Universidad ORT Uruguay

Facultad de Ingeniería

Bernard Wand Polak

Obligatorio
Taller de Servidores Linux

Agosto 2023

Vladimir Demari N° - 291649

Sebastián Rodríguez N° - 166526

Docente: Enrique Verdes

Contenido

Introducción del Proyecto

Este proyecto consta de la creación de tres servidores.

El primero llamado Bastion, cuya IP es 192.168.0.10 y con sistema operativo Rocky server, será nuestra interfaz grafica desde donde administraremos los otros dos servidores.

En el implementaremos el formato Workstation, para tener interfaz gráfica.

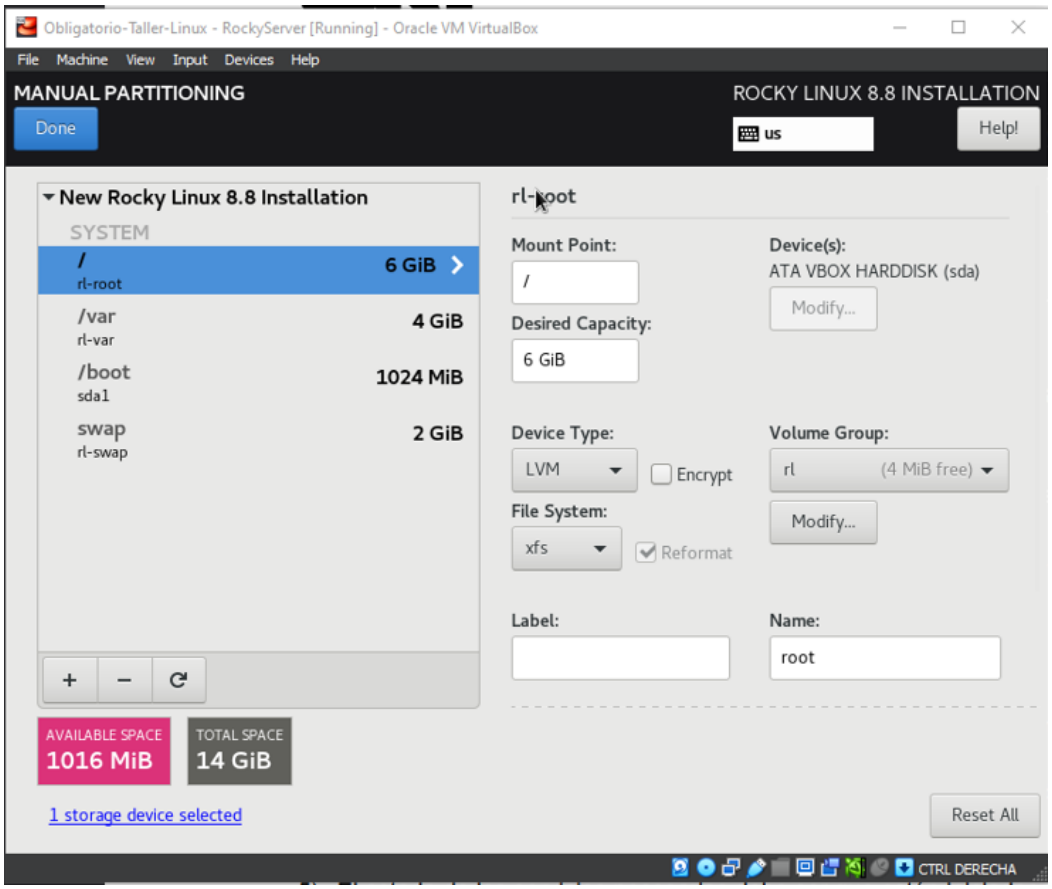
A su vez crearemos un servidor llamado RockyServer, cuya IP es 192.168.0.11, también con Linux Rocky server. Para este caso utilizaremos el formato “minimal” para reducir funcionalidad al mínimo, instalando nosotros los aplicativos necesario. El que sea “minimal” nos ayuda a reducir el acceso a personas que no tengan conocimientos de administración por línea de comandos.

Instalaremos un servidor con la distribución de Linux Ubuntu, que se llamara UbuntuServer, cuya IP es 192.168.0.12, en formato “minimal” con el mismo objetivo que para nuestro Rocky server. Reducir los servicios al mínimo necesario, y acotar las posibilidades de acceso.

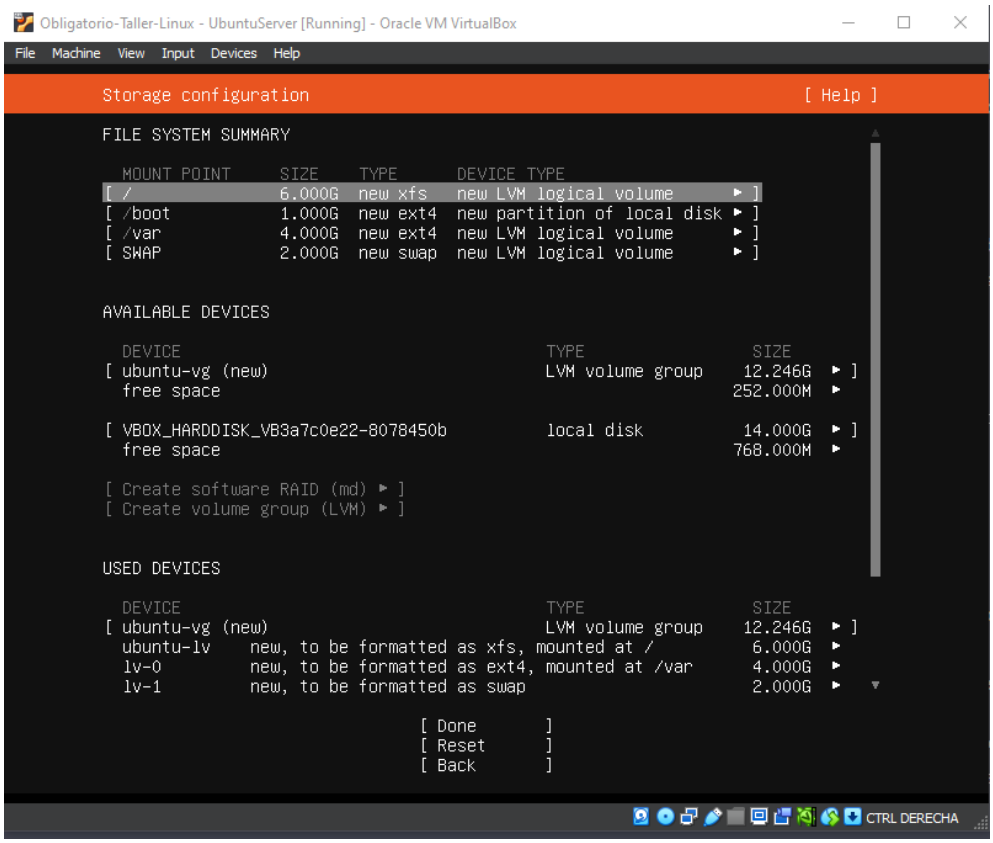
Particionado de disco

El proyecto indica que los servidores deben cumplir con las siguientes caracteristicas de particionado de disco:

Rocky Server



Ubuntu Server



Configuración de servidores

Generamos las claves en el Bastion las cuales utilizaremos para acceder a los servers y al repositorio en github.

```
[sysadmin@bastion ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/sysadmin/.ssh/id_rsa):
Created directory '/home/sysadmin/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/sysadmin/.ssh/id_rsa.
Your public key has been saved in /home/sysadmin/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:wZRyPjWMJDyP7aH1h+SofwcbvGO05o6yW6V/vSVcgLI sysadmin@bastion.ejemplo.com.uy
The key's randomart image is:
+---[RSA 3072]----+
|      ...o+      |
|       +++ +  .  |
|        Oo.... . |
|       . B..o  . |
|        +SOE.   .|
|       . ooO .. . |
|        .o. B .o .|
|       o. oO o .o |
|       o=o*++  .. |
+----[SHA256]-----+
[sysadmin@bastion ~]$
```

Agregamos al usuario Ansible en los servidores Rockyserver y Ubuntuserver, el cual utlizaremos para acceder remotamente con Ansible.

Agregaremos este usuarios al grupo “sudo” y le daremos permiso de ejecutar “sudo” sin contraseña.

En Rockyserver

```
[sysadmin@rockyserver ~]$ sudo adduser ansible
[sudo] password for sysadmin:
[sysadmin@rockyserver ~]$ sudo usermod -aG wheel ansible
[sysadmin@rockyserver ~]$ echo "ansible ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/ansible
ansible ALL=(ALL) NOPASSWD: ALL
[sysadmin@rockyserver ~]$
```

En Ubuntuserver

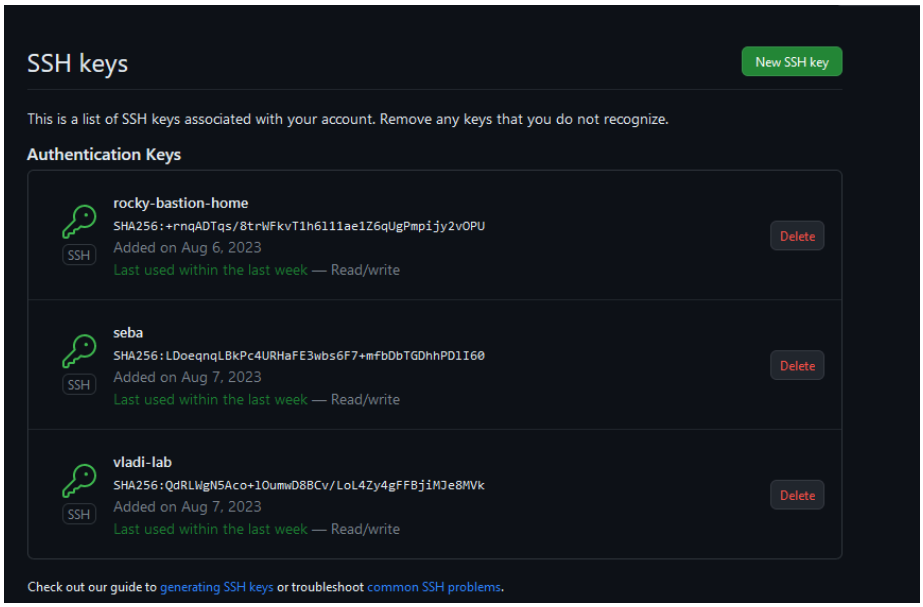
```
sysadmin@ubuntuserver:~$ sudo adduser ansible
[sudo] password for sysadmin:
Adding user `ansible' ...
Adding new group `ansible' (1001) ...
Adding new user `ansible' (1001) with group `ansible' ...
Creating home directory `/home/ansible' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for ansible
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
sysadmin@ubuntuserver:~$ sudo usermod -aG sudo ansible
sysadmin@ubuntuserver:~$ echo "ansible ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/ansible
ansible ALL=(ALL) NOPASSWD: ALL
```

Luego copiamos la llave pública desde el bastión hacia los servidores, para que el usuario ansible pueda conectarse sin utilizar contraseña.

```
[sysadmin@bastion ~]$ ssh-copy-id ansible@192.168.0.11
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
ansible@192.168.0.11's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'ansible@192.168.0.11'"
and check to make sure that only the key(s) you wanted were added.
```

```
[sysadmin@bastion ~]$ ssh-copy-id ansible@192.168.0.12
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
ansible@192.168.0.12's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'ansible@192.168.0.12'"
and check to make sure that only the key(s) you wanted were added.
```

Keys en Github



Luego realizamos la instalación de Git y Ansible. En el caso de Git ya lo teníamos instalado en nuestro Rocky bastion.

```
[sysadmin@bastion ~]$ sudo dnf install git
[sudo] password for sysadmin:
Last metadata expiration check: 21:55:20 ago on Fri 04 Aug 2023 12:35:09 AM -03.
Package git-2.39.3-1.el8_8.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

```
[sysadmin@bastion ~]$ sudo dnf install ansible
[sudo] password for sysadmin:
Last metadata expiration check: 11:42:37 ago on Fri 04 Aug 2023 11:16:42 PM -03.
Dependencies resolved.
================================================================================
 Package                                                    Architecture
================================================================================
Installing:
 ansible                                                    noarch
Installing dependencies:
 ansible-core                                               x86_64
 mpdecimal                                                  x86_64
```

Clonamos nuestro repositorio previamente creado en Github.

```
[sysadmin@bastion ~]$ git clone git@github.com:vladimirdemari/Obligatorio-ASLX.git
Cloning into 'Obligatorio-ASLX'...
```

Luego creamos los archivos de configuración para nuestro proyecto

- ansible.cfg

```
$ ansible-config init --disabled > ansible.cfg
```

```
# (pathlist) Comma separated list of Ansible inventory sources
inventory=/home/sysadmin/Obligatorio-ASLX/inventario
```

- inventario

```
[redhat]
rockyserver     ansible_host=192.168.0.11

[debian]
ubuntuserver    ansible_host=192.168.0.12

[linux:children]
redhat
debian
```

- requeriment

```
# Roles requeridos
- name: geerlingguy.apache
  src: https://github.com/geerlingguy/ansible-role-apache
~
```

Instalamos el este rol

```
[sysadmin@bastion Obligatorio-ASLX]$ vim requirements.yml
[sysadmin@bastion Obligatorio-ASLX]$ ansible-galaxy install -r requirements.yml -p roles/
Starting galaxy role install process
- extracting geerlingguy.apache to /home/sysadmin/Obligatorio-ASLX/roles/geerlingguy.apache
- geerlingguy.apache was installed successfully
```

Playbook en Ansible

Decidimos crear una estructura de 3 bloques en nuestro main.yml principal. En cada estructura aplicaremos tareas a los diferentes Hosts definidos en el inventario.

Los bloques serán los siguientes:

- Linux: correrán tareas genéricas para todos los Linux
- RedHat: tareas para los servers Redhat
- Debian: tareas para los servers Debian

Linux:

```yaml
---
- name: Tareas Generales en Servidores Linux
  hosts: linux
  become: yes
  remote_user: ansible

  tasks:
  - name: Update RedHat Servers
    yum:
      name: "*"
      state: latest
    notify: Restart
    when: ansible_os_family == "RedHat"

  - name: Update Debian Servers
    apt:
      name: "*"
      state: latest
      update_cache: yes
    notify: Restart
    when: ansible_os_family == "Debian"

  handlers:
  - name: Restart
    reboot:
```

RedHat:

```yaml
- name: Configuracion de Servidores RedHad
  hosts: redhat
  become: yes
  remote_user: ansible

  roles:
    - geerlingguy.apache
    - container_tomcat

  tasks:

  - name: Open firewall ports
    firewalld:
      service: "{{ item }}"
      state: enabled
      permanent: true
      immediate: true
    loop:
    - http
    - https
```

Debian:

```yaml

- name: Configuracion de Servidores Debian
  hosts: debian
  become: yes
  remote_user: ansible

  roles:
    - mariadb_verdes
    - create.database
```

Roles Ansible Galaxy

En la estructura de carpetas contamos con 4 roles donde definiremos los roles y servicios que necesitamos instalar y configurar.

geerlingguy.apache

Se trata de un rol creado por geerlingguy.apache el cual instala y realiza la configuración básica de un servidor apache. Modificamos una parte del código para incluir nuestra configuración de vhot para el reverse proxy.

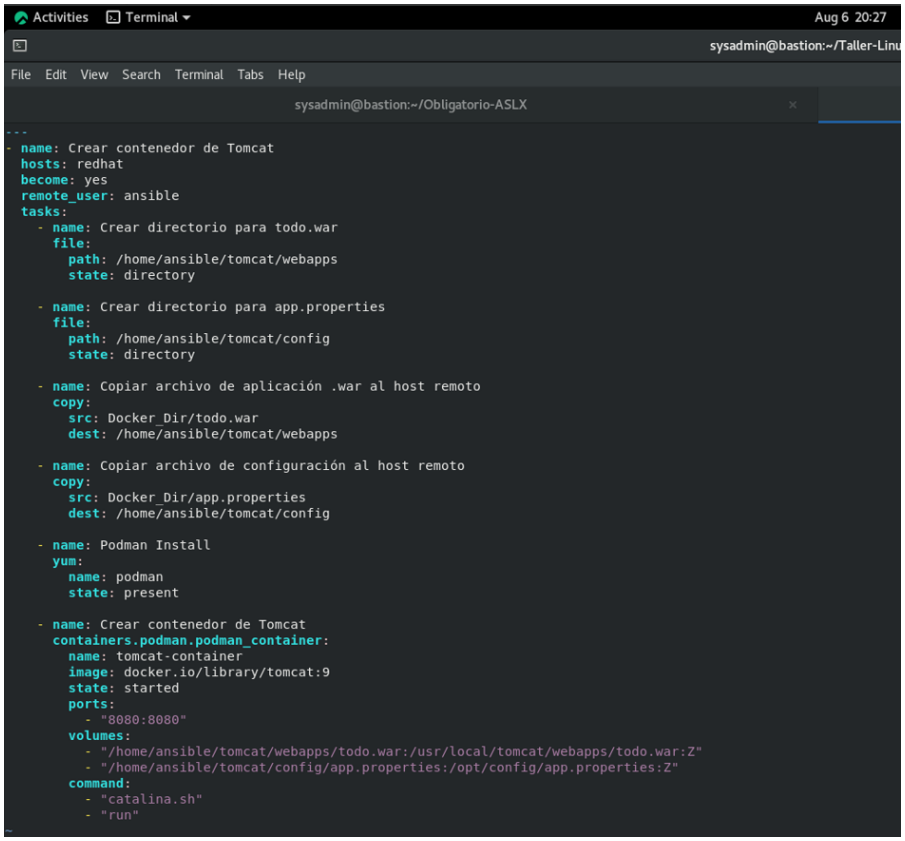
Creamos tareas en el playbook principal para configurar el reverse proxy.

```
- name: Create virtualhost configuration directory
  file:
    path: /etc/httpd/vhosts.d
    state: directory
    owner: root

- name: Copy virtualhost configuration
  copy:
    src: roles/geerlingguy.apache/files/virtualhost.conf
    dest: /etc/httpd/vhosts.d/ejemplo.conf
  notify: Restart apache

- name: Include vhosts directory
  lineinfile:
    path: /etc/httpd/conf/httpd.conf
    line: IncludeOptional /etc/httpd/vhosts.d/*.conf

handlers:

- name: Restart apache
  service:
    name: httpd
    state: restarted
```

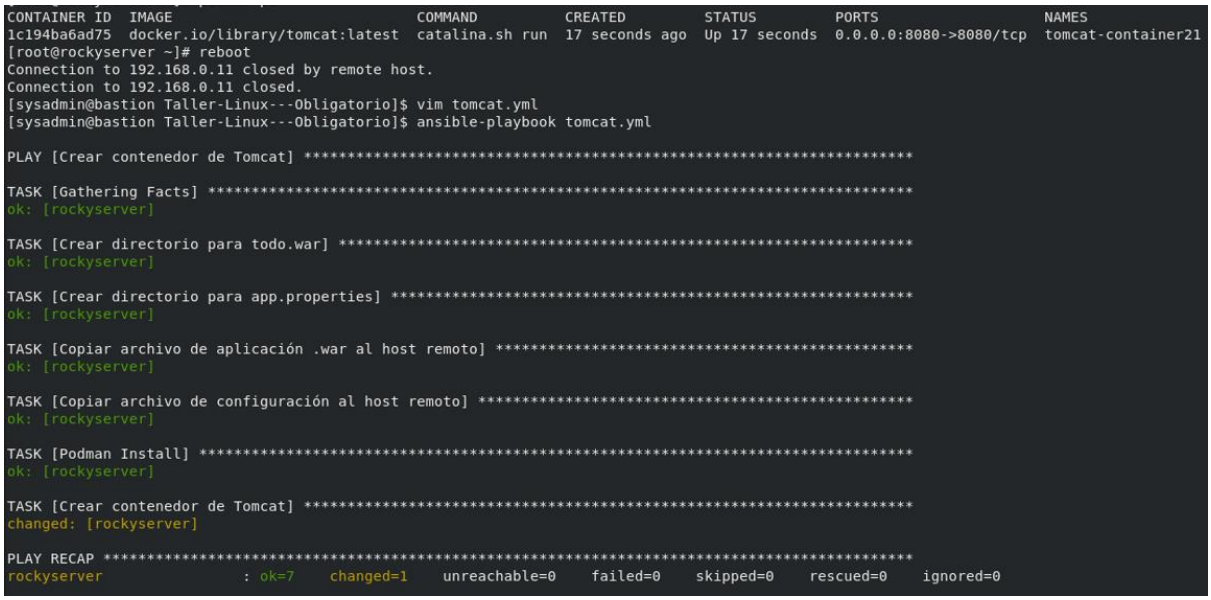
container_tomcat

Se trata de un Rol creados por nosotros para implementar un contenedor tomcat con una la aplicación todo.war corriendo en el mismo.
Las tareas que realizan son, copiar archivos necesarios para nuestra app al server anfitrión, instalar podman en el servidor Redhat y correr el módulo “containers.podman.podman_container” para la creación de la imagen del contenedor y la ejecución del mismo. Este modulo en lugar de copiar los archivos necesarios de la app mapea los mismos desde el anfitrión.

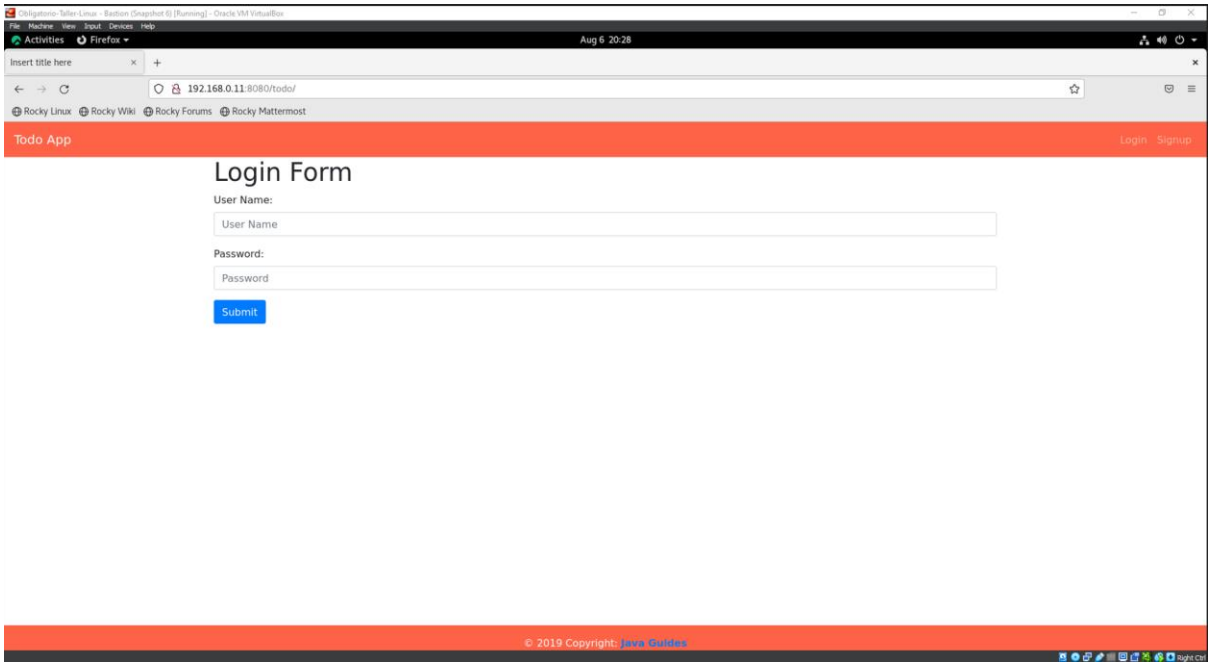
Playbook:



Ejecución de playbook:



Web todo.war



mariadb_verdes

En este rol incluimos el playbook visto en clase, el cual instala Mariadb, Python3, realiza la configuración inicial de Mariadb y configuración de seguridad.
A su vez, realiza la configuración del Firewall de Ubuntu y abriendo el puerto necesario y harderiza la configuración del usuario root.

create.database

Por un tema de orden decidimos crear este rol para incluir la configuración de nuestra base de datos. En las tareas de este rol creamos la base de datos y sus tablas, mediante un dump de la base de datos del obligatorio anterior. También creamos el usuario todo el cual utilizará la appweb para conectarse.

Playbook:

```yaml
---
# tasks file for roles/create.database

- name: Copiar archivo dump de la base de datos
  copy:
    src: roles/create.database/files/todo.sql
    dest: /home/ansible/todo.sql


- name: Restore database
  community.mysql.mysql_db:
    login_host: localhost
    login_user: root
    login_password: root
    name: todo
    state: import
    target: /home/ansible/todo.sql

- name: Creamos usuario todo
  mysql_user:
    login_host: localhost
    login_user: root
    login_password: root
    name: todo
    password: todo1234
#    encrypted: yes
    priv: '*.*:ALL,GRANT'
    state: present
```

Archivo dump para Mariadb:

```
-- MySQL dump 10.19  Distrib 10.3.35-MariaDB, for Linux (x86_64)
--
-- Host: localhost    Database: todo
-- ------------------------------------------------------
-- Server version       10.3.35-MariaDB

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `todos`
--

DROP TABLE IF EXISTS `todos`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `todos` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `description` varchar(255) DEFAULT NULL,
  `is_done` bit(1) NOT NULL,
  `target_date` datetime(6) DEFAULT NULL,
  `username` varchar(255) DEFAULT NULL,
  `title` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8mb4;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `todos`
--

LOCK TABLES `todos` WRITE;
/*!40000 ALTER TABLE `todos` DISABLE KEYS */;
/*!40000 ALTER TABLE `todos` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `users`
--
```

Files en los roles

Tanto en el rol container_tomcat como en el rol create.database tienen archivos en sus respectivas carpetas /files.

En el caso de container_tomcat tenemos el app.properties y el todo.war.

En el créate.database tenemos el archivo todo.sql para el dump de la base de datos.

Referencias

https://www.redhat.com/sysadmin/ansible-podman-container-deployment

https://galaxy.ansible.com/bertvv/mariadb

https://galaxy.ansible.com/containers/podman

https://docs.ansible.com/ansible/latest/collections/containers/podman/podman_image_module.html

https://docs.ansible.com/ansible/latest/collections/community/mysql/mysql_db_module.html

CHATGPT

BARD