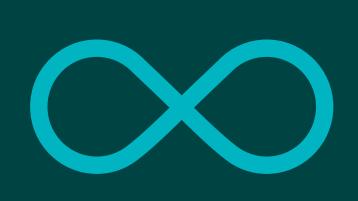


CURSO **525**

INFRAESTRUTURA ÁGIL COM PRÁTICAS DEVOPS







Construindo a infraestrutura como código

Ansible Galaxy

Anotações	

4LINU2	Objetivos da aula	
1	O que é uma Role	
2	Ansible Galaxy	
3	Usando Roles do Ansible Galaxy	
4	Criando Roles	

Anotações		

O que é uma Role

Existem momentos que nossa playbook pode ficar muito grande e complexa, dificultando a manutenção futura. Para facilitar, o Ansible possui o sistema de roles.

Uma role nada mais é do que um diretório contendo um conjunto de arquivos, variáveis e playbooks que deverão ser aplicados em seu ambiente.



Anotações	
-	

Estrutura de uma Role

Uma role é formada pelos seguintes diretórios:

default

Diretório, onde será informado os valores padrões das variáveis de sua role;

files

Diretório, onde são armazenados arquivos que deverão ser copiados para dentro do servidor;

handlers

Módulos que deverão ser executados, após um módulo ter sido executado – é feito em background.

Anotações

Anotações	

4LINUX Ansible Galaxy



Ansible Galaxy é um site de hospedagem de roles do Ansible, permitindo que você compartilhe suas roles com outros usuários, ou utilize as roles de outros usuários.

No site existem roles das mais variadas atividades, desde instalação do Prometheus até de gerenciamento do Bind.

Anotações		



Usando Roles do Ansible Galaxy

Para trabalhar com roles, usamos o comando ansible-galaxy que permite criar, listar, baixar e remover roles, igual quando usamos o apt e o yum. Vamos tentar localizar a role "geerlingguy.puppet" para utilizarmos.

Na pasta infraagil, conecte-se a máquina automation para acessar o diretório de roles: 1

- # vagrant ssh automation
- # cd /etc/ansible/roles
- Veja se consegue localizar a role geerlingguy.puppet: 2
 - # ansible-galaxy search puppet
- Instale a role geerlingguy.puppet: 3 # sudo ansible-galaxy install geerlingguy.puppet

automation.4labs.example

Abaixo, segue alguns dos parâmetros que o ansible-galaxy pode receber:

search: Procura por playbooks, cujo o nome seja o informado;

list: Lista playbooks que estejam instaladas no servidor no momento;

install: Instala uma playbook no servidor;

install -f: Atualiza uma playbook para a ultima versão;

remove: Remove uma playbook;

init: Inicializa um diretório com a estrutura de uma playbook.



Usando Roles do Ansible Galaxy

Crie uma playbook chamada compliance.yml dentro de /etc/ansible/playbooks para que possamos usar a role que baixamos.

Gere o arquivo compliance.yml utilizando o modelo da pasta Aula 6.1:

- # sudo cp /vagrant/4525/arquivos/Aula\ 6.1/compliance.yml
 /etc/ansible/playbooks/
- # cat /etc/ansible/playbooks/compliance.yml
- name: Aplicando as roles de Compliance hosts: compliance roles:
 - geerlingguy.puppet
- Execute a playbook:
 - # sudo ansible-playbook /etc/ansible/playbooks/compliance.yml

Anotações	

Criando Roles

Para criar uma role, usamos o parâmetro init do comando ansible-galaxy. Com ele, conseguimos criar a estrutura de uma role no diretório que estivermos. Vamos criar uma role de teste.

- Crie a role teste:
 - # sudo ansible-galaxy init teste
- Verifique a estrutura de diretórios da role:
 # tree /etc/ansible/roles/teste

automation.4labs.example

Anotações

/etc/ansible/roles/teste

- |-- README.md
- |-- defaults
- | `-- main.yml
- |-- files
- |-- handlers
- | `-- main.yml
- |-- meta
- | `-- main.yml
- |-- tasks
- | `-- main.yml
- |-- templates
- |-- tests
- | |-- inventory
- | `-- test.yml
- `-- vars
 - `-- main.yml

4LINU2	Recapitulando	10
1	O que é uma Role	
2	Ansible Galaxy	
3	Usando Roles do Ansible Galaxy	
4	Criando Roles	
,		

Anotações	





Construindo a infraestrutura como código

Construindo a Role puppet-agent

Anotações	

4LINU X	Objetivos da aula	2
1	Entendendo a estrutura	
2	Separando em Blocos	
3	Criando a Role do Puppet Agent	

Anotações	

Entendendo a Estrutura

O Puppet Agent será responsável pela comunicação da estação com o servidor Puppet.

Vamos separar nossa instalação em blocos, devido aos diferentes sistemas operacionais utilizados: CentOS, Debian e e Ubuntu.



Anotações		

Separando em Blocos

- ✔ Para facilitar a visualização das Playbooks, vamos separá-las em blocos de acordo com a sua distribuição;
- ✔ Blocos são agrupamentos lógicos de tarefas, e são muito utilizados quando queremos aplicar uma diretiva when, por exemplo em várias tasks;
- ✔ Utilizaremos a condicional when em cada bloco, para filtrar o que deve ser executado em cada distribuição.

Anotações		



Criando a Role Puppet Agent

É importante saber que devemos criar algumas estruturas condicionais, utilizando o **when** para cada bloco, conforme o exemplo a seguir.

- name: Download e instalação do puppet Agent - RedHat

block:

- yum:

name: 'https://yum.puppet.com/puppet6-release-el-7.noarch.rpm'

state: present

- yum:

name: puppet-agent

state: present

when: ansible_distribution == 'CentOS'

automation.4labs.example

O parâmetro **when** está alinhado com o bloco, isso significa que esta condicional serve para qualquer task dentro do bloco declarado.

Para verificar o ansible_distribution de cada máquina, podemos executar o comando:

ansible all -m setup -a "filter=ansible distribution"



Separando em Blocos

Vamos visualizar os blocos que iremos criar.

Bloco: Debian

- get_url:

url: https://apt.puppetlabs.com/puppet-release-buster.deb

dest: /tmp/puppet6.deb

- apt:

1

deb: /tmp/puppet6.deb

- apt:

name: puppet-agent
state: present
update_cache: yes

notações	



Separando em Blocos

Vamos visualizar os blocos que iremos criar.

Bloco: Ubuntu

- get_url:

url: https://apt.puppetlabs.com/puppet6-release-bionic.deb

dest: /tmp/puppet6.deb

- apt:

1

deb: /tmp/puppet6.deb

- apt:

name: puppet-agent
state: present
update_cache: yes

notações	

Separando em Blocos

Bloco: RedHat

- yum:

1

name: 'https://yum.puppet.com/puppet6-release-el-7.noarch.rpm'

state: present

- yum:

name: puppet-agent
state: present

Anotações	

1

Separando em Blocos

Neste bloco, vamos editar o arquivo de configuração do puppet para que todos os servidores conheçam o endereço do servidor.

```
Bloco: Adicionando Configurações ao puppet.conf
lineinfile:
   path: '/etc/puppetlabs/puppet/puppet.conf'
line: "{{ item }}"
loop:
   - '[main]'
   - 'runinterval = 1h'
   - '[agent]'
   - 'server = compliance.4labs.example'
   - 'certname = {{ ansible_nodename }}'
```

Anotações	

1

Separando em Blocos

No último bloco, vamos habilitar e executar o puppet agent.

Bloco: Executando o Puppet Agent nos servidores

- name: Executando o Puppet Agent nos servidores

systemd:

name: puppet
state: started
enabled: yes

Anotações	



Criando a Role Puppet Agent

Primeiramente, vamos criar a estrutura da nossa role utilizando o Ansible Galaxy, e então, prosseguir com a criação da mesma.

- Acesse a pasta infraagil e conecte-se a máquina automation, utilizando o vagrant ssh:
 - # cd infraagil
 - # vagrant ssh automation
- Acesse o diretório /etc/ansible/roles:
 - # cd /etc/ansible/roles
- Crie a role puppet-agent:
 - # sudo ansible-galaxy init puppet-agent

notações	

12

Criando a Role Puppet Agent

Agora, vamos criar nossa tasks.

Gere o arquivo **main.yml** utilizando o modelo da pasta Aula 6.2:

sudo cp /vagrant/4525/arquivos/Aula\ 6.2/main.yml
puppet-agent/tasks/

cat /etc/ansible/roles/puppet-agent/tasks/main.yml

Ou digite o conteúdo em notas, passando também as condicionais.

Anotações	



=== /etc/ansible/roles/puppet-agent/tasks/main.yml ====

```
# tasks file for puppet-agent
- name: Download e instalação do Puppet Agent - Ubuntu
 block:
 - get url:
   url: https://apt.puppetlabs.com/puppet6-release-bionic.deb
   dest: /tmp/puppet6.deb
 - apt:
   deb: /tmp/puppet6.deb
 - apt:
   name: puppet-agent
   state: present
   update_cache: yes
 when: ansible_distribution == 'Ubuntu'
- name: Download e instalacao do Puppet Agent - Debian
 block:
 - get url:
   url: https://apt.puppetlabs.com/puppet-release-buster.deb
   dest: /tmp/puppet6.deb
 - apt:
   deb: /tmp/puppet6.deb
 - apt:
   name: puppet-agent
   state: present
   update_cache: yes
 when: ansible distribution == 'Debian'
- name: Download e instalacao do puppet Agent - RedHat
 block:
 - yum:
   name: 'https://yum.puppet.com/puppet6-release-el-7.noarch.rpm'
   state: present
 - yum:
   name: puppet-agent
   state: present
 when: ansible_distribution == 'CentOS'
- name: Adicionando o binario do Puppet ao PATH
 block:
  - file:
    path: /root/.bashrc
    state: touch
  - lineinfile:
    path: /root/.bashrc
    line: 'PATH=/sbin:/bin:/usr/sbin:/usr/bin:/opt/puppetlabs/bin:/opt/puppetlabs/bin
- name: Adicionando configurações do puppet
 lineinfile:
  path: '/etc/puppetlabs/puppet/puppet.conf'
  line: "{{ item }}"
 loop:
  - '[main]'
  - 'server = compliance.4labs.example'
  - 'runinterval = 1h'
- name: Executando o Puppet Agent nos servidores
 systemd:
  name: puppet
  state: restarted
  enabled: yes
```



Criando a Role Puppet Agent

Agora, precisamos criar nossa playbook para executar a role puppet-agent.

Gere o arquivo **puppet-agent.yml** utilizando o modelo da pasta Aula 6.2:

- # sudo cp /vagrant/4525/arquivos/Aula\ 6.2/puppet-agent.yml
 /etc/ansible/playbooks/
- # cat /etc/ansible/playbooks/puppet-agent.yml
- name: Instalando e configurando o Puppet Agent

hosts: all roles:

- puppet-agent

Execute a playbook:

sudo ansible-playbook /etc/ansible/playbooks/puppet-agent.yml

notações	

4LINU2	Recapitulando	15
1	Entendendo a estrutura	
2	Separando em Blocos	
3	Criando a Role do Puppet Agent	

Anotações	

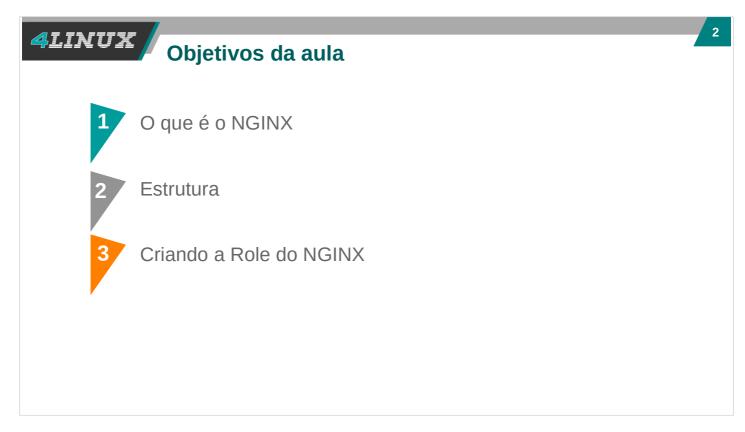




Construindo a infraestrutura como código

Construindo a Role nginx

Anotações	



Anotações		

O que é o NGINX

O NGINX é um servidor web conhecido por sua alta performance, estabilidade, funcionalidades, simples configuração e baixo consumo de recurso.



Anotações	



O NGINX possui alguns diretórios importantes para seu funcionamento:

sites-available

Diretório onde ficam localizados todos arquivos de configuração dos Virtual Hosts, disponíveis para serem habilitados.

sites-enabled

Diretório onde ficam os Virtual Hosts habilitados, normalmente são utilizados links simbólicos para arquivos localizados no diretório sites-available.

notações	

Estrutura

O NGINX utiliza arquivos de Virtual Host para configurar o destino de uma requisição que o mesmo receba.

```
server {
    listen 80;
    root /var/www/html;
    index index.html index.htm
    location / {
        try_files $uri $uri/ =404;
    }
}
```

Anotações		

Criando a Role NGINX

Primeiramente, vamos criar a estrutura da nossa role utilizando o Ansible Galaxy, e então, prosseguir com a criação da mesma.

- Acesse a pasta infraagil e conecte-se a máquina automation, utilizando o vagrant ssh:
 - # cd infraagil
 - # vagrant ssh automation
- Acesse o diretório /etc/ansible/roles:
 - # cd /etc/ansible/roles
- Crie a role nginx:
 - # sudo ansible-galaxy init nginx

notações	

Jose Luiz Bento Da Silva / luizbsilva@gmail.com	4 <u>1.</u>	1
	1	;
®gmail.com	2	7

<u>41.</u>	Criando a Role NGINX
	Vamos criar nossa task.
1	<pre>Crie o arquivo index.html: # sudo vim nginx/files/index.html <h1> Bem vindo a 4labs </h1></pre>
2	Gere o arquivo main.yml utilizando o modelo da pasta Aula 6.3: # sudo cp /vagrant/4525/arquivos/Aula\ 6.3/main.yml nginx/tasks/ # cat nginx/tasks/main.yml

Anotações		



=== /etc/ansible/roles/nginxt/tasks/main.yml ====

tasks file for nginx

- name: Instalando o NGINX

package:

name: nginx

state: present

- name: Removendo website padrao

file:

path: "{{ item }}"

state: absent

loop:

- /var/www/html/index.nginx-debian.html
- /var/www/html/index.html
- /var/www/html/index.html

- name: Copiando website 4labs

copy:

src: index.html

dest: /var/www/html/index.html

- name: Habilitando e reiniciando o serviço do NGINX

service:

name: nginx

state: restarted

enabled: yes

Criando a Role NGINX

Agora, vamos criar nossa playbook.

Gere o arquivo **nginx.yml** utilizando o modelo da pasta Aula 6.3:

sudo cp /vagrant/4525/arquivos/Aula\ 6.3/nginx.yml
/etc/ansible/playbooks/

cat /etc/ansible/playbooks/nginx.yml

- name: Aplicando as roles de log

hosts: log roles:

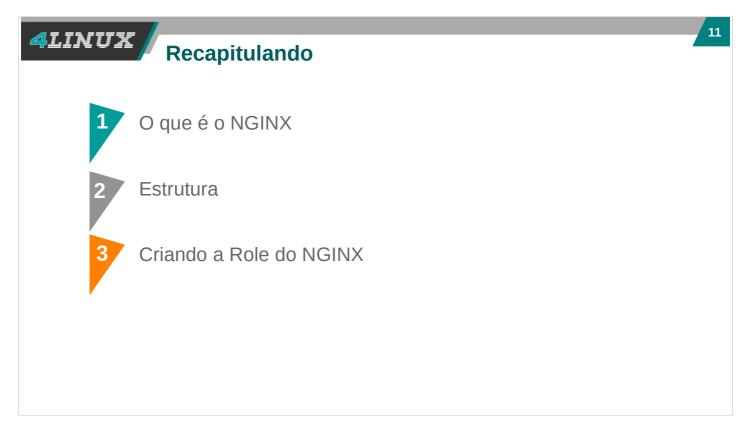
- nginx

Execute a playbook:

sudo ansible-playbook /etc/ansible/playbooks/nginx.yml

Anotações		





Anotações		





Construindo a infraestrutura como código

Construindo a Role iSCSI

Anotações	

4LINU2	Objetivos da aula	2
1	O que é o iSCSI	
2	Z Estrutura	
3	Criando a Role	

Anotações		

O que é iSCSI

iSCSI (Internet SCSI) é um protocolo de transporte para comandos SCSI (Small Computer System Interface) entre um computador anfitrião (initiator) e um dispositivo de destino (target). É tipicamente utilizado no contexto de SAN (Storage Area Network) e funciona sobre uma rede IP.



O iSCSI se difere dos outros modelos de compartilhamento de arquivos como CIFS e NFS, principalmente por entregar um disco diretamente sem particionamento.

O acesso aos dados de Storage iSCSI é um block level (nível de bloco), isso significa que o iSCSI Target serve um iSCSI Initiator (host). Tal característica é a mesma quando o S.O. lê o protocolo SCSI, o sistema operacional interpreta o protocolo e o identifica como um novo disco. Esse armazenamento compartilhado em blocos, vem sendo muito utilizado em pequenas empresas, pois sua configuração é mais fácil e os custos de implementação e manutenção são mais baratos, em comparação a um Storage SAN (Storage Area Network).

Vantagens do Storage iSCSI:

- ✓ * Solução para armazenamento compartilhado para bancos de dados como SQL e Oracle, alteração ou instalação de sistemas de compartilhamento de arquivos, baseados em servidores;
- ✓ * Otimização do espaço em disco: o usuário pode aumentar a capacidade de armazenamento, ou então, diminuí-lo conforme julgar necessário;
- ✓ * Dispositivo de armazenamento semelhante ao sistema ZFS, dispositivo de arquivamento para o sistema operacional Linux;
- ✓ * Backup com espelhamento, função que consiste na gravação dos discos rígidos;
- ✓ * Função autorreparação que identifica e elimina falhas, sem comprometer a produtividade;
- ✓ * Iniciadores e alvos são identificados pelo Protocolo CHAP (Challenge Handshake Authentication Protocol), autenticação que verifica a identidade do operador do sistema.

4LINUX O que é iSCSI

- ✓ O iSCSI diferente do NFS (Network File System) e do CIFS (Common Internet File System), não entrega um diretório e sim uma LUN (Logical Unit Number) com um dispositivo de bloco por completo;
- ✓O initiator poderá então utilizar o storage do target iSCSI como se fosse um disco local e inclusive criar partições neste disco.

Anotações	

Estrutura

Teremos que configurar nossa playbook para instalar e configurar o iscsi, o target e o initiator;

Após essa configuração, vamos também formatar partições para que o disco seja utilizável.

Anotações	



Criando a Role iSCSI

Primeiramente, vamos criar a estrutura da nossa role utilizando o Ansible Galaxy, e então, prosseguir com a criação da mesma.

- Acesse a pasta infraagil e conecte-se a máquina automation, utilizando o vagrant ssh:
 - # cd infraagil
 - # vagrant ssh automation
- Acesse o diretório /etc/ansible/roles:
 - # cd /etc/ansible/roles
- Crie a role iscsi:
 - # sudo ansible-galaxy init iscsi

notações	

Criando a Role iSCSI

Vamos criar nossa tasks.

```
Gere o arquivo main.yml utilizando o modelo da pasta Aula 6.4:
# sudo cp /vagrant/4525/arquivos/Aula\ 6.4/main.yml
iscsi/tasks/
# cat iscsi/tasks/main.yml
---
# tasks file for iscsi
- name: Instalando o tgt e configurando o iscsi
block:
    package:
(...)
```

Anotações	



```
=== /etc/ansible/roles/puppet-agent/tasks/main.yml ====
# tasks file for iscsi
- name: Instalando o tgt e configurando o iscsi
 block:
 - package:
   name: scsi-target-utils
   state: present
 - blockinfile:
   create: yes
   path: /etc/tgt/conf.d/4labs_scsi.conf
   block: |
     <target iqn.2019-05.automation.4labs.example:lun>
      backing-store /dev/sdb
     </target>
 - service:
   name: tgtd
   state: restarted
   enabled: yes
 when: ansible hostname == "automation"
- name: Instalando e configurando o openiscsi
 block:
 - package:
   name: ["open-iscsi", "parted", "xfsprogs", "btrfs-progs"]
   state: present
 - open iscsi:
   show_nodes: yes
   discover: yes
   portal: 10.5.25.10
  register: iscsi
 - open_iscsi:
   login: yes
   target: "{{ iscsi.nodes[0] }}"
 - replace:
   path: "/etc/iscsi/nodes/{{ iscsi.nodes[0] }}/10.5.25.10,3260,1/default"
   regexp: "node.startup = manual"
   replace: "node.startup = automatic"
 - service:
   name: open-iscsi
   state: restarted
   enabled: yes
 when: ansible_hostname == "log"
(...)
```



=== /etc/ansible/roles/puppet-agent/tasks/main.yml ====

(...)

- name: Criando partiões no Disco

block: - parted:

device: /dev/sdb

number: 1 state: present part_end: 50%

- parted:

device: /dev/sdb

number: 2 state: present part_start: 50%

filesystem: fstype: ext4 dev: /dev/sdb1

filesystem: fstype: ext3 dev: /dev/sdb2

when: ansible_hostname == 'log'



Criando a Role iSCSI

Agora, vamos criar nossa playbook.

Gere o arquivo iscsi.yml utilizando o modelo da pasta Aula 6.4:

- # sudo cp /vagrant/4525/arquivos/Aula\ 6.4/iscsi.yml
 /etc/ansible/playbooks/
- # cat /etc/ansible/playbooks/iscsi.yml
- name: Instalando o iSCSI hosts: automation log roles:
 - iscsi
- Execute a playbook:
 - # sudo ansible-playbook /etc/ansible/playbooks/iscsi.yml

Anotações		

- -		
I D	<u>41.</u>	INU2
5		
		Para ve
5 _	1	Verifiqu

Criando a Role iSCSI

Para verificar se tudo ocorreu corretamente, podemos usar o comando iscsiadm.

- 1 Verifique se o target foi mapeado na VM Log:
 - # sudo ansible log -m shell -a "iscsiadm -m session"
- Verifique se o bloco sdc1 e sdc2 foram criados na VM Log:
 # sudo ansible log -m shell -a "lsblk"

Anotações	

4LINUX	Recapitulando	12
1	O que é o iSCSI	
2	Estrutura	
3	Criando a Role	

Anotações		

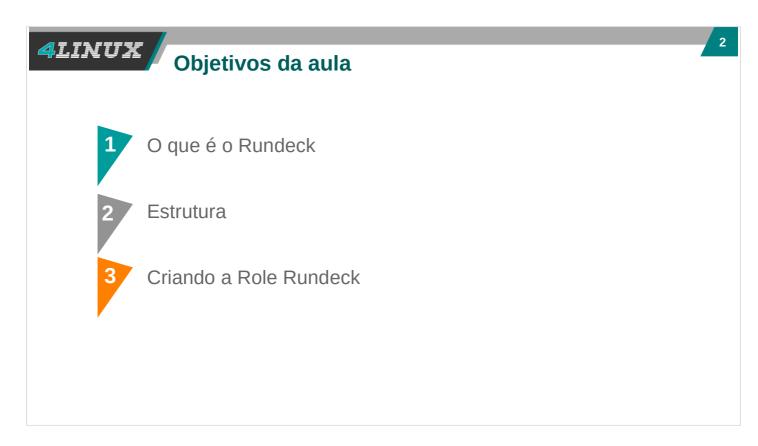




Construindo a Infraestrutura como Código

Construindo a Role Rundeck

Anotações	



Anotações		

O que é o Rundeck

O Rundeck é um software de código aberto para automatização de processos e rotinas de operação ampla, utilizado em ambientes DevOps de Entrega Contínua.



Anotações		

Estrutura

O Rundeck possui alguns arquivos importantes para seu funcionamento e parametrização.

framework.properties

Arquivo de configuração utilizado pelo core do Serviço do Rundeck, onde são configurados o hostname, porta de conexão, dentre outras configurações.

rundeck-config.properties

Arquivo onde é configurado o webapp do Rundeck, define o nível padrão de log, origem de dados e customização da interface gráfica.

Anotações	

Estrutura

Vamos modificar alguns campos no **framework.properties** para que o nosso rundeck responda pelo nome e porta corretos, por padrão, o rundeck responde na porta 4440, iremos manter esta configuração.

framework.properties

framework.server.hostname = automation.4labs.example

framework.server.name = 4labs

framework.server.port = 4440

framework.server.url = http://automation.4labs.example:4440

Anotações	



Estrutura

Temos também que modificar um campo chamado **grails.serverURL** no arquivo rundeck-config.properties.

rundeck-config.properties

grails.serverURL=http://automation.4labs.example:4440

Anotações	

Criando a Role Rundeck

Primeiramente, vamos criar a estrutura da nossa role utilizando o Ansible Galaxy, e então, prosseguir com a criação da mesma.

- Acesse a pasta infraagil e conecte-se a máquina automation, utilizando o vagrant ssh:
 - # cd infraagil
 - # vagrant ssh automation
- Acesse o diretório /etc/ansible/roles:
 - # cd /etc/ansible/roles
- Crie a role rundeck:
 - # sudo ansible-galaxy init rundeck

notações	

Criando a Role Rundeck

```
Gere o arquivo main.yml utilizando o modelo da pasta Aula 6.5:

# sudo cp /vagrant/4525/arquivos/Aula\ 6.5/main.yml rundeck/tasks/

# cat rundeck/tasks/main.yml
---

# tasks file for rundeck
- name: Instalacao das Dependencias e Repositorio Rundeck
yum:
    name: "{{ item }}"
    state: present
loop:
(...)
```

notações	



```
=== /etc/ansible/roles/rundeck/tasks/main.yml ===
# tasks file for rundeck
- name: Instalacao das Dependencias e Repositorio Rundeck
 yum:
  name: "{{ item }}"
  state: present
 loop:
  - vim
  - java-1.8.0
  - https://repo.rundeck.org/latest.rpm
- name: Instalação do Rundeck
 yum:
  name: rundeck
  state: present
- name: Configurando Rundeck
 lineinfile:
  path: "{{ item.path }}"
  regexp: "{{ item.regexp }}"
  line: "{{ item.line }}"
 loop:
       - {path: '/etc/rundeck/framework.properties', regexp: '^framework.server.name', line:
   'framework.server.name = 4labs'}
      - {path: '/etc/rundeck/framework.properties', regexp: '^framework.server.hostname', line:
   'framework.server.hostname = automation.4labs.example'}
        - {path: '/etc/rundeck/framework.properties', regexp: '^framework.server.port', line:
   'framework.server.port = 4440'}
         - {path: '/etc/rundeck/framework.properties', regexp: '^framework.server.url',
                                                                                             line:
   'framework.server.url = http://automation.4labs.example:4440'}
              {path: '/etc/rundeck/rundeck-config.properties', regexp: '^grails.server',
                                                                                             line:
   'grails.serverURL=http://automation.4labs.example:4440'}
- name: Iniciando e Habilitando o serviço do rundeck
 systemd:
  name: rundeckd
  state: started
  enabled: yes
```

Criando a Role Rundeck

Agora, vamos criar nossa playbook.

Gere o arquivo **rundeck.yml** utilizando o modelo da pasta Aula 6.5:

sudo cp /vagrant/4525/arquivos/Aula\ 6.5/rundeck.yml
/etc/ansible/playbooks/

cat /etc/ansible/playbooks/rundeck.yml

 name: Instalação do Rundeck hosts: automation

roles:

- rundeck

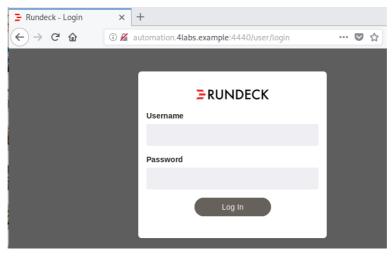
Execute a playbook:

sudo ansible-playbook /etc/ansible/playbooks/rundeck.yml

Anotações		

Criando a Role Rundeck

Acesse pelo navegador o endereço http://automation.4labs.example:4440.



automation.4labs.example

Anotações



Anotações	





Construindo a Infraestrutura como Código

Construindo a Role Docker

Anotações	

4LINU2	Objetivos da aula	
1	O que é o Docker	
2	Estrutura	
3	Criando a Role Docker	

Anotações	

O que é o Docker

Docker é uma plataforma Open Source escrita em Go, que facilita a criação e Administração de Ambientes Isolados.



Anotações	

4LINUX Estrutura

A Solução Docker será composta por alguns pacotes:

- ✓ docker-ce → Docker Comunity Edition, é o daemon do Docker;
- ✓ docker-ce-cli → Docker Command Line Interface, é a interface de terminal para utilizar o docker;
- ✓ containerd.io → É um runtime de contêineres com ênfase em simplicidade, rubustez e portabilidade;
- ✓ Docker Machine Bash Completion → São os recursos de auto completar do docker.

Anotações	

.

Criando a Role Docker

Primeiramente, vamos criar a estrutura da nossa role utilizando o Ansible Galaxy, e então, prosseguir com a criação da mesma.

- Acesse a pasta infraagil e conecte-se a máquina automation, utilizando o vagrant ssh:
- # cd infraagil
 - # vagrant ssh automation
- Acesse o diretório /etc/ansible/roles:
 - # cd /etc/ansible/roles
- Crie a role docker:
 - # sudo ansible-galaxy init docker

Anotações	

4LINUX Criando a Role Rundeck

```
Gere o arquivo main.yml utilizando o modelo da pasta Aula 6.6:
# sudo cp /vagrant/4525/arquivos/Aula\ 6.6/main.yml
docker/tasks/
# cat docker/tasks/main.yml
# tasks file for docker
- name: Instalacao das Dependencias
  yum:
    name: "{{ item }}"
    state: present
(...)
```

notações	



```
=== /etc/ansible/roles/docker/tasks/main.yml ===
# tasks file for docker
- name: Instalacao das Dependencias
 yum:
  name: "{{ item }}"
  state: present
 loop:
  - epel-release
  - yum-utils
  - device-mapper-persistent-data
  - curl
  - lvm2
  - vim
  - bash-completion
- name: Adicionar repositório do Docker-Engine
 get url:
  url: https://download.docker.com/linux/centos/docker-ce.repo
  dest: /etc/yum.repos.d/docer-ce.repo
- name: Instalar o Docker-Engine
 yum:
  name: "{{ item }}"
  state: present
 loop:
  - docker-ce
  - docker-ce-cli

    containerd.io

- name: Instalar o Docker Machine bash completion
 get url:
   url:
         https://raw.githubusercontent.com/docker/machine/v0.16.0/contrib/completion/bash/docker-
   machine.bash
  dest: /etc/bash completion.d/docker-machine
- name: Habilitando e iniciando o serviço do Docker-ce
 systemd:
  name: docker
  state: started
  enabled: yes
- name: Instalando o docker-compose
 get url:
  url: https://github.com/docker/compose/releases/download/1.24.0/docker-compose-Linux-x86 64
```

Documento produzido pela 4Linux

mode: 0755

dest: /usr/local/bin/docker-compose

8

Criando a Role Docker

Agora, vamos criar nossa playbook:

Gere o arquivo docker.yml utilizando o modelo da pasta Aula 6.6:

sudo cp /vagrant/4525/arquivos/Aula\ 6.6/docker.yml
/etc/ansible/playbooks/

cat /etc/ansible/playbooks/docker.yml

- name: Instalação do Docker

hosts: container

roles:

- docker

Execute a playbook:

sudo ansible-playbook /etc/ansible/playbooks/docker.yml

Anotações		

Anotações	

4LINU2	Recapitulando	10
1	O que é o Docker	
2	Estrutura	
3	Criando a Role Docker	

Anotações			