Homework

Instaling Jenkins with Docker in ubuntu 20.04 trough SSH without a password

Ubuntu

Let's begin the installation with Vagrand through windows terminal in this case using PowerShell to make this task once you Vagrant up, to start running, I use SSH to start the configurations needed.

Once you're in the terminal you must change to the root directory and start update, to have the files in order let's create a new personal user in this occasion, I use Pablo my name, after that I use usermod AG sudo Pablo.

Let's see that our user Pablo has the sudo command activate it then it's time to configure the SSH in that way we can use it from windows terminal to connect it directly, we restart the service then go to the windows Terminal to connect directly without the Vagrand, using Pablo@127.0.0.1 in the port 2222 once you're there, they will ask you for a password. Now we are going to make the SSH to connect without a password we're making the .SSH files needed and directory that's it's intended to use, we're given the permissions needed then we're going to create an SSH key we are using the default method to do that, after during this configuration we going to copy the key created into the windows authorized case and now hosts, then we're going to take out the password needed to connect if there is an edge SSH configurated then restart the service and try to connect from windows to boot with their SSH and without a password needed and like you could see in the next pictures that is attached to the document with all the steps needed to complete the tasks.

```
Adding new user `pablo' (1003) with group `pablo' ...
Creating home directory `/home/pablo' ...
Copying files from `/etc/skel' ...
New password:

Retype new password:
passwd: password updated successfully
Changing the user information for pablo
Enter the new value, or press ENTER for the default
        Full Name []: Juan Pablo Bedoya
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
root@ubuntu-focal:/home/jpablo# usermod -aG sudo pablo
root@ubuntu-focal:/home/jpablo# exit
exit
jpablo@ubuntu-focal:~$ exit
logout
vagrant@ubuntu-focal:~$ sudo su - pablo
```

```
PS C:\Users\PC\.ssh> ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\PC/.ssh/id_rsa):
C:\Users\PC/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter same passphrase again:
Your identification has been saved in C:\Users\PC/.ssh/id_rsa.
Your public key has been saved in C:\Users\PC/.ssh/id_rsa.pub.
The key fingerprint is:
The key's randomart image is:
+---[RSA 3072]----+
|         .o..    |
|      . oo.o .   |
|o  . o o..  o    |
|=E. o   .        |
|+o.   o.So       |
|.. o o +*o.      |
|  o . o+o+oo     |
|   . .  +.Boo.   |
|      .+*O.+B=.  |
+----[SHA256]-----+
```

```
PS C:\Users\PC\.ssh>  ssh -p 2222 pablo@127.0.0.1
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-105-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Apr  8 05:01:16 UTC 2022

  System load:  0.01              Processes:               172
  Usage of /:   4.4% of 38.71GB   Users logged in:         2
  Memory usage: 26%               IPv4 address for enp0s3: 10.0.2.15
  Swap usage:   0%

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

0 updates can be applied immediately.


Last login: Fri Apr  8 04:50:05 2022 from 10.0.2.2
pablo@ubuntu-focal:~$
```

Installing Docker and Jenkins

Now we're we going to stall install Jenkins using docker, first we have to install docker in our ubuntu to make this I'm going to put the commands needed to make this installation.

```
pablo@ubuntu-focal:~$ sudo apt-get install \
>     ca-certificates \
>     curl \
>     gnupg \
>     lsb-release
```

I've updated the system to have all the files in order then I install curl that is needed and certificates that the documentation in docker indicates that we have to do this to have a correct installation

```
pablo@ubuntu-focal:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor
 -o /usr/share/keyrings/docker-archive-keyring.gpg
pablo@ubuntu-focal:~$ echo \
>   "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.g
pg] https://download.docker.com/linux/ubuntu \
>   $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
pablo@ubuntu-focal:~$ sudo apt-get update
```

Finally, we're installing docker with sudo to get installed docker

```
pablo@ubuntu-focal:~$  sudo apt-get install docker-ce docker-ce-cli containerd.io
Reading package lists... Done
```

```
pablo@ubuntu-focal:~$ sudo docker run -p 80:8080 --name=jenkins-master jenkins/jenkins
Running from: /usr/share/jenkins/jenkins.war
```

UNIX ONLINE TEST

Total Questions – 20	14:31:77	Max Time – 20 Min

You scored 63.16%

Total Questions: 19, Attempted: 20, Correct: 12, Time Taken: 5.47 Min

---

Using the VirtualBox SDK of the language you choose, create a CLI tool to manage your virtual machines

---

```
import virtualbox
vbox = virtualbox.VirtualBox()
session = virtualbox.Session()

newVM = virtualbox.VirtualBox()

def showVirtualMachines():
    print([m.name for m in vbox.machines])

def standUpVM(nameVM):
    machine = vbox.find_machine(nameVM)
```

```
    #if Virtual is 6.1 and below use the next
    #progress = machine.launch_vm_process(session, "gui", "")
    #else
    # For virtualbox API 6_1 and above (VirtualBox 6.1.2+), use the following:
    progress = machine.launch_vm_process(session, "gui", [])
    progress.wait_for_completion()

def consultingVM():
    print(session.state)
    #If the answer is => SessionState(2)  # locked

def writeVM(sentence):
    session.console.keyboard.put_keys(sentence)

def loginVM(user, password):
    guest_session = session.console.guest.create_session(user, password)

def powerDownVM():
    session.console.power_down()

def createVM(nameVM):
#                pruebita2     =     createdVM.create_machine("/Users/PC/VirtualBox
VMs/ubuntu20/ubuntu20.vbox","PruebaUbuntu2",[],"ubuntu64Guest","")

    vm = newVM.create_machine("",nameVM,[],"ubuntu64Guest","")
    newVM.register_machine(vm)

def deleteVM(nameVM):
    machineBorrar = newVM.find_machine(nameVM)
    machineBorrar.remove()
```