Fabric is a Python library and command-line tool for streamlining the use of SSH for application deployment or systems administration tasks. Fabric is very simple and powerful and can help to automate repetitive command-line tasks. This approach can save time by automating your entire workflow.

STEPS: To SSH remotely using Fabric

1. Create a folder for your scripts named pyscripts
2. create your python script named fabfile for testing remotely
3. Setup your remote SSH access
4. Setup web service remotely using fabric

Let’s Start:

**Create a folder for your scripts named pyscripts in the main host**

1. mkdir pyscripts
2. cd pyscripts
3. install pip - wget <https://bootstrap.pypa.io/get-pip.py>
4. ls
5. install python2 – apt install python
6. Execute the script - python get-pip.py
7. install Fabric - pip install ‘fabric<2.0’ (fabric<2.0 means less than 2.0 version)
8. mkdir fabric
9. cd fabric

**TO EXECUTE TASKS ON THE LOCAL MACHINE**

1. vim fabfile.py

#!/user/bin/python3

from fabric.api import \*

def system\_info():

print(“Disk Space”)

local(“df –h”)

print(“Ram size”)

local(“free –m”)

print(“System uptime”)

local(“uptime”)

:wq

1. fab system\_info (to run the function in the script)

**TO EXECUTE TASKS ON THE REMOTE MACHINE (SSH)**

1. vim fabfile.py (the script up and these can be combined)

#!/user/bin/python3

from fabric.api import \*

def remote\_exec():

print(“Get system Info”)

run(“hostname”) (run is used for remote commands)

print(“Disk Space”)

run(“df –h”)

print(“Ram size”)

run (“free –m”)

print(“System uptime”)

run (“uptime”)

sudo(“yum install mariadb-server –y) (sudo is used to upgrade your privileges)

sudo(“systemctl start mariadb-server)

sudo(“systemctl enable mariadb-server)

:wq

fab –l (to show the content of the script)

**TO SSH TO A MACHINE REMOTELY**

1. cd into the directory where your vagrant file is
2. In GitBash, run the following commands:

vagrant up web01

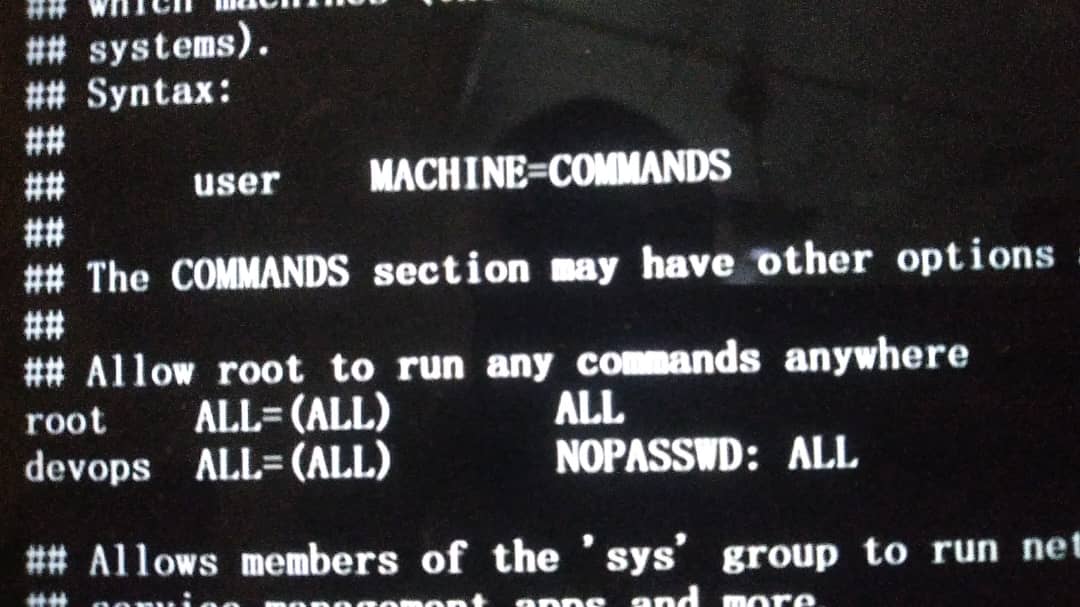
vagrant ssh web01

sudo –i

useradd devops

passwd devops

visudo (Edit the vim)



:wq

vi /etc/ssh/sshd\_config (look for password Authentication: change to yes)

:wq

systemctl restart sshd

logout

Repeat step 14 for Web02 and Web03 Vagrant machines

Bring up Scriptbox machine

1. SSH to scriptbox
2. Try to login to Web01 from scriptbox machine --- ssh [devops@192.168.10.3](mailto:devops@192.168.10.3)
3. type the given password
4. TRY TO GIVE IT ROOT USER --- SUDO –I
5. Try to login to Web02 from scriptbox machine --- ssh devops@192.168.10.4
6. type the given password
7. TRY TO GIVE IT ROOT USER --- SUDO –I

**FOR REMOTE SSH PASSWORD BASED LOGIN**

1. sudo -i
2. ssh-keygen (to generate the ssh keys)
3. /home/vagrant/.ssh/id\_rsa (file to save the key)
4. save the key generated
5. copy the generated key to Web01 --- ssh-copy-id [devops@192.168.10.3](mailto:devops@192.168.10.3)
6. enter the password
7. copy the generated key to Web02 --- ssh-copy-id [devops@192.168.10.4](mailto:devops@192.168.10.4)
8. enter the password
9. try to ssh from scriptbox to web01 --- ssh [devops@192.168.10.3](mailto:devops@192.168.10.3)
10. try to ssh from scriptbox to web02 --- ssh [devops@192.168.10.4](mailto:devops@192.168.10.4)
11. logout
12. return back to the root user of scriptbox
13. cd /opt/
14. ls
15. cd pyscripts/ (to change directory into the python scripts folder)
16. ls
17. cd fabric/ (to change directory into the folder where the fabric script created above is)
18. fab –l (to check contents of the scripts)
19. fab –H 192.168.10.3 -u devops remote\_exec ---- (These script will run on Web03 from scriptbox) (no need for –p (password) since we are using key based login, if it is password it would show in the history which is not good)

**SETUP WEB SERVICE REMOTELY USING FABRIC (PROJECT)**

STEPS:

1. Prepare your script
2. Get the link containing your Artifact (E.g. from tooplate.com)
3. Do a remote SSH connection to the web server

vim fabfile.py

#!/user/bin/python3

from fabric.api import \*

def system\_info():

print(“Disk Space”)

local(“df –h”)

print(“Ram size”)

local(“free –m”)

print(“System uptime”)

local(“uptime”)

def remote\_exec():

print(“Get system Info”)

run(“hostname”) (run is used for remote commands)

print(“Disk Space”)

run(“df –h”)

print(“Ram size”)

run (“free –m”)

print(“System uptime”)

run (“uptime”)

sudo(“yum install mariadb-server –y) (sudo is used to upgrade your privileges)

sudo(“systemctl start mariadb-server)

sudo(“systemctl enable mariadb-server)

def web\_setup(WEBURL, DIRNAME):

print(“###############################################################”)

local(“apt install zip unzip –y”)

print(“###############################################################”)

print(“Installing Dependencies”)

print(“###############################################################”)

sudo(“yum install httpd wget unzip –y”)

print(“###############################################################”)

print(“Start & Enable service”)

print(“###############################################################”)

sudo(“systemctl start httpd”)

sudo(“systemctl enable httpd”)

print(“###############################################################”)

print(“Downloading and pushing website to webservers”)

print(“###############################################################”)

local((“wget -0 website.zip %s”) % WEBURL)

local(“unzip –o website.zip”)

with lcd(DIRNAME): (LCD means local change directory)

local(“zip –r tooplate.zip \* “)

put(“tooplate.zip”, “/var/www/html/”, use\_sudo=True)

with cd(“/var/www/html/”): (these command runs in the remote machine)

sudo(“unzip -o tooplate.zip”)

sudo(“systemctl restart httpd”)

print(“Website setup is done.”)

:wq

**REMOTE SSH CONNECTION TO THE WEB SERVER**

* Get the URL of the artifact
* fab –H 192.168.10.3 -u devops web\_setup:(URL of the artifact, Directory name where it is saved)
* Test it using the IP of the web server
* To set it up on more than one server, use:

fab –H 192.168.10.3, 192.168.10.4 -u devops web\_setup:(URL of the artifact, Directory name where it is saved)

**VIRTUAL ENVIRONMENTS**

Save all your Libraries in your created Virtual Environment

* pip install virtualenv
* pip install virtualenv^C (To activate the installed Virtual Environment)
* virtualenv automation-env (A Directory will be created)
* ls
* cd automation-env/
* ls
* source bin/activate
* pip install jenkinsapi (This will be installed in the virtual environment and not system)
* You can have different versions of the libraries, different environments in the same Machine having different libraries having different versions, without interfering with python system libraries.
* deactivate (To deactivate the Virtual Environment )