<u>Using Ansible to manage Passwords , Users and Groups</u>

<u>Enforcing Complex Passwords :</u>

Installing libpam-pwquality :

Navigate to the ansible/chapter2/ directory (present in devops_for_the_desperate cloned from github) and open the pam_pwquality.yml file in your favorite editor. This file contains two tasks: Install libpam-pwquality and Configure pam_pwquality. Let's focus on the first task that uses the Ansible package module to install libpam-pwquality on the VM. At the top of the file, the install task should look like this:

```
---
- name: Install libpam-pwquality
  package:
    name: "libpam-pwquality"
    state: present
--snip--
```

Configuring pam_pwquality to Enforce a Stricter Password Policy :

With the pam_pwquality task file still open, let's review the second task from the top. It should look like this:

```
--snip--
- name: Configure pam_pwquality
  lineinfile:
    path: "/etc/pam.d/common-password"
    regexp: "pam_pwquality.so"

    line: "password required pam_pwquality.so minlen=12 lcredit=-1 ucredit=-1
           dcredit=-1 ocredit=-1 retry=3 enforce_for_root"
    state: present
--snip--
```

<u>Linux User Types :</u>

Getting Started with the Ansible User Module:

Open the user_and_group.yml file located in the ansible/chapter2/ directory. This file contains the following five tasks:

1. Ensure group developers exists.

2. Create the user bender.

3. Assign bender to the developers group.

4. Create a directory named engineering.

5. Create a file in the engineering directory.

These tasks will create a group and a user, assign a user to a group, and create a shared directory and file. Though it’s counterintuitive, let’s start by focusing on the second task on the list, which creates the user bender. It should look like this:

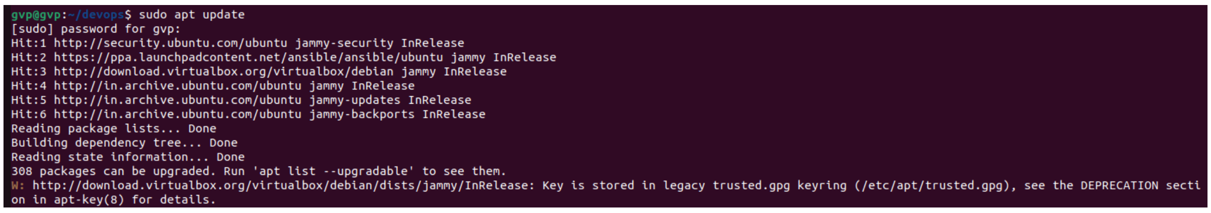
```
--snip--
- name: Create the user 'bender'
  user:
    name: bender
    shell: /bin/bash
    password: $6$...(truncated)
--snip--
```

Generating a Complex Password :

You can use a combination of two command line applications, pwgen and mkpasswd, to create the complex password. The pwgen command can generate secure passwords, and the mkpasswd command can generate passwords using different hashing algorithms. The pwgen application is provided by the pwgen package, and the mkpasswd application is provided by a package named whois. Together, these tools can generate the hash that Ansible and Linux expect.

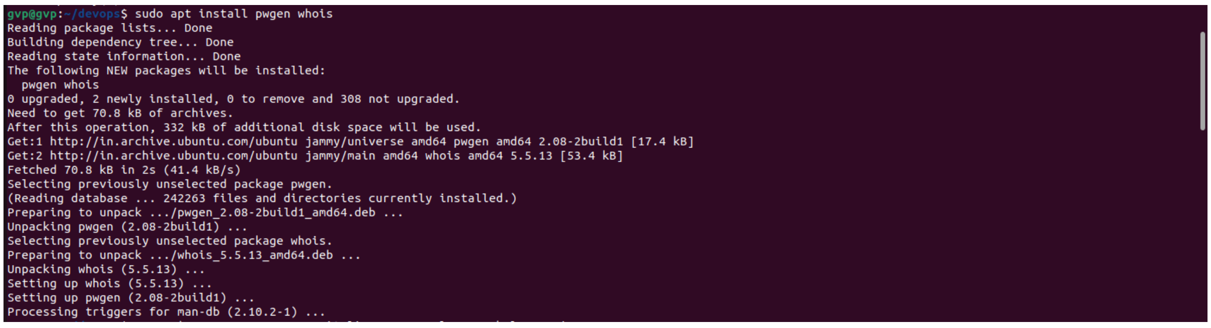
On an Ubuntu system, the password hashing algorithm is SHA-512 by default. To create your own SHA-512 hash for Ansible’s user module, use the commands below on an Ubuntu host:

$ sudo apt update



```
gvp@gvp:~/devops$ sudo apt update
[sudo] password for gvp:
Hit:1 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:2 https://ppa.launchpadcontent.net/ansible/ansible/ubuntu jammy InRelease
Hit:3 http://download.virtualbox.org/virtualbox/debian jammy InRelease
Hit:4 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Hit:5 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:6 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
308 packages can be upgraded. Run 'apt list --upgradable' to see them.
W: http://download.virtualbox.org/virtualbox/debian/dists/jammy/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION secti
on in apt-key(8) for details.
```

$ sudo apt install pwgen whois



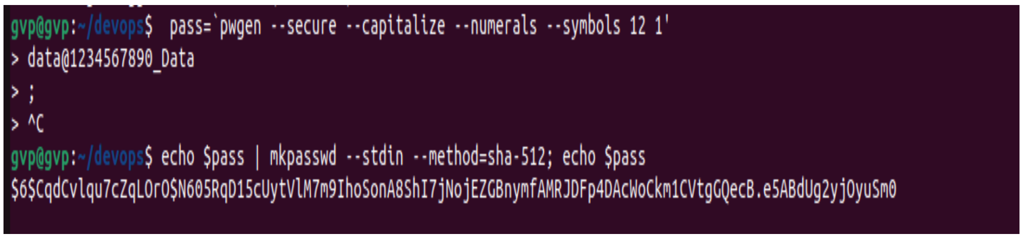
```
gvp@gvp:~/devops$ sudo apt install pwgen whois
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  pwgen whois
0 upgraded, 2 newly installed, 0 to remove and 308 not upgraded.
Need to get 70.8 kB of archives.
After this operation, 332 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 pwgen amd64 2.08-2build1 [17.4 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 whois amd64 5.5.13 [53.4 kB]
Fetched 70.8 kB in 2s (41.4 kB/s)
Selecting previously unselected package pwgen.
(Reading database ... 242263 files and directories currently installed.)
Preparing to unpack .../pwgen_2.08-2build1_amd64.deb ...
Unpacking pwgen (2.08-2build1) ...
Selecting previously unselected package whois.
Preparing to unpack .../whois_5.5.13_amd64.deb ...
Unpacking whois (5.5.13) ...
Setting up whois (5.5.13) ...
Setting up pwgen (2.08-2build1) ...
Processing triggers for man-db (2.10.2-1) ...
```

$ pass=`pwgen --secure --capitalize --numerals --symbols 12 1`

Use this command to create a password which should contain atleast 1 uppercase , lowercase, digit, special symbols. In my case it is : data@1234567890_Data

$ echo $pass | mkpasswd --stdin --method=sha-512; echo $pass

Use this command to convert the password in to hash using SHA-512



```
gvp@gvp:~/devops$  pass=`pwgen --secure --capitalize --numerals --symbols 12 1'
> data@1234567890_Data
> ;
> ^C
gvp@gvp:~/devops$ echo $pass | mkpasswd --stdin --method=sha-512; echo $pass
$6$CqdCvlqu7cZqLOrO$N605RqD15cUytVlM7m9IhoSonA8ShI7jNojEZGBnymfAMRJDFp4DAcWoCkm1CVtgGQecB.e5ABdUg2yjOyuSm0
```

<u>Linux Groups :</u>

Getting Started with the Ansible Group Module :

Open the user_and_group.yml file in your editor to review the group creation task. The first task in the file should look like this:

```
- name: Ensure group 'developers' exists
  group:
    name: developers
    state: present
--snip--
```

Assigning a User to the Group :

To add a user to a group with Ansible, you'll leverage the user module once again. In the user_and_group.yml file, locate the task that assigns bender to the developers group (the third task from the top in the file). It should look like this:

```
--snip--
- name: Assign 'bender' to the 'developers' group
  user:
    name: bender
    groups: developers
    append: yes
--snip--
```

Creating Protected Resources :

The last two tasks in the user_and_group.yml file, which deal with creating a directory (/opt/engineering/) and a file (/opt/engineering/private.txt) on the VM. You'll use this directory and file to test user access for bender later. With the user_and_group.yml file still open, locate the two tasks. Start with the directory creation task (the fourth from the top in the file), which should look like this:

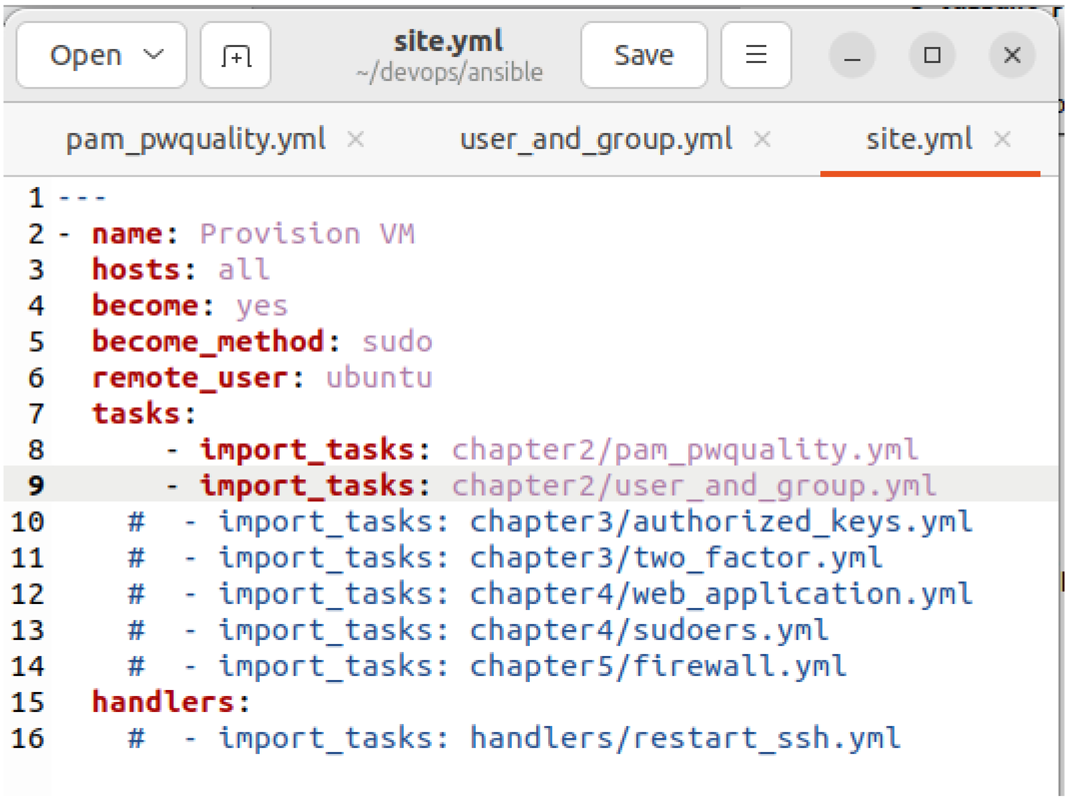
```
- name: Create a directory named 'engineering'
  file:
    path: /opt/engineering
    state: directory
    mode: 0750
    group: developers
```

The last task in the user_and_group.yml file creates an empty file inside the /opt/engineering/ directory you just created. The task, located at the bottom of the file, should look like this:

```
- name: Create a file in the engineering directory
  file:
    path: "/opt/engineering/private.txt"
    state: touch
    mode: 0770
    group: developers
```

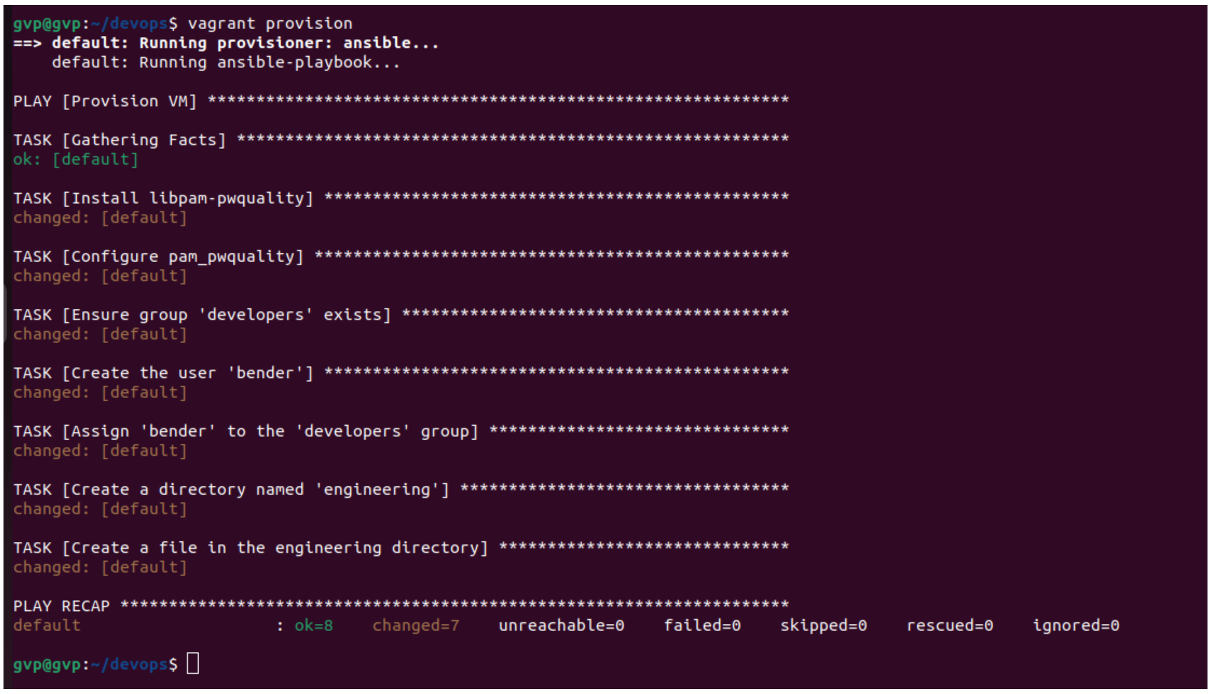
Updating the VM:

To provision the VM, you'll need to uncomment the tasks in the playbook under the ansible/ directory. The site.yml file is the playbook you referenced in the provisioners section of your Vagrantfile. Open the site.yml playbook file in your editor and locate the Chapter 2 tasks and uncomment the chapter2 tasks. It will look like this:



```
---
- name: Provision VM
  hosts: all
  become: yes
  become_method: sudo
  remote_user: ubuntu
  tasks:
      - import_tasks: chapter2/pam_pwquality.yml
      - import_tasks: chapter2/user_and_group.yml
   #  - import_tasks: chapter3/authorized_keys.yml
   #  - import_tasks: chapter3/two_factor.yml
   #  - import_tasks: chapter4/web_application.yml
   #  - import_tasks: chapter4/sudoers.yml
   #  - import_tasks: chapter5/firewall.yml
  handlers:
   #  - import_tasks: handlers/restart_ssh.yml
```

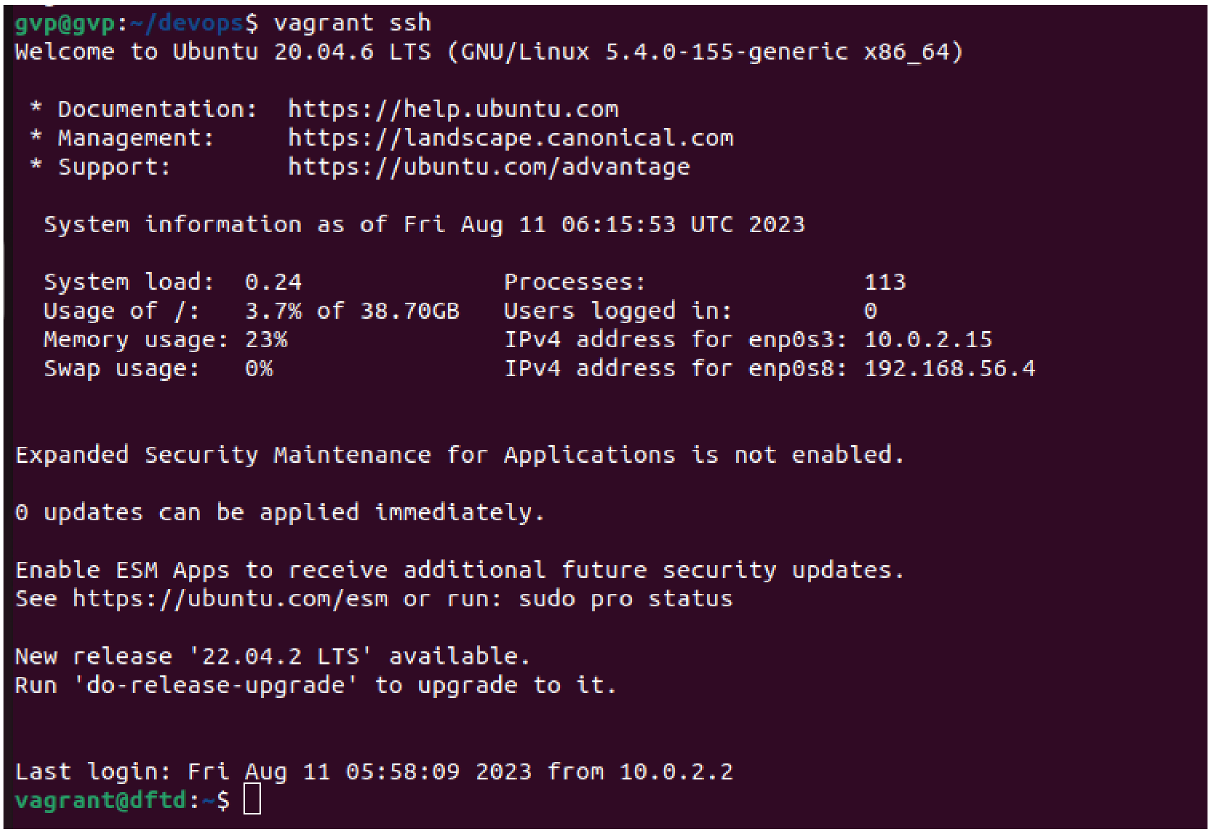
Both Chapter 2 tasks, pam_pwquality and user_and_group, are now uncommented, so they will execute the next time you provision the VM. Save and close the playbook file for now. . With the VM running, all you need to do is issue the vagrant provision command :



```
gvp@gvp:~/devops$ vagrant provision
==> default: Running provisioner: ansible...
    default: Running ansible-playbook...

PLAY [Provision VM] ************************************************************

TASK [Gathering Facts] *********************************************************
ok: [default]

TASK [Install libpam-pwquality] ************************************************
changed: [default]

TASK [Configure pam_pwquality] *************************************************
changed: [default]

TASK [Ensure group 'developers' exists] ****************************************
changed: [default]

TASK [Create the user 'bender'] ************************************************
changed: [default]

TASK [Assign 'bender' to the 'developers' group] *******************************
changed: [default]

TASK [Create a directory named 'engineering'] **********************************
changed: [default]

TASK [Create a file in the engineering directory] ******************************
changed: [default]

PLAY RECAP *********************************************************************
default                    : ok=8    changed=7    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

gvp@gvp:~/devops$ 
```

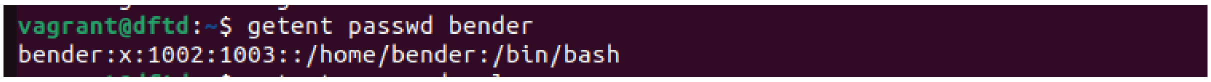
Testing User and Group Permissions :

To test the user and group permissions you just configured, you'll issue the ssh command for vagrant to access the VM. Make sure you are in the devops/ directory so you have access to the Vagrantfile. Once there, enter the command below in your terminal to log in to the VM:



```
gvp@gvp:~/devops$ vagrant ssh
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-155-generic x86_64)

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

  System information as of Fri Aug 11 06:15:53 UTC 2023

  System load:  0.24              Processes:               113
  Usage of /:   3.7% of 38.70GB   Users logged in:         0
  Memory usage: 23%               IPv4 address for enp0s3: 10.0.2.15
  Swap usage:   0%                IPv4 address for enp0s8: 192.168.56.4


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '22.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Fri Aug 11 05:58:09 2023 from 10.0.2.2
vagrant@dftd:~$ 
```

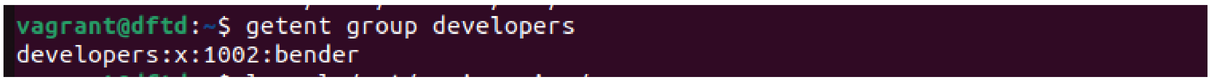
Next, to verify the user bender was created, you'll use the getent command to query the passwd database for the user. To check bender's existence, enter the following command:



```
vagrant@dftd:~$ getent passwd bender
bender:x:1002:1003::/home/bender:/bin/bash
```

Your result should look similar to the output above. If the user was not created, the command will complete without any result.

Now, you should check whether the developers group exists and whether bender is a member of it. Query the group database for this information:

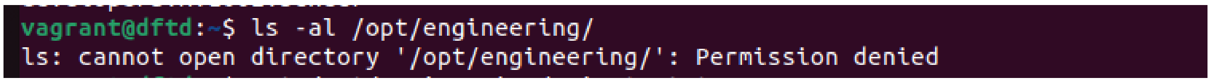


```
vagrant@dftd:~$ getent group developers
developers:x:1002:bender
```

The result should look like the output above, with a developers group and the user bender assigned to it. If the group did not exist, the command would have exited without any result.

For the final check, test that only members of the developers group can access the /opt/engineering/ directory and the private.txt file.

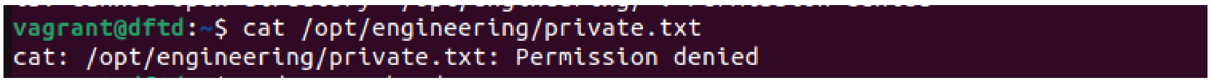
To do this, try to access the directory and file once as the vagrant user and then again as the bender user. While logged in as vagrant, enter the command below to list the /opt/ engineering/ directory and its contents:



```
vagrant@dftd:~$ ls -al /opt/engineering/
ls: cannot open directory '/opt/engineering/': Permission denied
```

The output indicates that access is denied when trying to list files in /opt/engineering as the vagrant user. This is because the vagrant user is not a member of the developers group and thus does not have read access to the directory.

Now, to test the file permissions for vagrant, use the cat command to view the /opt/engineering/private.txt file:

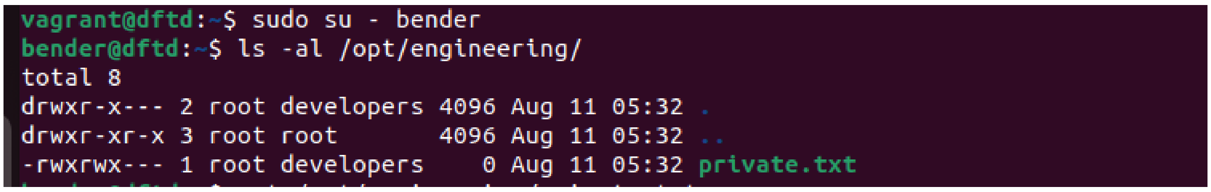


```
vagrant@dftd:~$ cat /opt/engineering/private.txt
cat: /opt/engineering/private.txt: Permission denied
```

The same error occurs because the vagrant user does not have read permissions on the file.

The next test is to verify that bender has access to this same directory and file. To do this, you must be logged in as the bender user.

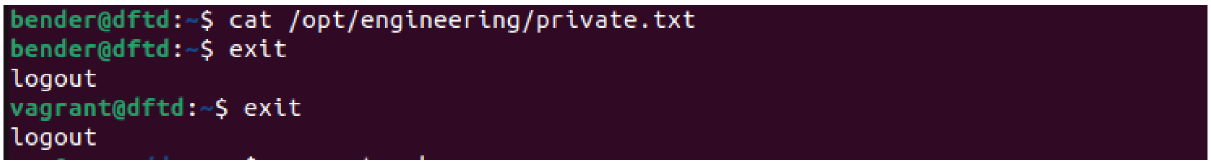
Switch users from vagrant to bender using the sudo su command. Once you have successfully switched users, try the command to list the directory again:



```
vagrant@dftd:~$ sudo su - bender
bender@dftd:~$ ls -al /opt/engineering/
total 8
drwxr-x--- 2 root developers 4096 Aug 11 05:32 .
drwxr-xr-x 3 root root       4096 Aug 11 05:32 ..
-rwxrwx--- 1 root developers    0 Aug 11 05:32 private.txt
```

Now, as you can see, you have successfully accessed the directory and its contents as bender, and the private.txt file is viewable.

Next, enter the following command to check whether bender can read the contents of the /opt/engineering/private.txt file:



```
bender@dftd:~$ cat /opt/engineering/private.txt
bender@dftd:~$ exit
logout
vagrant@dftd:~$ exit
logout
```

You use the cat command again to view the contents of the file. Since the file is empty, there is no output. Now exit from the user using exit command.

Using Ansible to Configure SSH

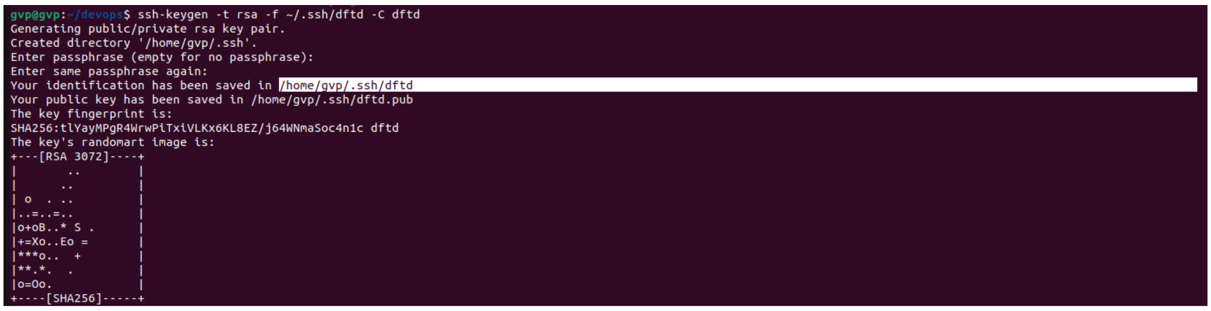
Generating a Public Key Pair :

To generate a key pair, you'll use the ssh-keygen command line tool. This tool, usually installed on Unix hosts by default as part of the ssh package, generates and manages authentication key pairs for SSH.

In a terminal on your local host, enter the following command to generate a new key pair:

While running this command it will ask to enter a passphrase. Enter any passphrase and keep it in mind for further use.

It will generate hash value for your passphrase and key's randomart image.



```
gvp@gvp:~/devops$ ssh-keygen -t rsa -f ~/.ssh/dftd -C dftd
Generating public/private rsa key pair.
Created directory '/home/gvp/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/gvp/.ssh/dftd
Your public key has been saved in /home/gvp/.ssh/dftd.pub
The key fingerprint is:
SHA256:tlYayMPgR4WrwPiTxiVLKx6KL8EZ/j64WNmaSoc4n1c dftd
The key's randomart image is:
+---[RSA 3072]----+
|       ..        |
|      ..         |
| o  . ..         |
|..=..=..         |
|o+oB..* S .      |
|+=Xo..Eo =       |
|***o..  +        |
|**.*.   .        |
|o=Oo.            |
+----[SHA256]-----+
```

Using Ansible to Get Your Public Key on the VM :

This task and all the other tasks related to this chapter are located in the cloned repository under the ansible/chapter3/ directory.

Open the authorized_keys.yml file in your favorite editor to review the Ansible task.

The first thing you should notice is that this file has only one task. It should look like this:

```
- name: Set authorized key file from local user
  authorized_key:
    user: bender
    state: present
    key: "{{ lookup('file', lookup('env','HOME') + '/.ssh/dftd.pub') }}"
```

Adding Two-Factor Authentication:

To enforce 2FA on your VM, you'll use some provided Ansible tasks to install another PAM module, configure the SSH server, and enable 2FA. To review the provided tasks, first open the two_factor.yml file in your editor. This file has seven tasks, and each task has a specific job to enable 2FA.

The tasks are named as follows:

1. Install the libpam-google-authenticator package.

2. Copy over preconfigured GoogleAuthenticator config.

3. Disable password authentication for SSH.

4. Configure PAM to use GoogleAuthenticator for SSH logins.

5. Set ChallengeResponseAuthentication to Yes.

6. Set Authentication Methods for bender, vagrant, and ubuntu.

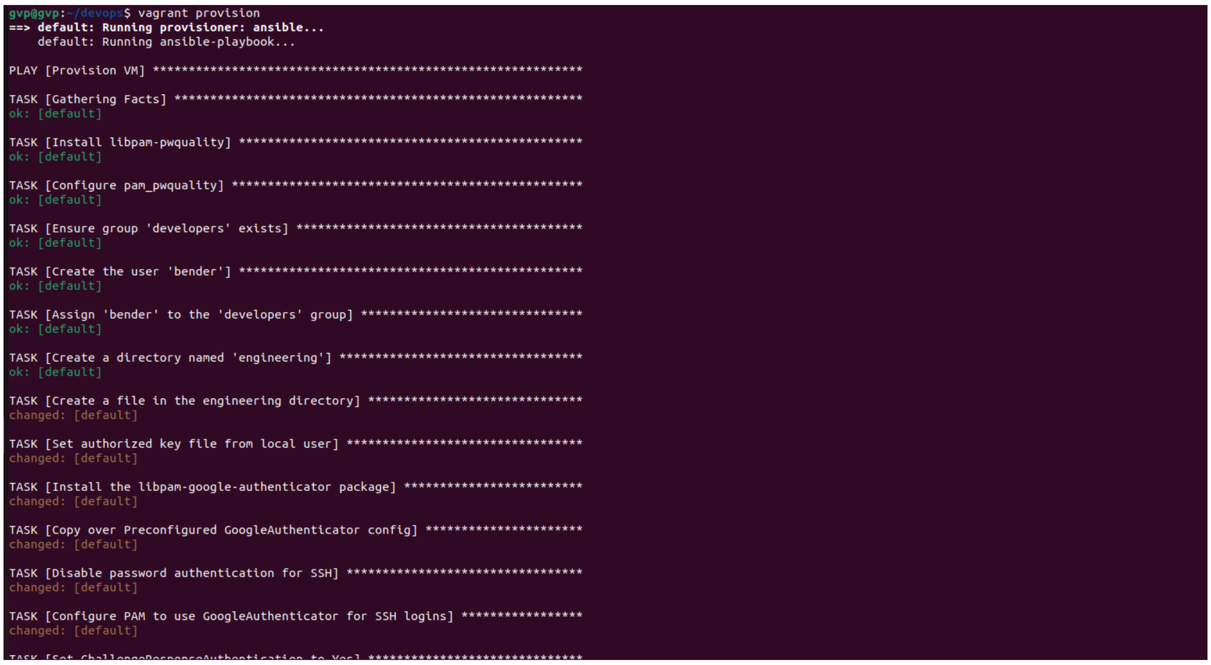
7. Insert an additional line here that reads: Restart SSH Server.

Provisioning the VM:

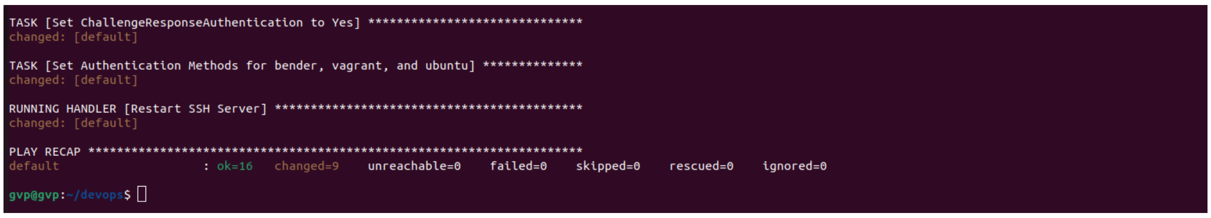
To provision the VM with all the tasks described thus far, you'll need to uncomment them in the playbook. Follow essentially the same process that you followed in Chapter 2, but this time , you'll need to uncommment two tasks and a handler. Open the site.yml file in your editor and locate the tasks and uncomment them, which should look like this:

Now, you'll automate the configuration of the VM using Vagrant.

Navigate to the devops/ directory, and once there, enter the following command:



```
gvp@gvp:~/devops$ vagrant provision
==> default: Running provisioner: ansible...
    default: Running ansible-playbook...

PLAY [Provision VM] ************************************************************

TASK [Gathering Facts] *********************************************************
ok: [default]

TASK [Install libpam-pwquality] ************************************************
ok: [default]

TASK [Configure pam_pwquality] *************************************************
ok: [default]

TASK [Ensure group 'developers' exists] ****************************************
ok: [default]

TASK [Create the user 'bender'] ************************************************
ok: [default]

TASK [Assign 'bender' to the 'developers' group] *******************************
ok: [default]

TASK [Create a directory named 'engineering'] **********************************
ok: [default]

TASK [Create a file in the engineering directory] ******************************
changed: [default]

TASK [Set authorized key file from local user] *********************************
changed: [default]

TASK [Install the libpam-google-authenticator package] *************************
changed: [default]

TASK [Copy over Preconfigured GoogleAuthenticator config] **********************
changed: [default]

TASK [Disable password authentication for SSH] *********************************
changed: [default]

TASK [Configure PAM to use GoogleAuthenticator for SSH logins] *****************
changed: [default]
```



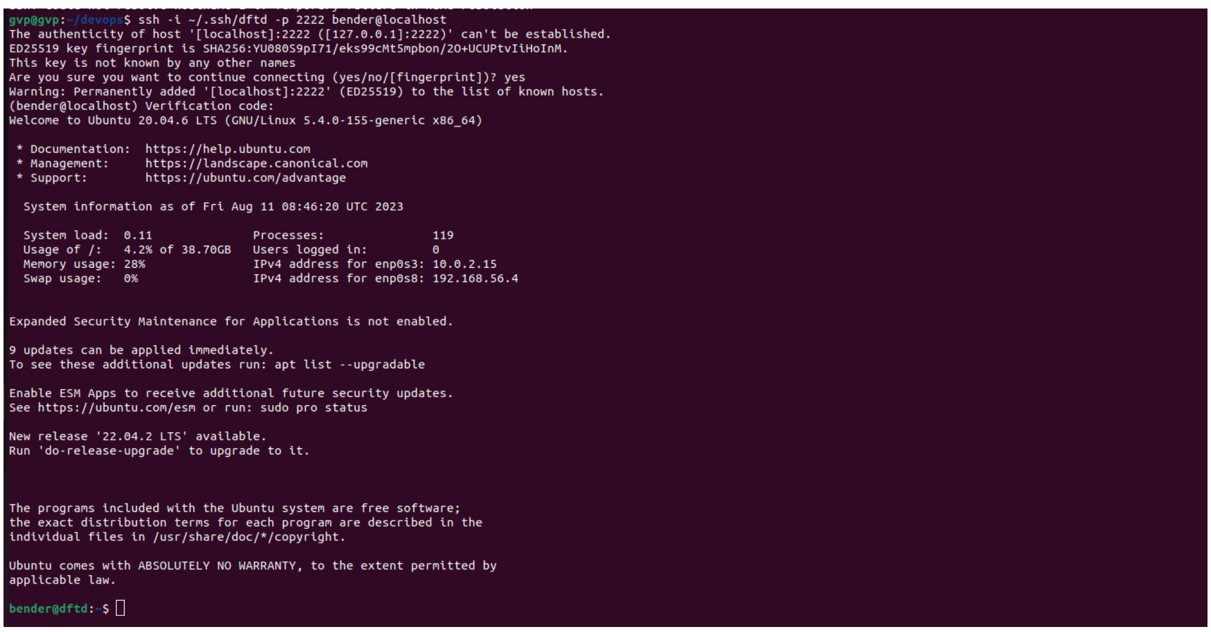
```
TASK [Set ChallengeResponseAuthentication to Yes] ******************************
changed: [default]

TASK [Set Authentication Methods for bender, vagrant, and ubuntu] **************
changed: [default]

RUNNING HANDLER [Restart SSH Server] *******************************************
changed: [default]

PLAY RECAP *********************************************************************
default                    : ok=16   changed=9    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

gvp@gvp:~/devops$ 
```

Testing SSH Access:

With the VM successfully provisioned, you should test bender’s access over SSH. To test public key and 2FA over SSH, you’ll need the private key you created earlier and one of the emergency tokens from the google_authenticator file in the repository.

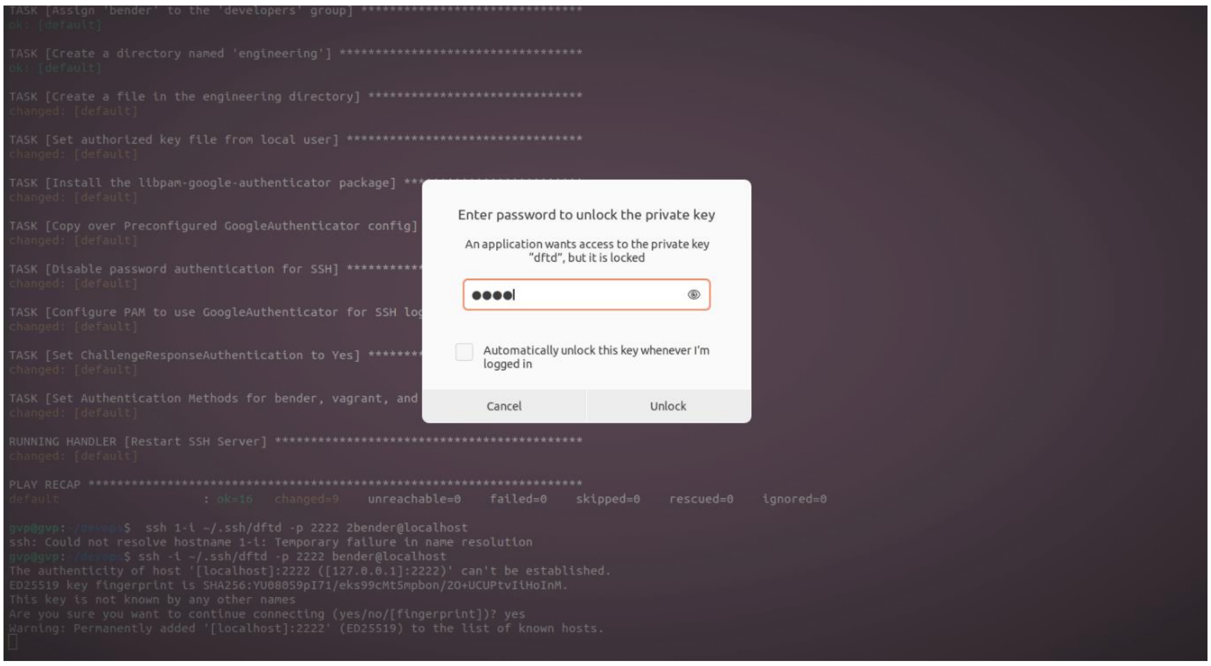
The emergency tokens are the 10 eight-digit numbers located at the bottom of the ansible/ chapter3/google_authenticator file. Choose the first one.

To ssh in to the VM as bender, open a terminal on your local host and enter the following command:



```
gvp@gvp:~/devops$ ssh -i ~/.ssh/dftd -p 2222 bender@localhost
The authenticity of host '[localhost]:2222 ([127.0.0.1]:2222)' can't be established.
ED25519 key fingerprint is SHA256:YU080S9pI71/eks99cMt5mpbon/2O+UCUPtvIiHoInM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[localhost]:2222' (ED25519) to the list of known hosts.
(bender@localhost) Verification code:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-155-generic x86_64)

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

  System information as of Fri Aug 11 08:46:20 UTC 2023

  System load:  0.11               Processes:               119
  Usage of /:   4.2% of 38.70GB    Users logged in:         0
  Memory usage: 28%                IPv4 address for enp0s3: 10.0.2.15
  Swap usage:   0%                 IPv4 address for enp0s8: 192.168.56.4


Expanded Security Maintenance for Applications is not enabled.

9 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '22.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

bender@dftd:~$ 
```

While executing this command it will ask to enter a passphrase that you have entered before while generating a public key pair.



```
TASK [Assign 'bender' to the 'developers' group] ********************************
ok: [default]

TASK [Create a directory named 'engineering'] **********************************
ok: [default]

TASK [Create a file in the engineering directory] ******************************
changed: [default]

TASK [Set authorized key file from local user] *********************************
changed: [default]

TASK [Install the libpam-google-authenticator package] ***
changed: [default]

TASK [Copy over Preconfigured GoogleAuthenticator config]
changed: [default]

TASK [Disable password authentication for SSH] **********
changed: [default]

TASK [Configure PAM to use GoogleAuthenticator for SSH log
changed: [default]

TASK [Set ChallengeResponseAuthentication to Yes] *******
changed: [default]

TASK [Set Authentication Methods for bender, vagrant, and
changed: [default]

RUNNING HANDLER [Restart SSH Server] *******************************************
changed: [default]

PLAY RECAP *********************************************************************
default                    : ok=16   changed=9    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

gvp@gvp:~/devops$  ssh 1-i ~/.ssh/dftd -p 2222 2bender@localhost
ssh: Could not resolve hostname 1-i: Temporary failure in name resolution
gvp@gvp:~/devops$ ssh -i ~/.ssh/dftd -p 2222 bender@localhost
The authenticity of host '[localhost]:2222 ([127.0.0.1]:2222)' can't be established.
ED25519 key fingerprint is SHA256:YU080S9pI71/eks99cMt5mpbon/2O+UCUPtvIiHoInM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[localhost]:2222' (ED25519) to the list of known hosts.
```

Enter password to unlock the private key

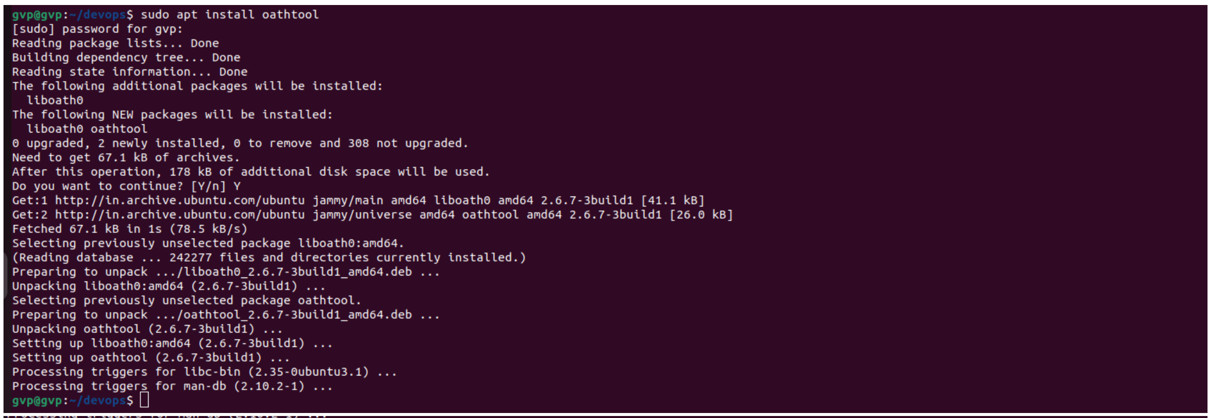
An application wants access to the private key “dftd”, but it is locked

●●●●

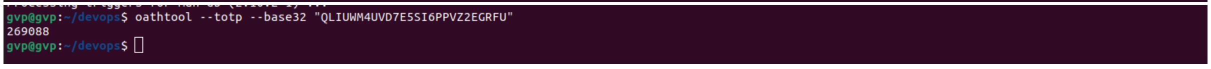
Automatically unlock this key whenever I’m logged in

Each of the 10 tokens provided is for one-time use. Every time you successfully use one, it’s removed from the /home/bender/.google_authenticator file. If, for some reason, you burn through all the tokens, run the vagrant provision command again to replace the file and replenish the tokens.

Another option is to use a TOTP application like oathtool and generate a time-based one-time token by using the Base32 secret at the top of the /home/bender/.google_authenticator file. You can install oathtool with Ubuntu's package manager by using the apt install oathtool command. Every time you need a token, you can use the following command:



```
gvp@gvp:~/devops$ sudo apt install oathtool
[sudo] password for gvp:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  liboath0
The following NEW packages will be installed:
  liboath0 oathtool
0 upgraded, 2 newly installed, 0 to remove and 308 not upgraded.
Need to get 67.1 kB of archives.
After this operation, 178 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 liboath0 amd64 2.6.7-3build1 [41.1 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 oathtool amd64 2.6.7-3build1 [26.0 kB]
Fetched 67.1 kB in 1s (78.5 kB/s)
Selecting previously unselected package liboath0:amd64.
(Reading database ... 242277 files and directories currently installed.)
Preparing to unpack .../liboath0_2.6.7-3build1_amd64.deb ...
Unpacking liboath0:amd64 (2.6.7-3build1) ...
Selecting previously unselected package oathtool.
Preparing to unpack .../oathtool_2.6.7-3build1_amd64.deb ...
Unpacking oathtool (2.6.7-3build1) ...
Setting up liboath0:amd64 (2.6.7-3build1) ...
Setting up oathtool (2.6.7-3build1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
Processing triggers for man-db (2.10.2-1) ...
gvp@gvp:~/devops$
```



```
gvp@gvp:~/devops$ oathtool --totp --base32 "QLIUWM4UVD7E5SI6PPVZ2EGRFU"
269088
gvp@gvp:~/devops$
```

Here, you pass oathtool your Base32 secret in the double quotes and set the flags --totp and --base32 to generate the token. In this result, the token 269088 is generated and can be used when prompted for a verification c