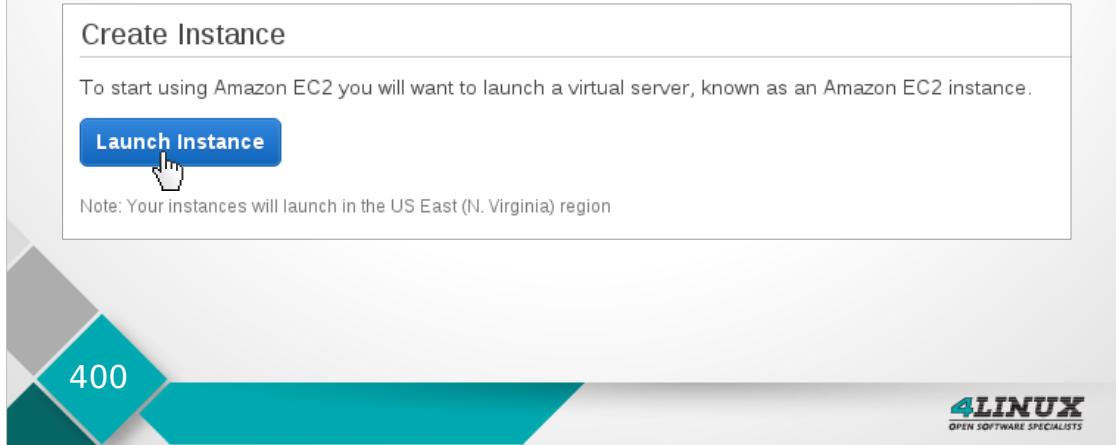
---

---

## Configuração da EC2

Agora que temos o nosso par de chaves configurado, vamos criar a nossa instância EC2.

Selecione a opção **Launch Instances** para criar novas instâncias do serviço.



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração da EC2

Para o nosso laboratório, iremos utilizar AMI do Ubuntu Server 16.04.  
Selecione a imagem clicando no botão **SELECT**.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração da EC2

Selecione a instância do tipo **t2.micro** e clique em **Next**. Configure **Instance Details** para configurar as informações de disco, network, security groups, entre outras configurações.

The screenshot shows a table of instance types. The columns are Family, Type, vCPUs, and Memory (GiB). There are two rows: one for t2.nano (1 vCPU, 0.5 GiB) and one for t2.micro (1 vCPU, 1 GiB, labeled as 'Free tier eligible'). A blue arrow points to the t2.micro row. Below the table is a navigation bar with buttons for Cancel, Previous, Review and Launch (which is highlighted in blue), and Next: Configure Instance Details.

	Family	Type	vCPUs	Memory (GiB)
<input type="checkbox"/>	General purpose	t2.nano	1	0.5
<input checked="" type="checkbox"/>	General purpose	t2.micro Free tier eligible	1	1

Cancel Previous **Review and Launch** Next: Configure Instance Details

402

4LINUX  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração da EC2

Insira o número **2** para criar duas instâncias idênticas.

Selecione a **VPC** e **Subnet** configurada.

Após a configuração, clique em **Next: Add Storage**.

2 Launch into Auto Scaling Group ⓘ

You may want to consider launching these instances into an Auto Scaling Group to the future. [Learn how Auto Scaling can help your application stay healthy and cost efficient](#)

Request Spot instances

vpc-b64066d2 (172.30.0.0/16) [Create new VPC](#)

subnet-3f46e667(172.30.0.0/24) | us-east-1a [Create new subnet](#)  
251 IP Addresses available

Use subnet setting (Enable)

None [Create new IAM role](#)

Stop

[Review and Launch](#) [Next: Add Storage](#)



**4LINUX**  
OPEN SOFTWARE SPECIALISTS

## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

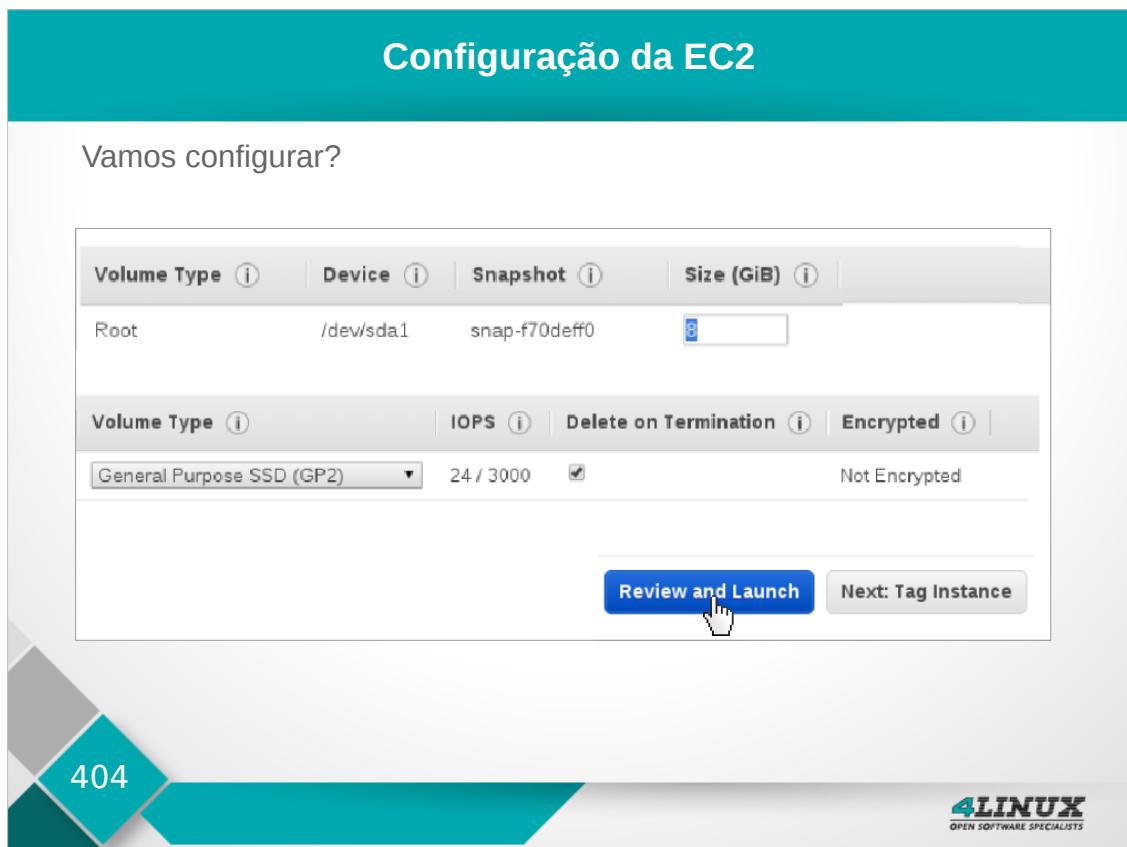
---

---

---

---

---



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

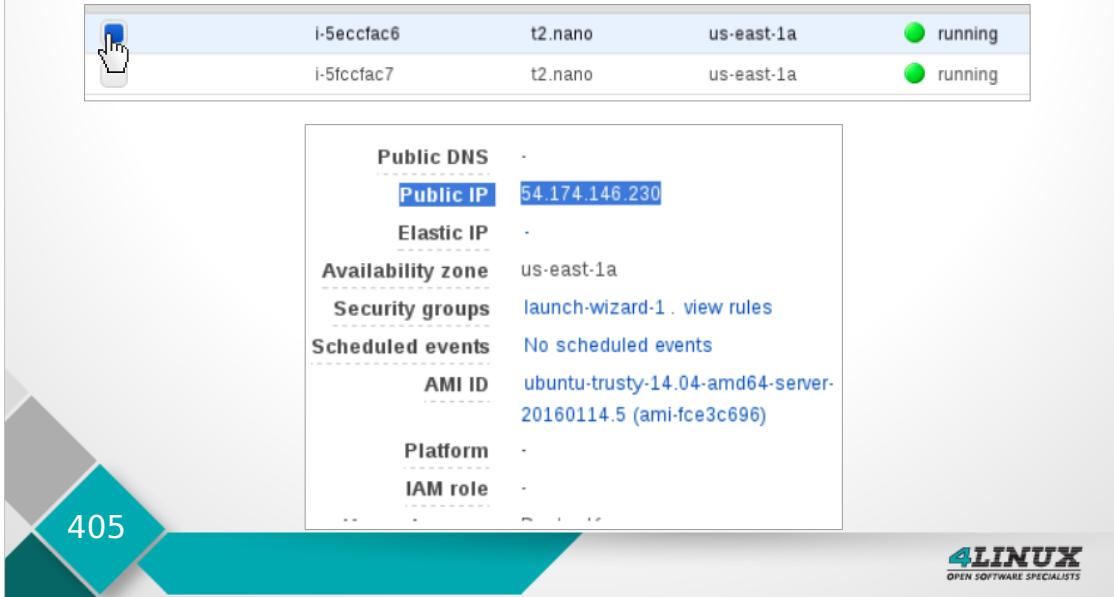
---

---

---

## Configuração da EC2

Após a criação da instância, vamos identificar o **IP Público** para realizar o acesso via SSH.



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração da EC2

Realize o acesso via SSH nos contêiners, utilizando a chave criada no começo do capítulo.

1

Seguindo e exemplo: (substitua o **ip-da-máquina**)

```
1# ssh -i DexterKey.pem ubuntu@<ip-da-máquina>
```

Repita o processo para a segunda instância criada.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

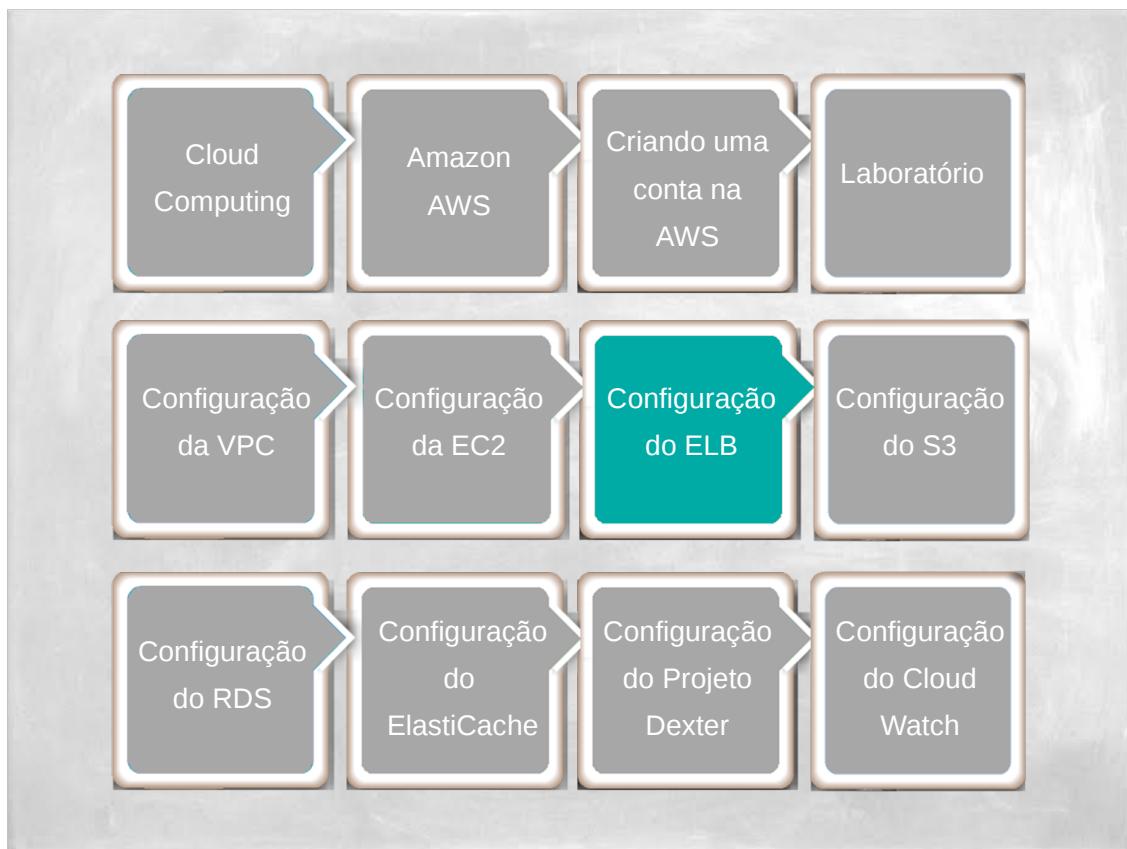
---

---

---

---

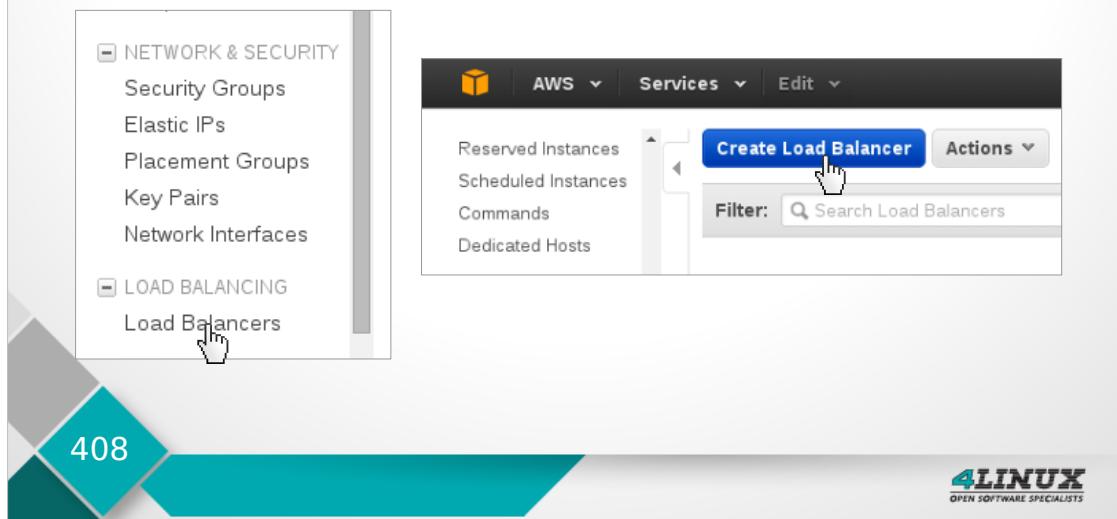
---



## Anotações:

## Configuração do Load Balancer

Na página principal do EC2, selecione a opção **Load Balancers** no menu lateral e depois clique em **Create Load Balancer** para criar um novo Load Balancer.



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

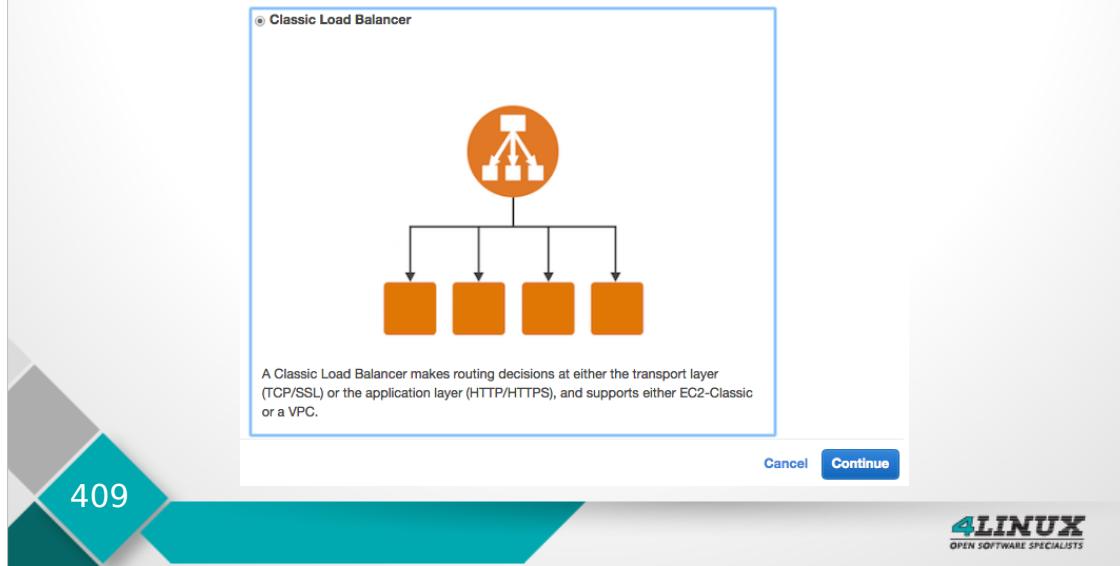
---

---

---

## Configuração do Load Balancer

Existem dois modelos de Load balancer, como estamos usando uma VPC, use o método “Classic”



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração do Load Balancer

Defina o nome do **Load Balancer name**. Escolha o protocolo HTTP para ser balanceado e clique em **Next: Assign Security Groups**.

The screenshot shows the 'Create New Load Balancer' step of the AWS Load Balancer configuration wizard. The 'Load Balancer name' is set to 'dexterlb'. The 'Create LB Inside' dropdown is set to 'EC2-Classic'. Under 'Listener Configuration', there is one entry: 'Load Balancer Protocol: HTTP', 'Load Balancer Port: 80', 'Instance Protocol: HTTP', and 'Instance Port: 80'. At the bottom right are 'Cancel' and 'Next: Assign Security Groups' buttons. A decorative graphic with the number '410' is visible on the left.

Load Balancer name: dexterlb  
Create LB Inside: EC2-Classic

Listener Configuration:

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port
HTTP	80	HTTP	80

Cancel Next: Assign Security Groups

410

4LINUX  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração do Load Balancer

Vamos configurar como o Load Balancer realiza o **Heathy Check** dos nós do cluster.

O **Heathy Check** é usado para definir se uma instância pode ou não responder pela conexão.

**Step 4: Configure Health Check**

Your load balancer will automatically perform health checks on you from the load balancer. Customize the health check to meet your s

Ping Protocol: TCP  
Ping Port: 80

**Advanced Details**

Response Timeout	5	seconds
Health Check Interval	30	seconds
Unhealthy Threshold	5	
Healthy Threshold	10	

**Cancel** **Previous** **Next: Add EC2 Instances**

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Heathy Check

**Ping Protocol:** Define o protocolo utilizado para testar a conexão entre os nós do cluster.

**Ping Port:** Define a porta que será testada durante a conexão.

**Response Timeout:** Tempo de espera para o teste de conexão entre os nodes. Por exemplo, **5 segundos** significa que será enviado o pacote. Caso a máquina não responda durante 5 segundos é considerado como pacote perdido.

**Heath Check Interval:** Intervalo de tempo para os próximos testes.

**Unhealthy Threshold:** O número define quantos pacotes com falhas consecutivos devem ser utilizados para declarar que uma instância do nó não está apta a receber as conexões do Load Balancer.

**Healthy Threshold:** O número define quantos pacotes enviados com sucesso consecutivos devem ser recebidos antes de declarar que uma instância do nó está apta a receber as conexões novamente.

## Configuração do Load Balancer

Selecione as duas instâncias para fazer parte do cluster do nosso **Load Balancer**.

**Step 5: Add EC2 Instances**  
The table below lists all your running EC2 Instances. Check the boxes in the Select column to add those instances to

VPC vpc-b64066d2 (172.30.0.0/16)

Select	Instance	Name	State	Security Groups
<input checked="" type="checkbox"/>	i-5eccfac6		running	launch-wizard-1
<input checked="" type="checkbox"/>	i-5fcctac7		running	launch-wizard-1

**Cancel** **Previous** **Next: Add Tags** 



412

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração do Load Balancer

Vamos instalar o apache e testar o Load Banlancer.

- 1 Execute nas duas Instâncias.

```
1# apt-get install apache2
```

- 1.1 Execute na primeira Instância.

```
2# echo "Server Node 01" >  
/var/www/html/index.html
```

- 1.2 Execute na segunda Instância.

```
3# echo "Server Node 02" >  
/var/www/html/index.html
```



**4LINUX**  
OPEN SOFTWARE SPECIALISTS

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração do Load Balancer

Para acessar precisamos identificar o nome criado para Load Balancer.

Na página de configuração no Load Balancer, selecione o Load Balancer criada e na aba **Description** identifique a opção **DNS Name**.

Copie o valor do DNS em um web browser para visualizar as páginas.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

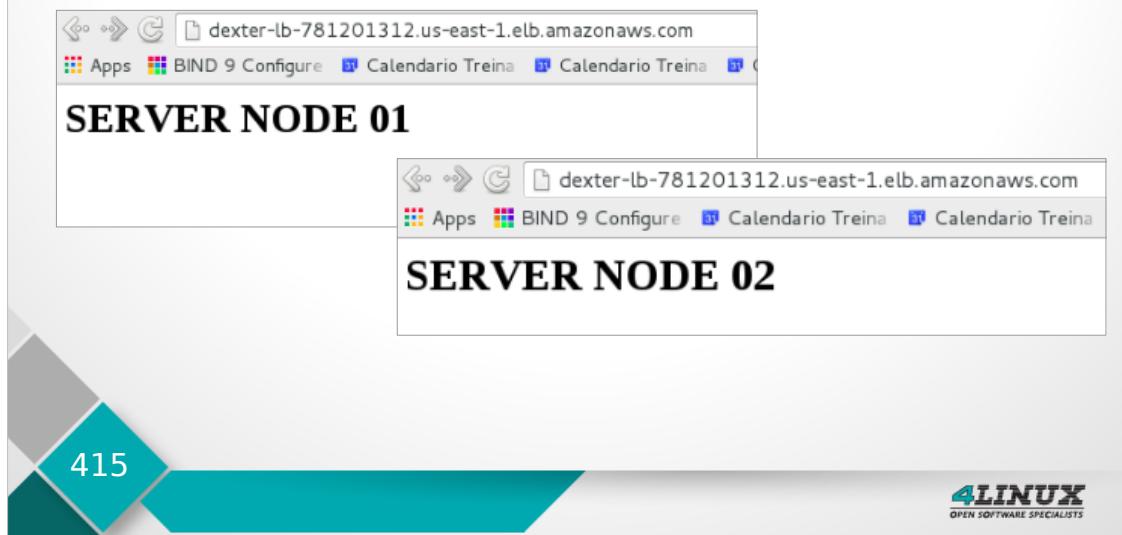
---

---

---

## Configuração do Load Balancer

Para realizar o teste recarregue a página algumas vezes e perceba as mudanças de páginas.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

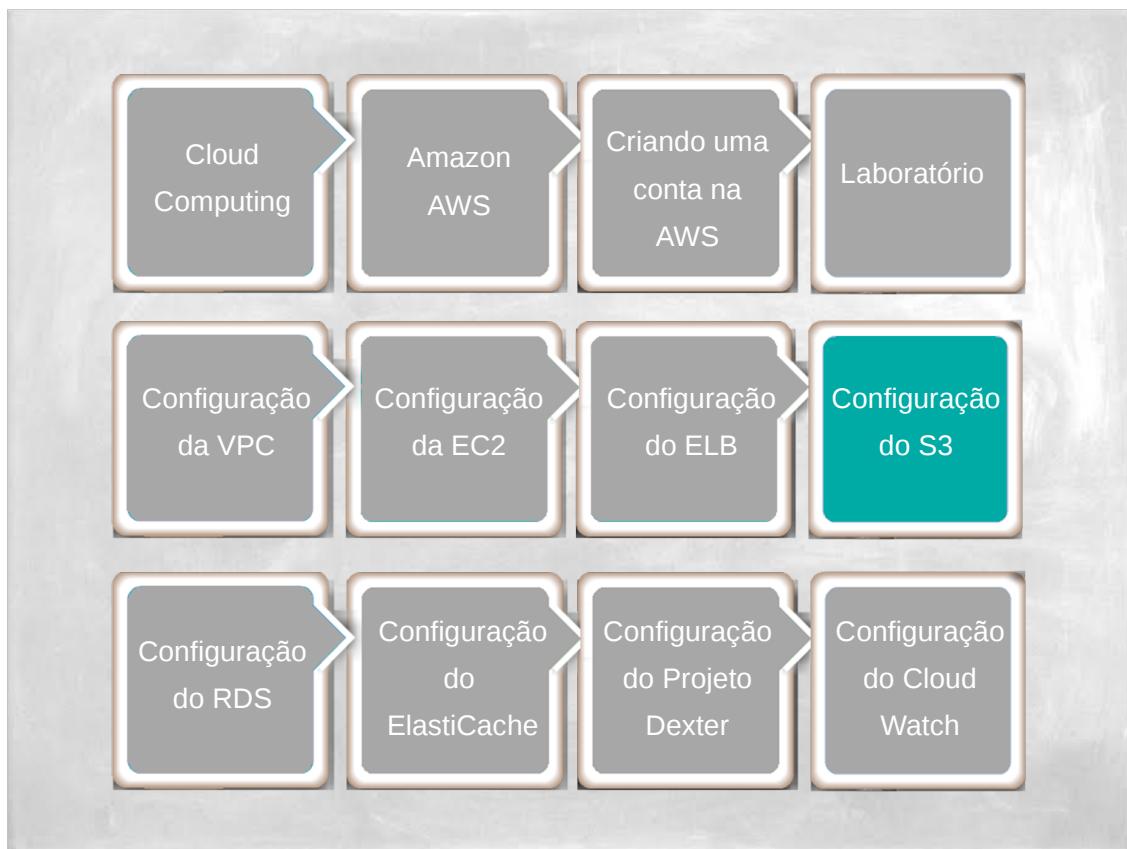
---

---

---

---

---

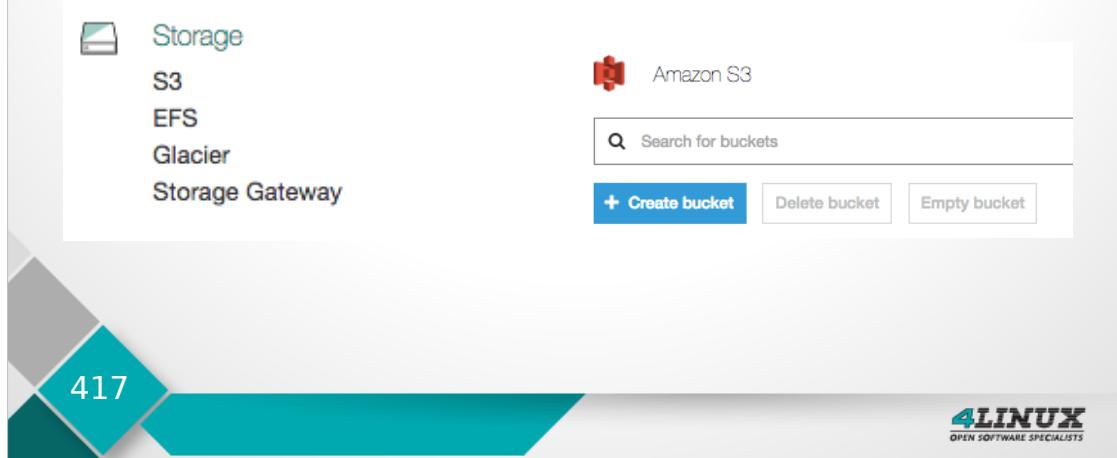


## Anotações:

## Configuração do S3

No portal da Dexter, iremos utilizar o **S3** para armazenar as imagens do site da Dexter.

Vamos criar um **Bucket** dentro do S3. Clique no botão **Create Bucket**.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

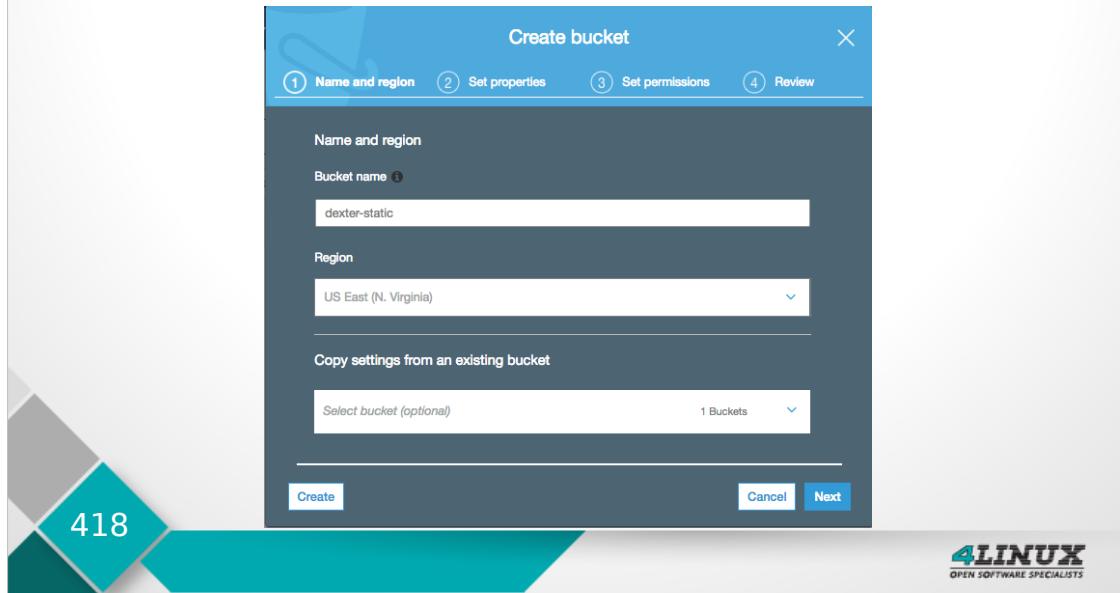
---

---

---

## Configuração do S3

Agora, vamos definir o nome do nosso **Bucket**.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

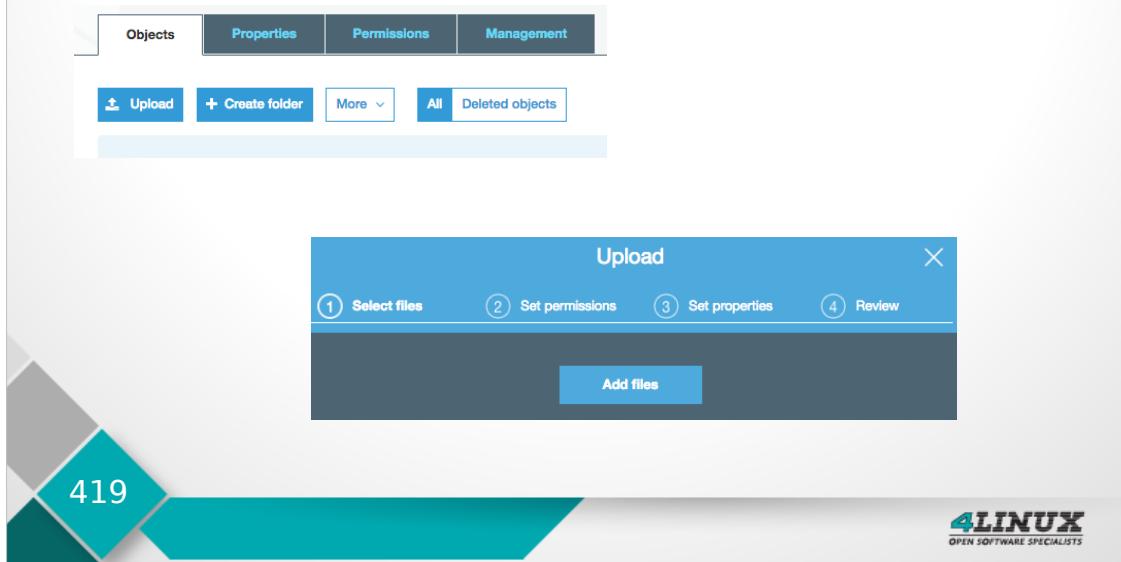
---

---

---

## Configuração do S3

Realize o **upload** do logo da Dexter. Depois, realize o download do site da Dexter no ambiente ead da 4linux.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

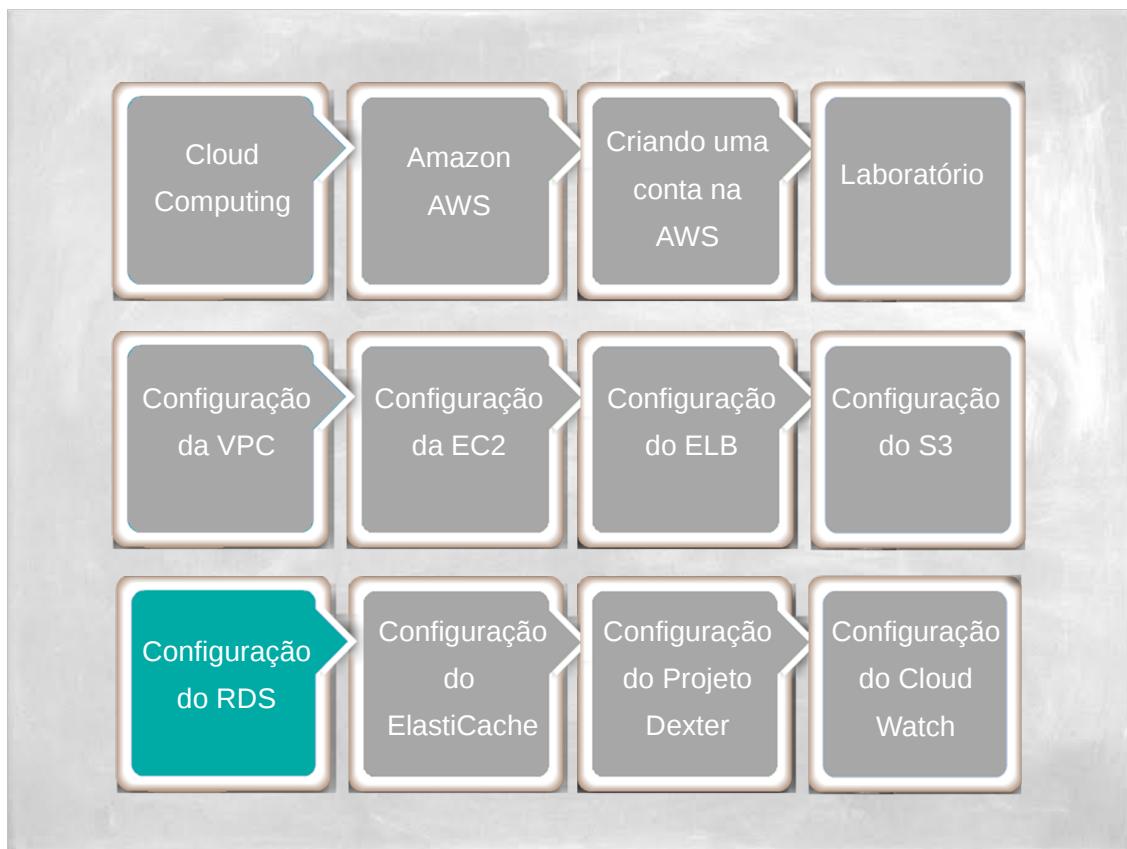
---

---

---

---

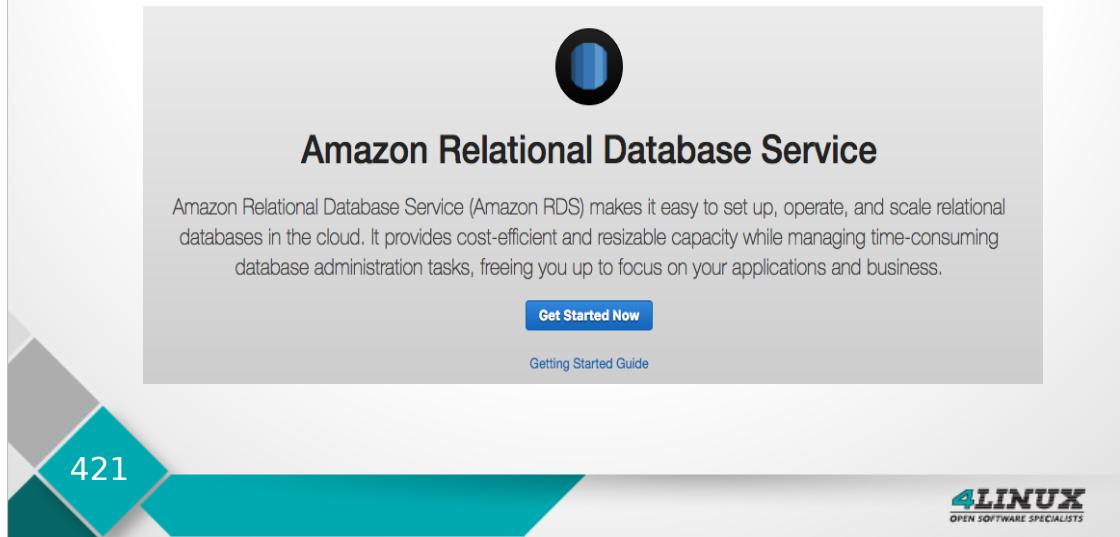
---



## Anotações:

## Configuração do RDS

Na página de serviços da AWS, selecione RDS. Após, clique em **Launch DB Instance** para criar uma nova instância da AWS.



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração do RDS

O Portal da Dexter irá utilizar o Banco de Dados **MySQL** para armazenar seus dados.

**Select Engine**

To get started, choose a DB Engine below and click Select.

Amazon Aurora

MySQL  
MySQL Community Edition  
**Free tier eligible**

MariaDB

PostgreSQL

ORACLE

SQL Server

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 6 TB.
- Instances offer up to 32 vCPUs and 244 GiB Memory.
- Supports automated backup and point-in-time recovery.
- Supports cross-region read replicas.

422

4LINUX  
OPEN SOFTWARE SPECIALISTS

## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração do RDS

Vamos selecionar o MySql “Free Usage Tier”.

Do you plan to use this database for production purposes?

Production

Amazon Aurora  
**Recommended**  
MySQL-compatible, enterprise-class database at 1/10th the cost of commercial databases.

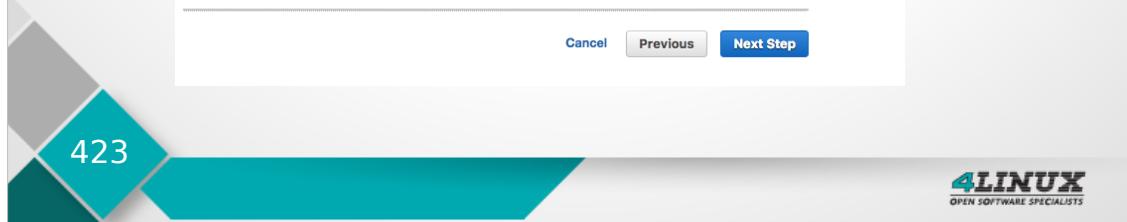
MySQL  
Use Multi-AZ Deployment and Provisioned IOPS Storage as defaults for high availability and fast, consistent performance.

Dev/Test

MySQL  
This instance is intended for use outside of production or under the RDS Free Usage Tier.

Billing is based on [RDS pricing](#).

[Cancel](#) [Previous](#) [Next Step](#)



The graphic features a stylized '4' composed of overlapping triangles in shades of grey and teal. To the right of the '4' is the 4LINUX logo, which includes a stylized '4' icon followed by the word 'LINUX' in a bold, sans-serif font, with 'OPEN SOFTWARE SPECIALISTS' in smaller text below it.

### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração do RDS

Vamos configurar a informação do banco de dados.

Escolha o tipo da instância **db.t2.micro** e aloque **5GB** inicialmente para o nosso banco.

### Instance Specifications

DB Engine mysql  
License Model general-public-license  
DB Engine Version MySQL 5.6.27

Review the Known Issues/Limitations to learn about potential compatibility issues with specific database versions.

DB Instance Class db.t2.micro — 1 vCPU, 1 GiB RAM  
Multi-AZ Deployment No  
Storage Type General Purpose (SSD)  
Allocated Storage\* 5 GB



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração do RDS

Defina as informações de acesso à base de dados.

**Nome do banco:** dexter-dbmaster

**Nome do usuário:** dexter

**Senha:** portal@dexter

Settings

DB Instance Identifier*	dexter-dbmaster
Master Username*	dexter
Master Password*	.....
Confirm Password*	.....

\* Required      Cancel      Previous      Next Step



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

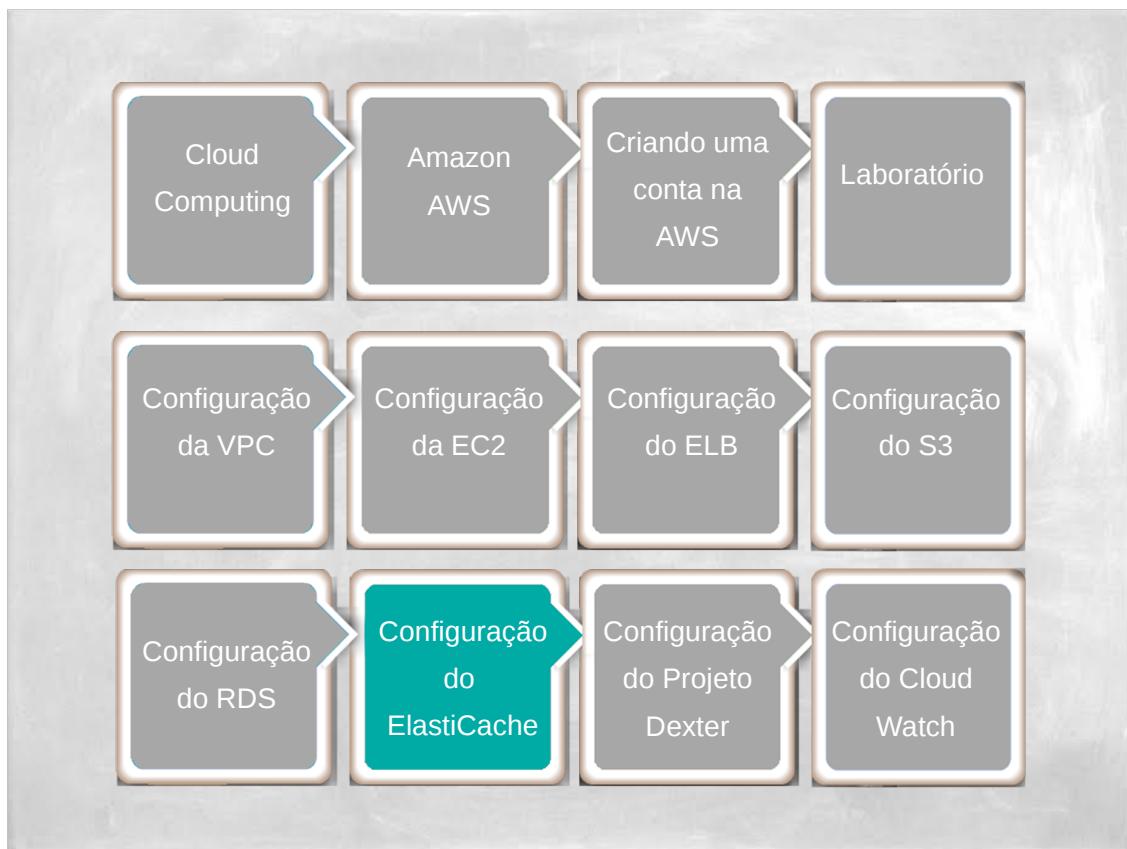
---

---

---

---

---



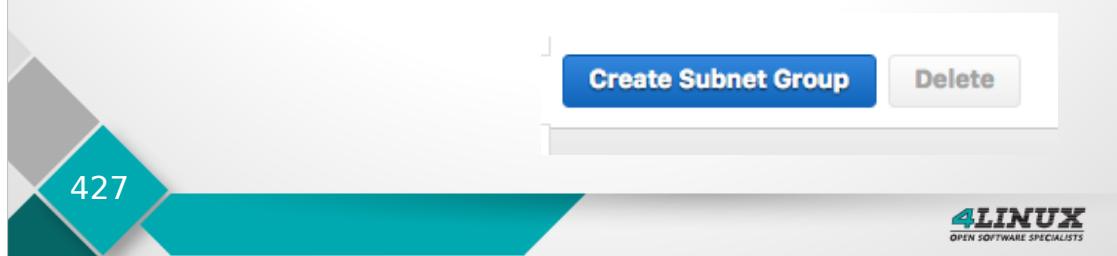
## Anotações:

## Configuração do ElastiCache

Para configurar algumas instâncias do serviço ElastiCache é necessário configurar um **Cache Subnet Group**.

Security Groups (1)  
Subnet Groups (3)  
Supported Platforms EC2,VPC  
Default Network none

Dentro da página do ElastiCache, selecione a opção para **Cache Subnet Groups** no menu lateral.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração do ElastiCache

Defina as opções de configuração do serviço. Adicione uma subnet configurada no capítulo de VPC e clique no botão **Create** para finalizar a criação do grupo.

To create a new Subnet Group give it a name, description, and select an existing VPC below. Once you select an existing VPC, you will be able to add subnets related to that VPC.

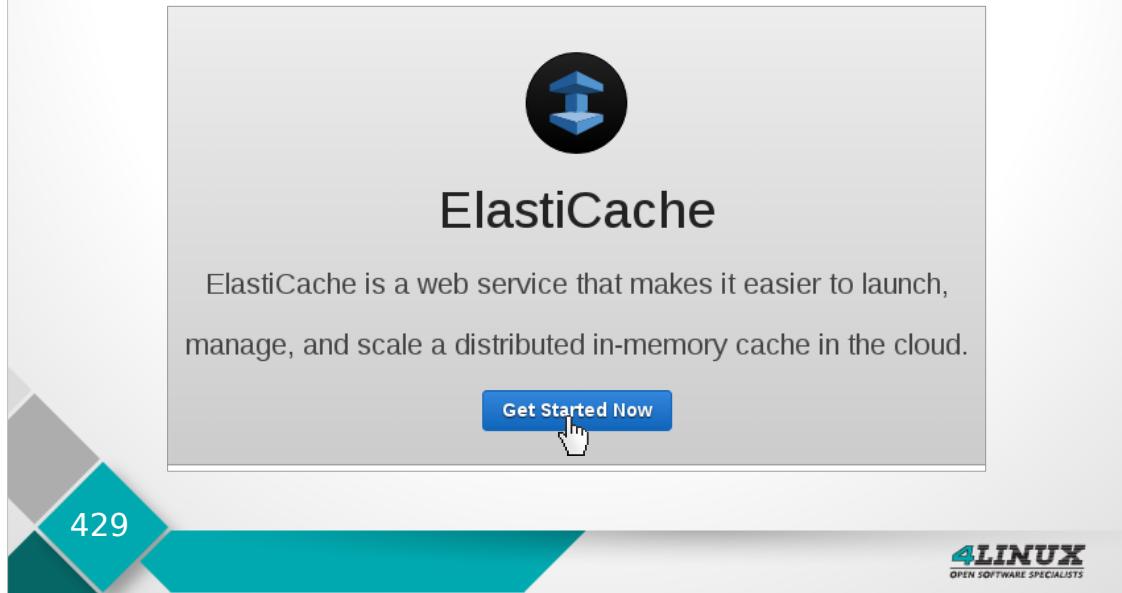
Name*	<input type="text" value="dexter-cache"/>				
Description*	<input type="text" value="Store PHP Sessions"/>				
VPC ID	<input type="text" value="vpc-b64066d2"/>				
Add Subnet(s) to this Subnet Group. You may add subnets one at a time below or <a href="#">add all the subnets</a> related to this VPC. You may make additions/edits after this group is created.					
Availability Zone	<input type="text" value="us-east-1a"/>	Availability Zone	Subnet ID	CIDR Block	Action
Subnet ID	<input type="text" value="subnet-3f46e667"/>				None added
<input type="button" value="Add"/>					
			<input type="button" value="Cancel"/>	<input type="button" value="Create"/>	

## Anotações:



## Configuração do ElastiCache

Agora, vamos configurar as instâncias do nosso servidor de cache na memória.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

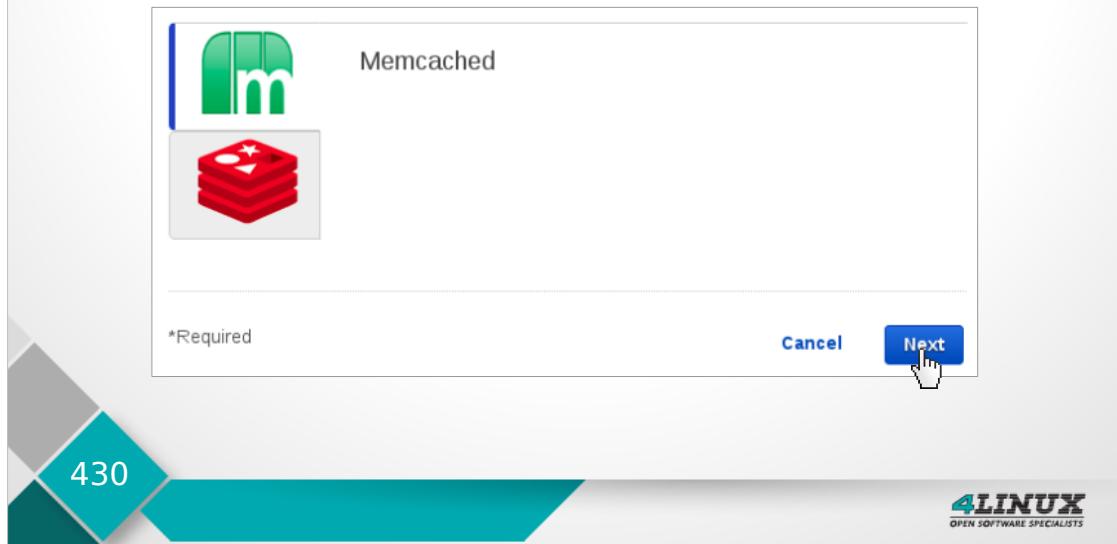
---

---

---

## Configuração do ElastiCache

Selecione o **Memcached** como nosso Engine de armazenamento de Cache em memória.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração do ElastiCache

Defina as configuração do nosso **ElastiCache**.

**Specify Cluster Details**

**Cluster Specifications**

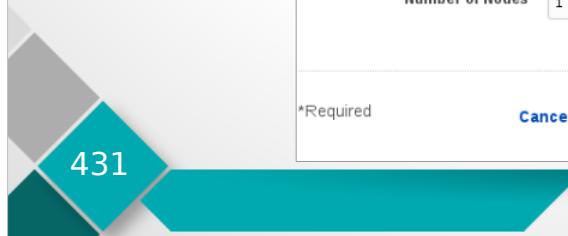
Engine	Memcached	<small>i</small>
Engine Version	1.4.24	<small>i</small>
Port*	11211	<small>i</small>
Parameter Group	default.memcached1.4	<small>i</small>

**Configuration**

Cluster Name*	memcache	<small>i</small>
Node Type	cache.t2.micro (555 MB me...)	<small>i</small>
Number of Nodes*	1	<small>i</small>

\*Required      Cancel      Previous      **Next** 

**4LINUX**  
OPEN SOFTWARE SPECIALISTS



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

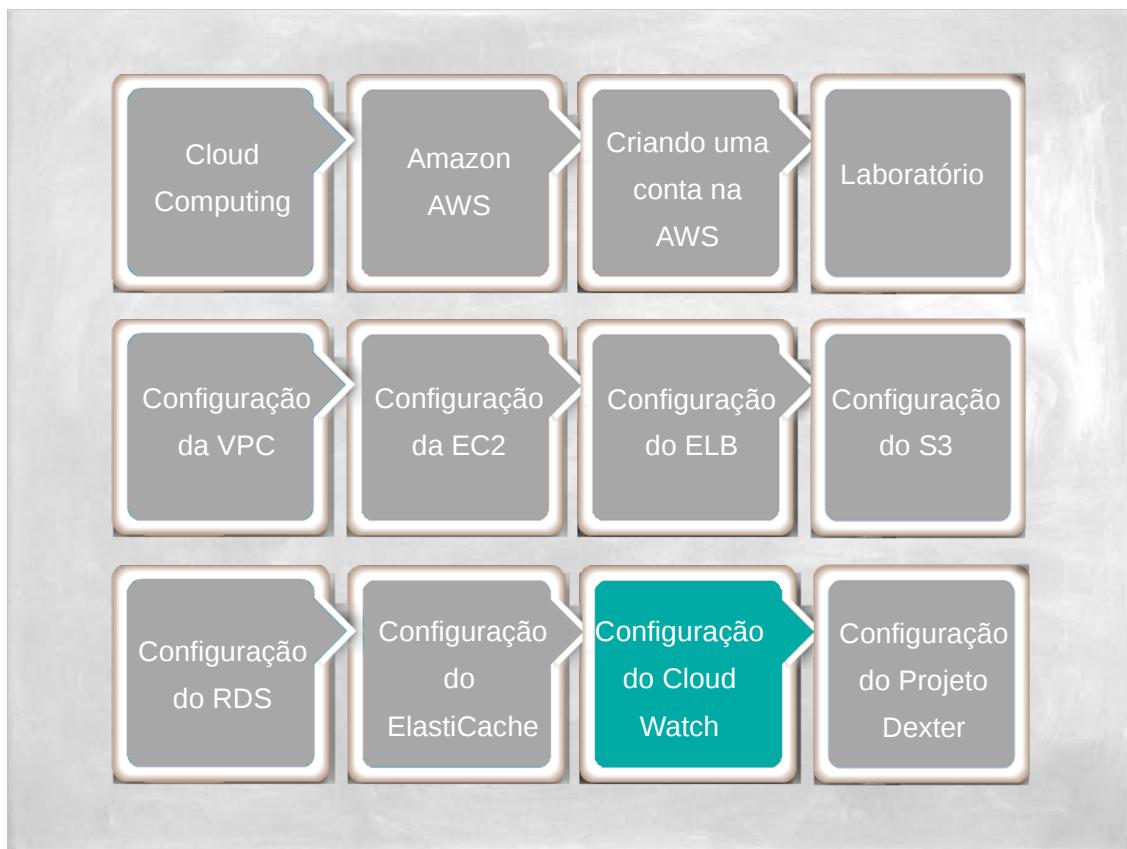
---

---

---

---

---

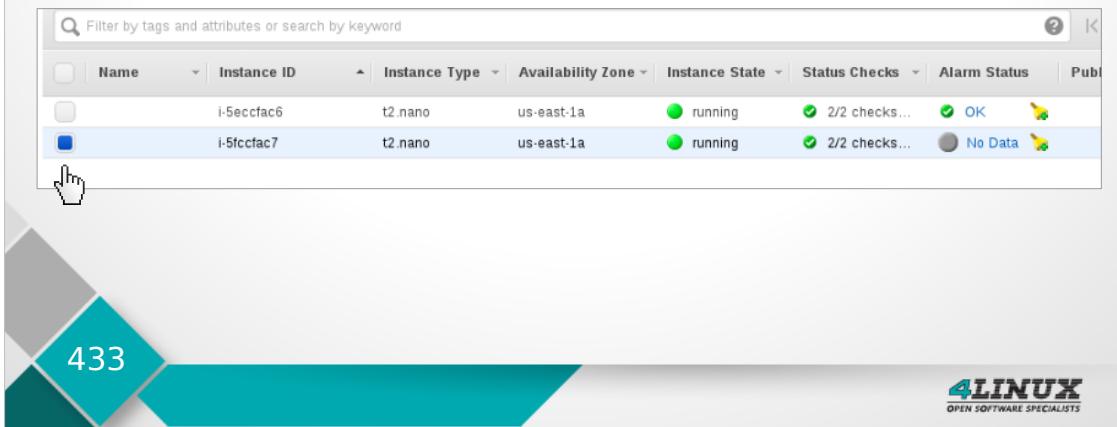


## Anotações:

## Configuração do Cloud Watch

Agora, vamos configurar alarmes do **Cloud Watch** automaticamente para verificar problemas em nossas instâncias.

Na página de instâncias do EC2, clique com o botão direito.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Configuração do Cloud Watch

Defina as opções do alarme do **Cloud Watch**.

**Create Alarm**

You can use CloudWatch alarms to be notified automatically whenever metric data reaches a level you define. To edit an alarm, first choose whom to notify and then define when the notification should be sent.

**Send a notification to:** dexter-4linux (rafael.silva@4linux.com.br) [create topic](#)

**Take the action:**  Recover this instance [i](#)  
 Stop this instance [i](#)  
 Terminate this instance [i](#)  
 Reboot this instance [i](#)

**Whenever:** Average of Network Out  
Is: > 1 Bytes

**For at least:** 1 consecutive period(s) of 5 Minutes

**Name of alarm:** awsec2-i-5fccfac7-High-Network-Out

**Network Out Bytes**



Cancel **Create Alarm**

434

4LINUX  
OPEN SOFTWARE SPECIALISTS

## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

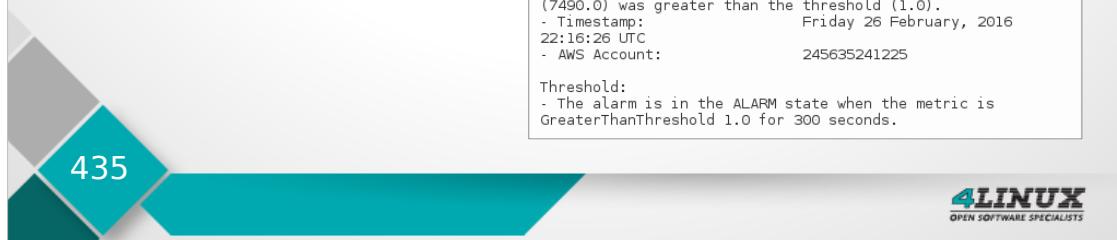
---

## Configuração do Cloud Watch

Você deve receber um e-mail para confirmar a sua inscrição no serviço de alarme da Amazon.

Após confirmar o e-mail, desligue a máquina e aguarde o alerta no e-mail.

Segue exemplo ao Lado:



## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Projeto Dexter

### Missão realizada!

Agora todos os clientes e fornecedores poderão utilizar o portal de gerenciamento de **Projetos da Dexter Courier**.



### Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Amazon Calculator



**Amazon Calculator:** A Amazon possui uma ferramenta online para calcular os gastos da AWS.



<http://calculator.s3.amazonaws.com/index.html>



437

**4LINUX**  
OPEN SOFTWARE SPECIALISTS

## Anotações:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



**Curso: 4525**

**Infraestrutura ágil com práticas  
DEVOPS usando Docker, Git,  
Jenkins, Puppet e Ansible**

Versão: 4.0

```
2# vim puppet/tasks/main.yml
---

# tasks file for puppet
- name: Download do Repositório do Puppet
  apt:
    deb: "https://apt.puppetlabs.com/puppetlabs-release-pc1-
{{ ansible_distribution_release }}.deb"
    state: present

- name: Instalação dos pacotes e dependências do puppet
  apt: name={{ item }} update_cache=yes state=present
  with_items:
    - openssl
    - ntp
    - puppet-lint
    - puppet
    - puppet-common
```

```
3# vim gitlab/tasks/main.yml
---

# tasks file for gitlab
- name: Instalação das dependências do GitLab
  apt: name={{ item }} update_cache=yes state=present
  with_items:
    - curl
    - openssh-server
    - postfix
    - ca-certificates

- name: Baixar o pacote com o repositório get_url:
  get_url:
    url: "http://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.deb.sh"
    dest: /tmp/script.deb.sh
    owner: root
    group: root
    mode: 0777
```

```
- name: Instalação do repositório do GitLab
  command: /tmp/script.deb.sh
```