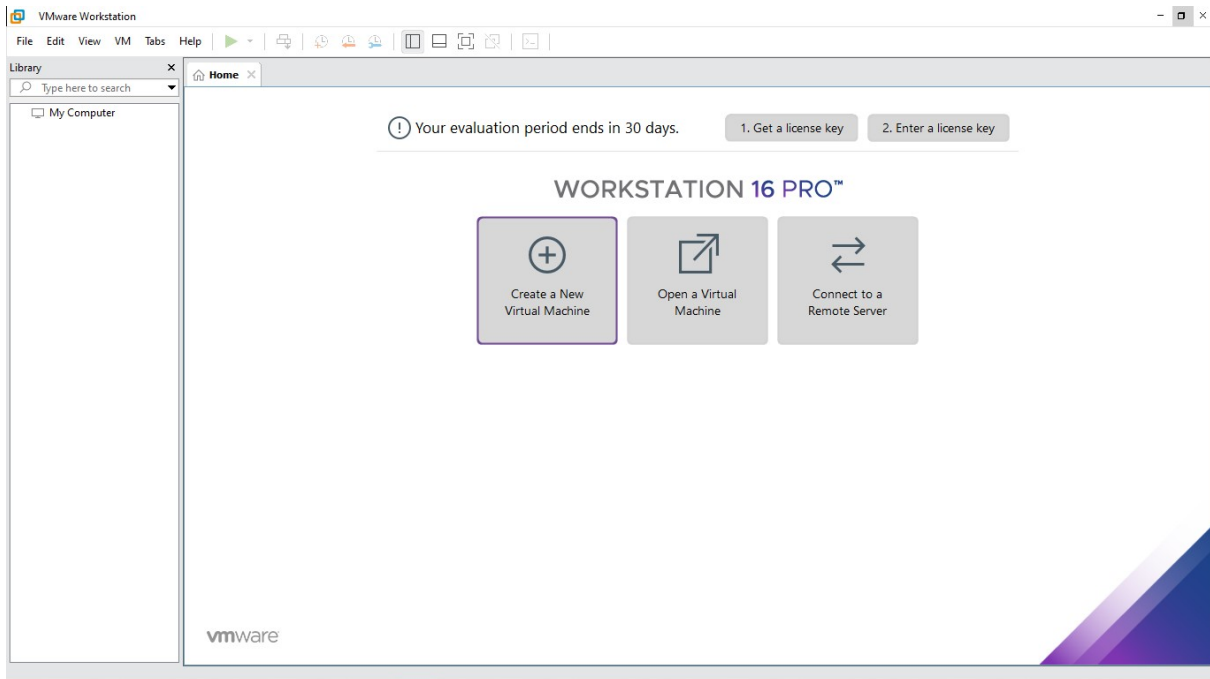


1) Installation and configuration of Virtual Machine using VMware.

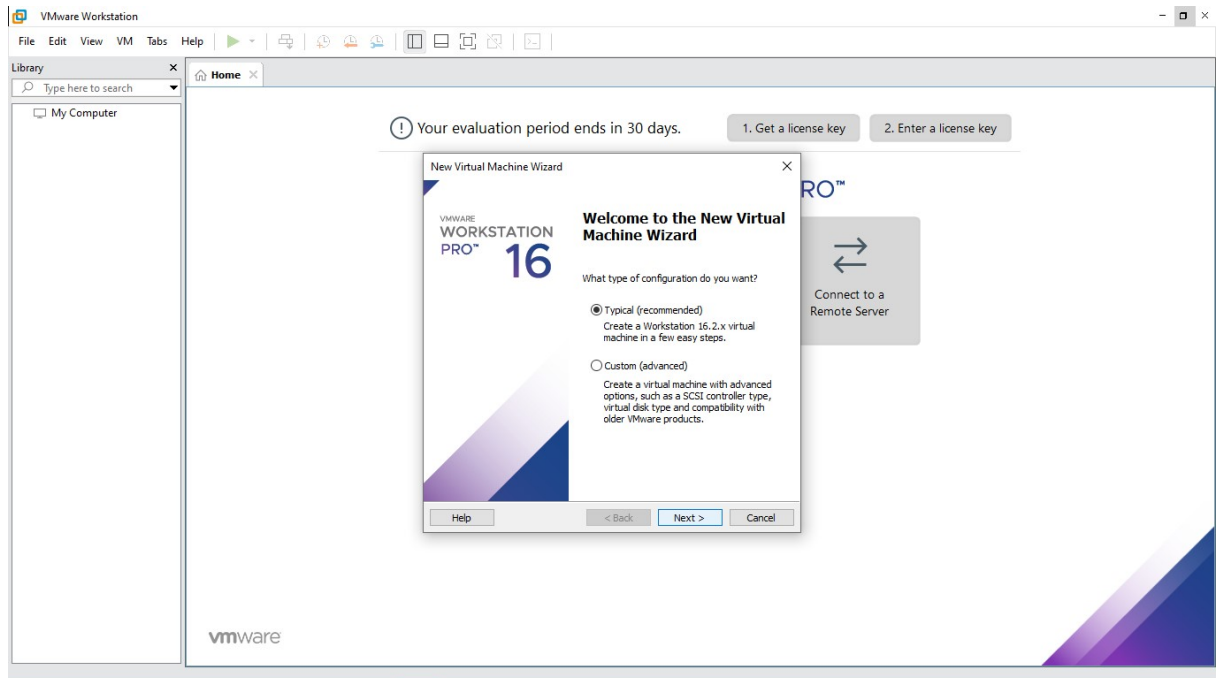
1. Launch VMware Workstation.



2. Click **New Virtual Machine**.

3. Select the type of virtual machine you want to create and click **Next**:

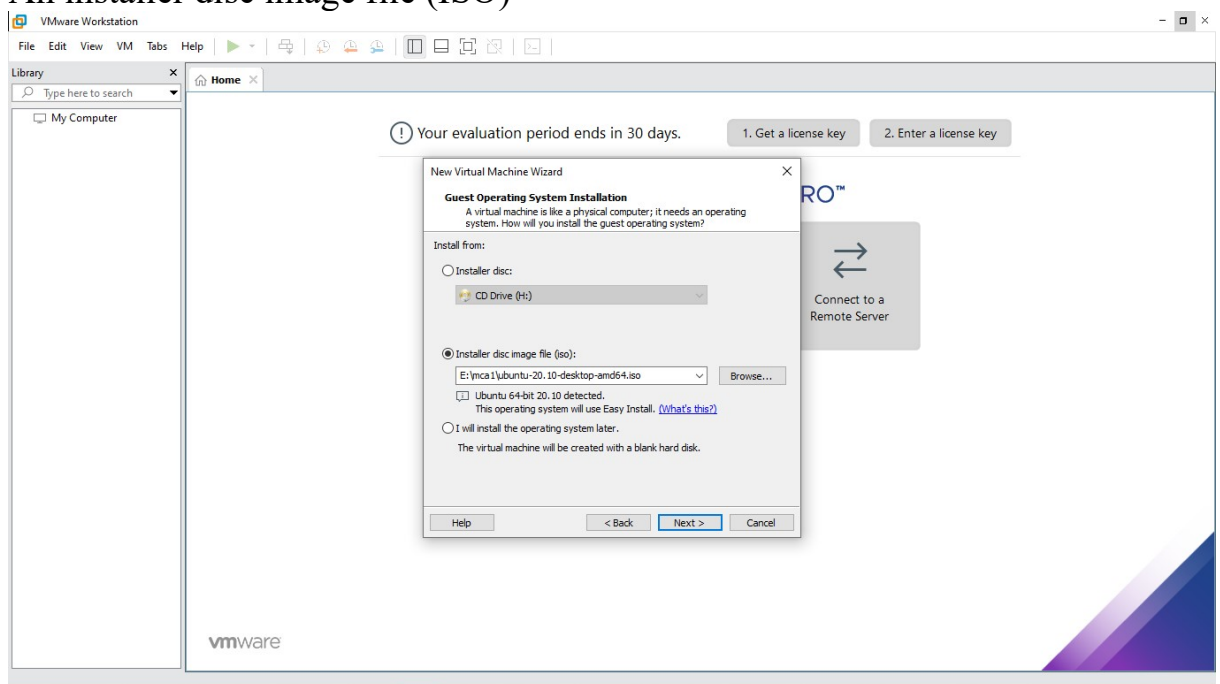
- **Custom:** This gives you an option to create a virtual machine and choose its hardware compatibility. You can choose from Workstation 16.x, Workstation 15.x, Workstation 14.x .
- **Typical:** This creates a virtual machine which has the same hardware version as the version of Workstation you are using. If you are using Workstation 16.x, it creates a virtual machine with hardware version 16. If you are using Workstation 15.x a virtual machine with hardware version 15 is created.



4. Click **Next**.

5. Select your guest operating system (OS), then click **Next**. You can install the OS using:

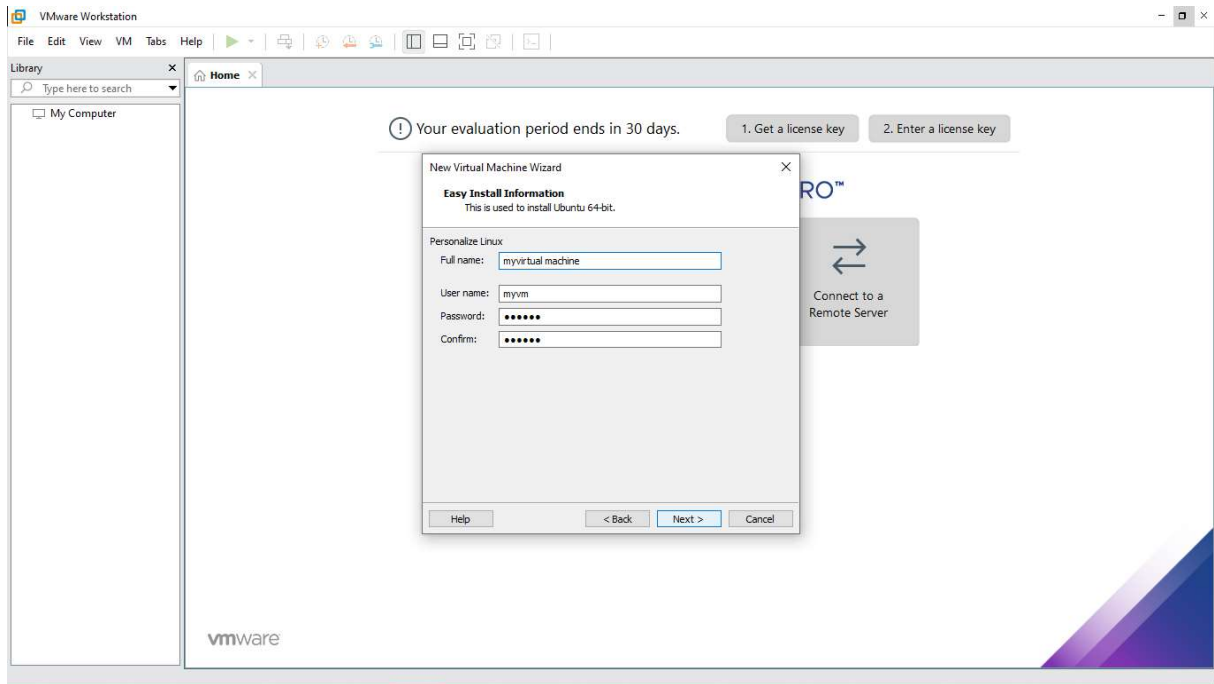
- An installer disc (CD/DVD)
- An installer disc image file (ISO)



6. Click **Next**.

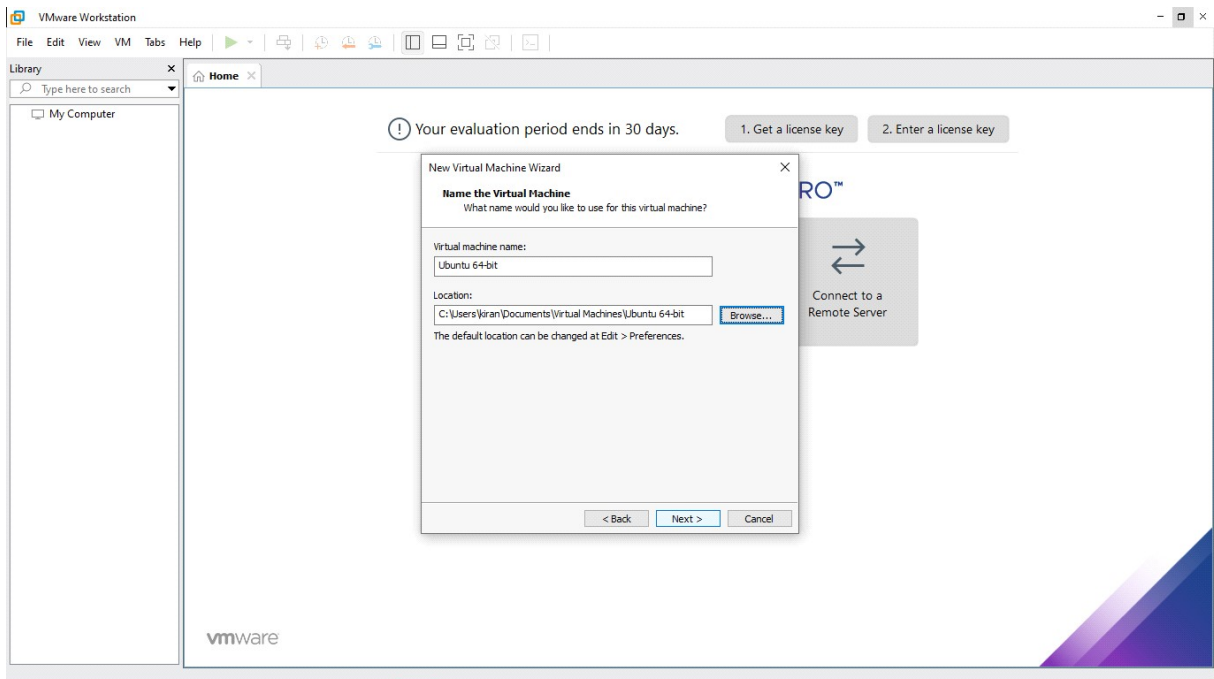
7. Enter your Product Key(for paid operating systems like windows).

8. Create a user name and password.



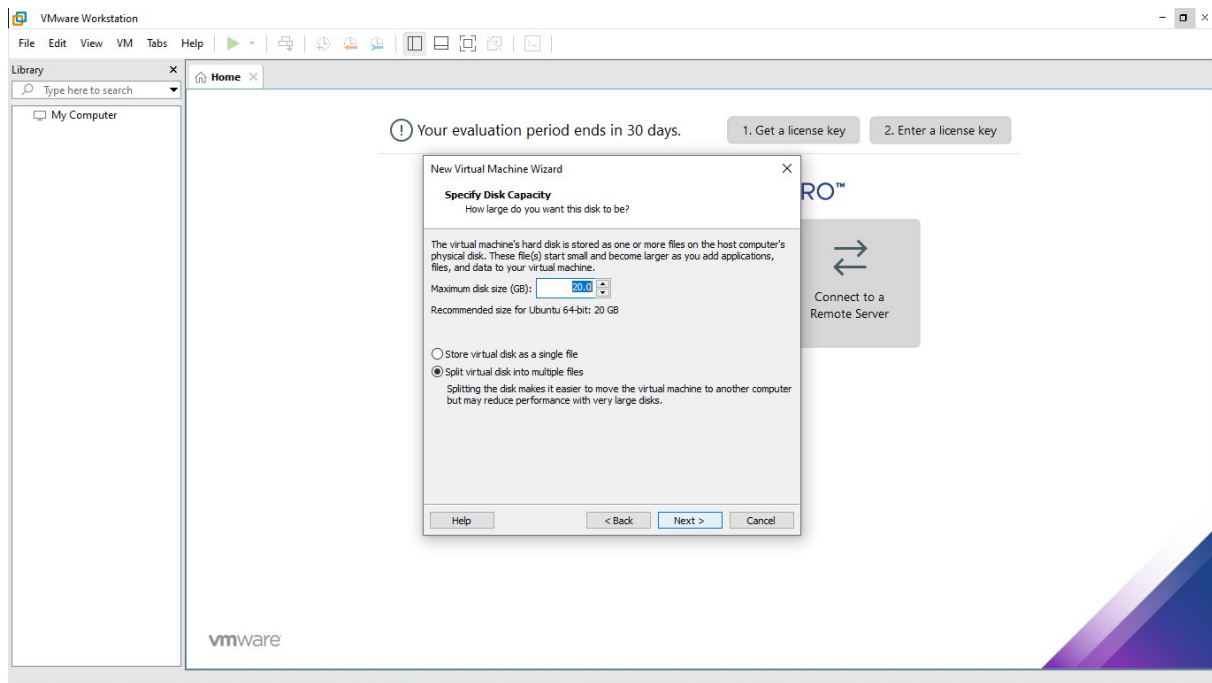
9. Click **Next**.

10. Enter a virtual machine name and specify a location for virtual machine files to be saved,



click **Next**.

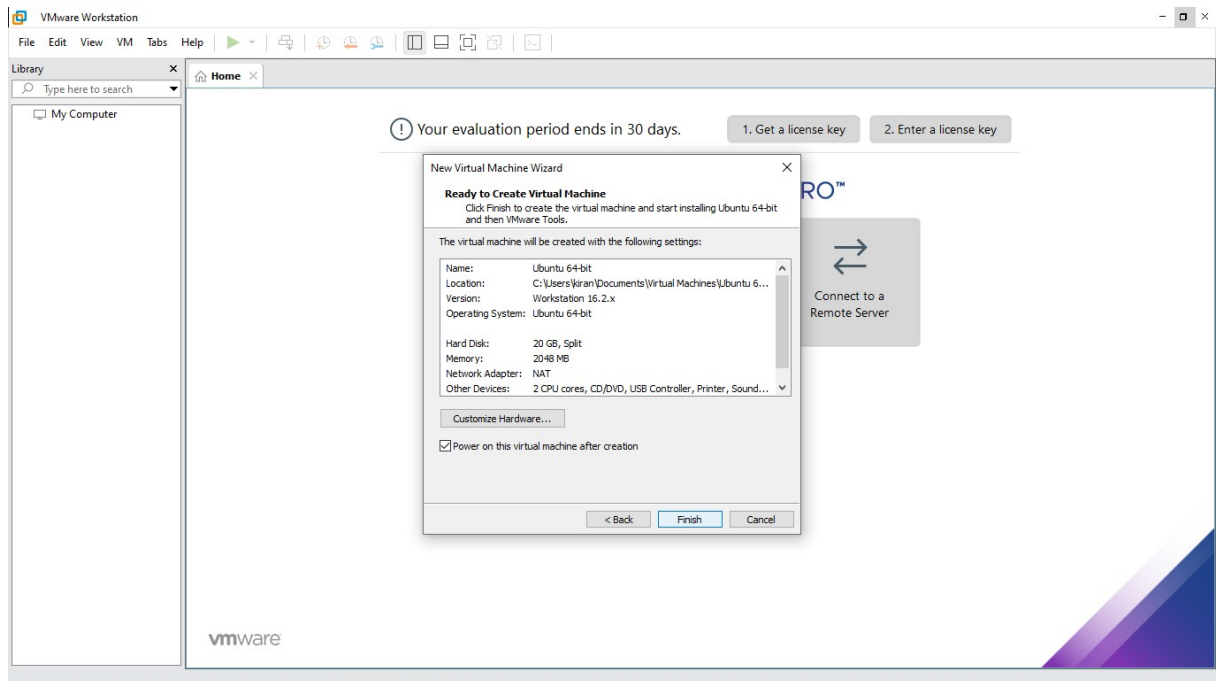
11. Establish the virtual machine's disk size, select whether to store the virtual disk as a single file or split the virtual disk into 2GB files,



click **Next**.

12. Verify the other configuration settings for your virtual machine:
 - Memory – change the amount of memory allocated to the virtual machine.
 - Processors – change the number of processors, number of cores per processor, and the virtualization engine.
 - CD / DVD – with advanced settings where you can choose between SCSI, IDE.
 - Network adapter – configure it to bridge, NAT, or Host-only mode, or customize where you can choose between 0 to 9 adapters.
 - USB Controller.
 - Sound card.

- Display – enable 3D graphics.



13. Click **Finish**.
14. When the virtual machine is powered on, the VMware Tools installation starts. You are prompted to restart your virtual machine once the Tools installation completes.

2)Installation and configuration of virtual machine using kvm

step1: installing libvirt package.

```
sudo apt-get install qemu-kvm libvirt-bin bridge-utils virt-manager
```

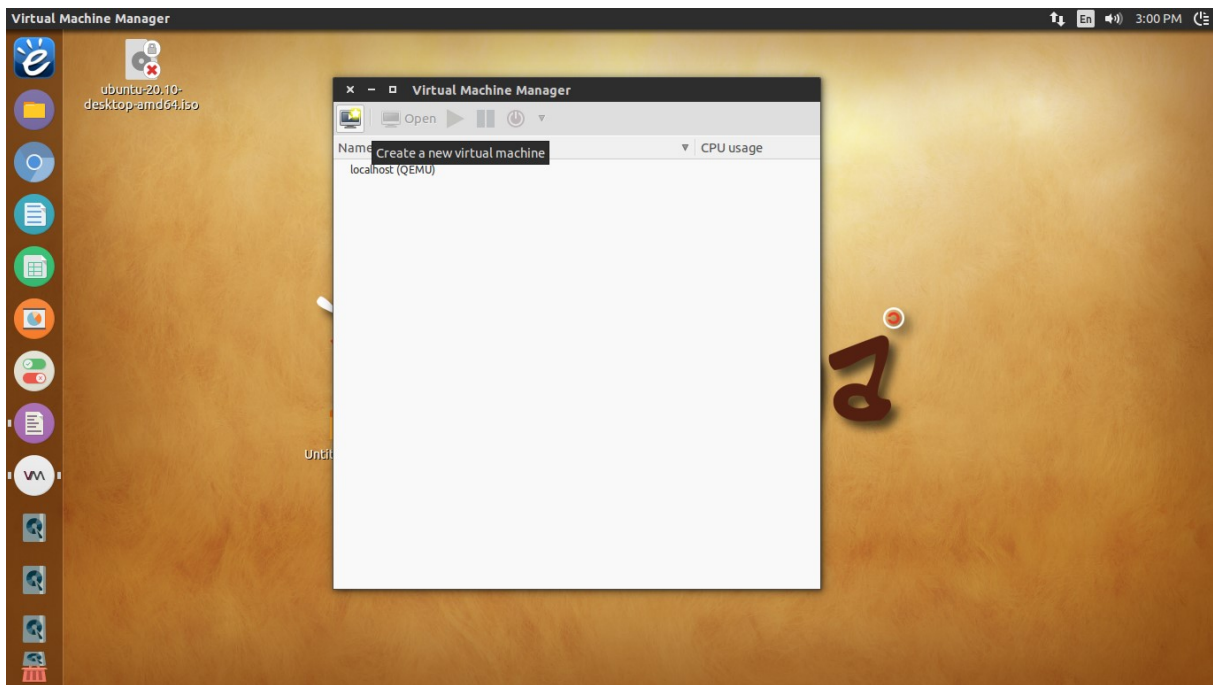
step2:#sudoadduserrait

logout and login as rait.

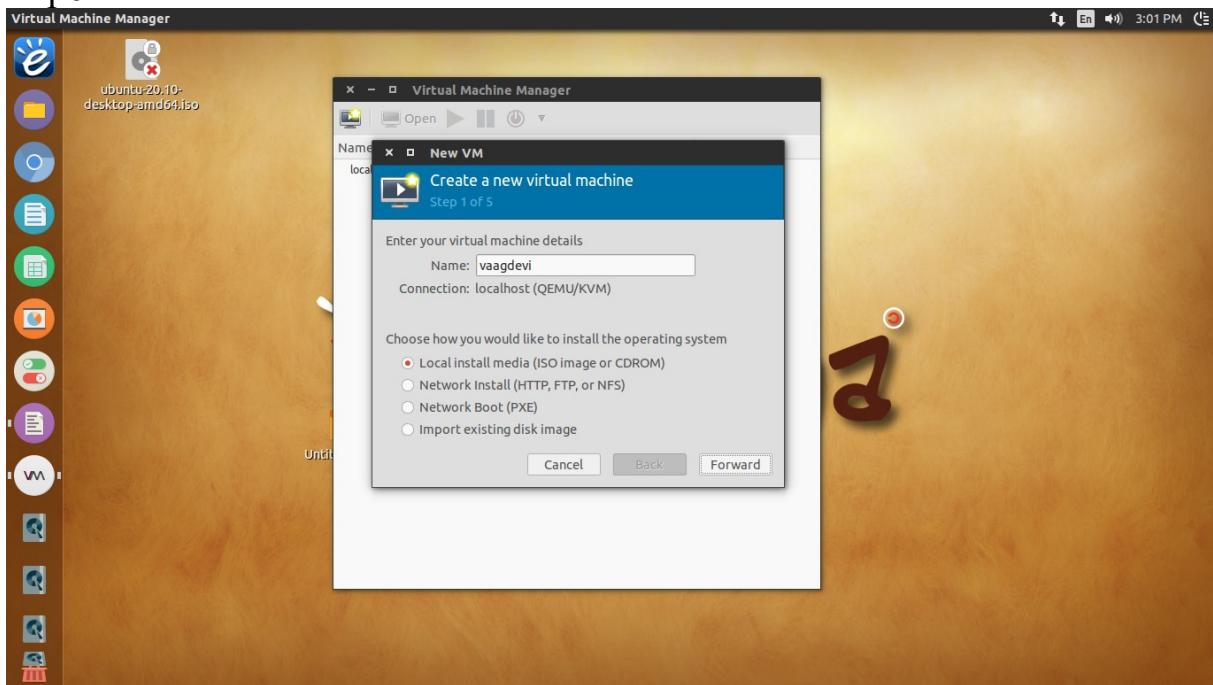
step3:#sudoadduserraitlibvirt

step4:open virtual machine manager.

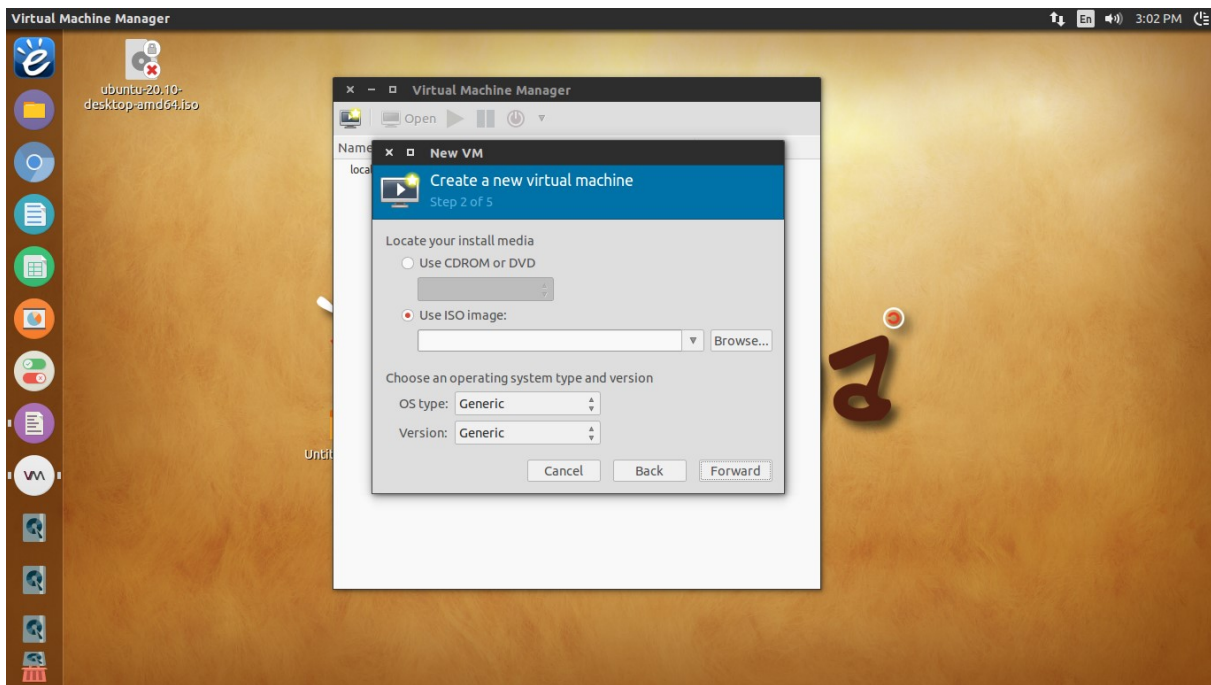
step5:Click on create a new virtual machine.



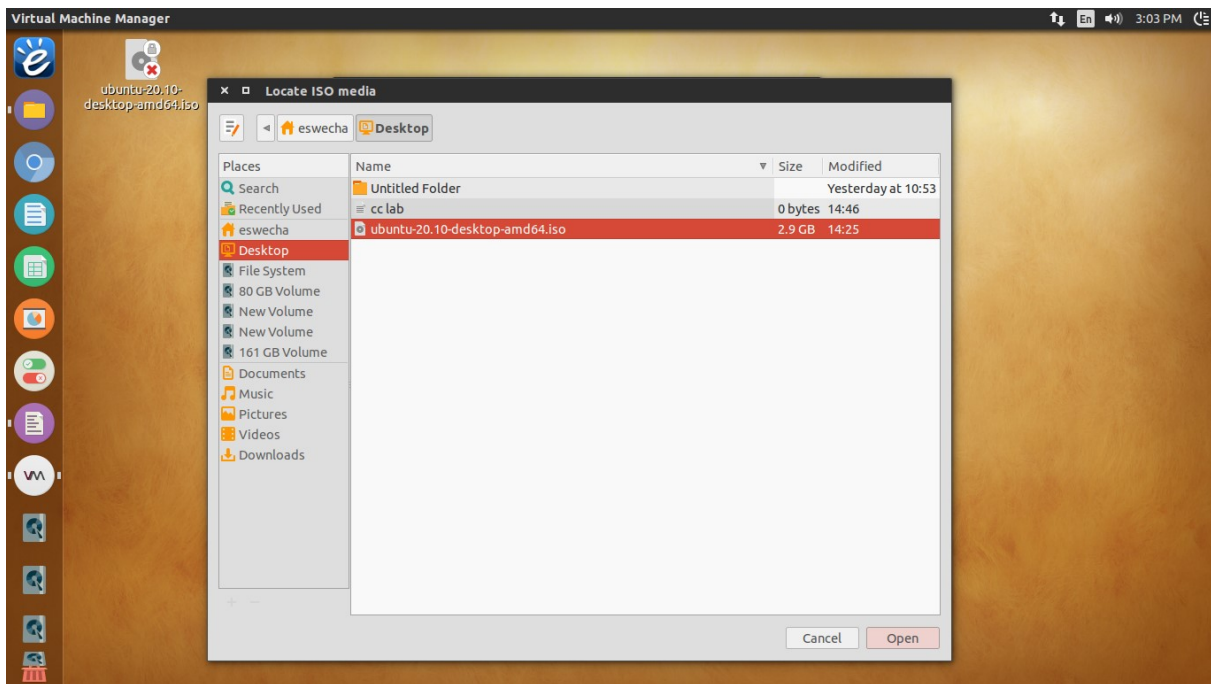
step6:Then enter name of virtual machine.



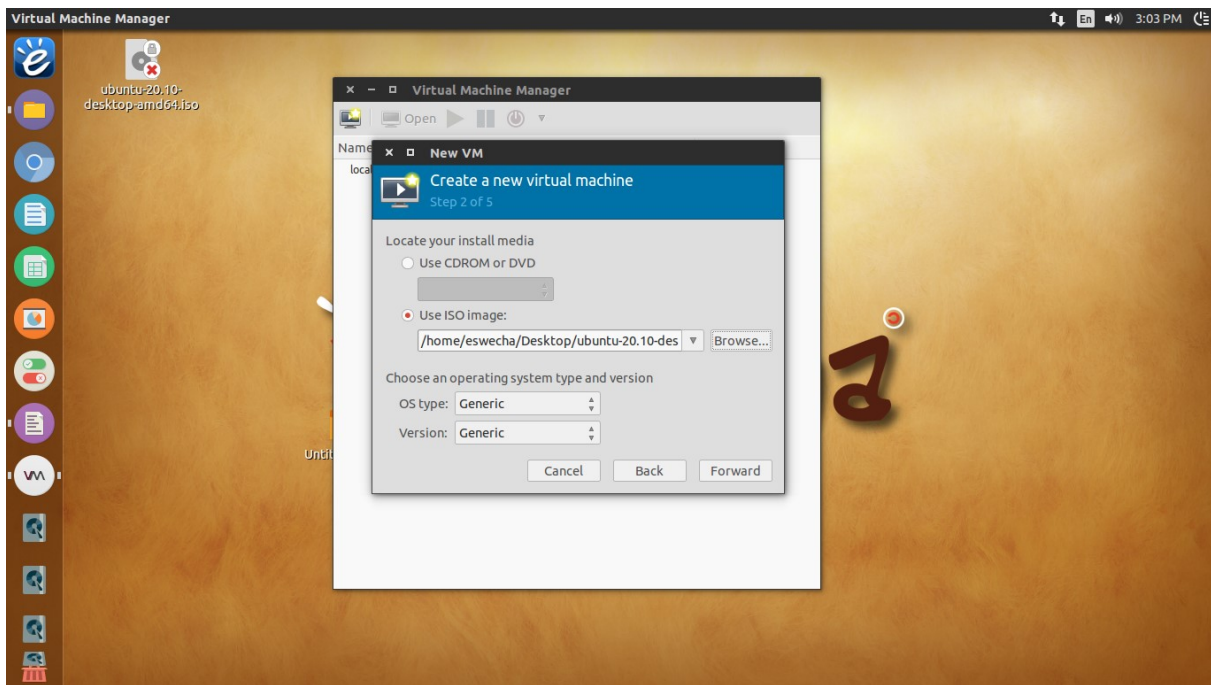
click on forward.



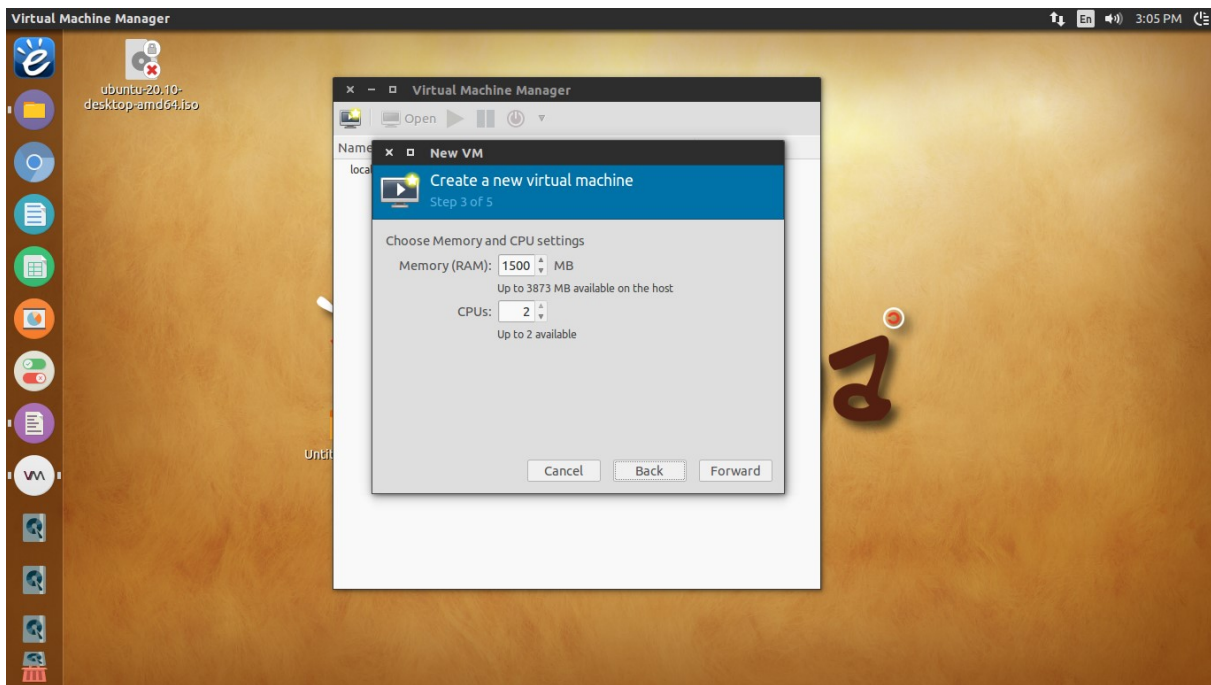
click on Browse.
click on Browse local.



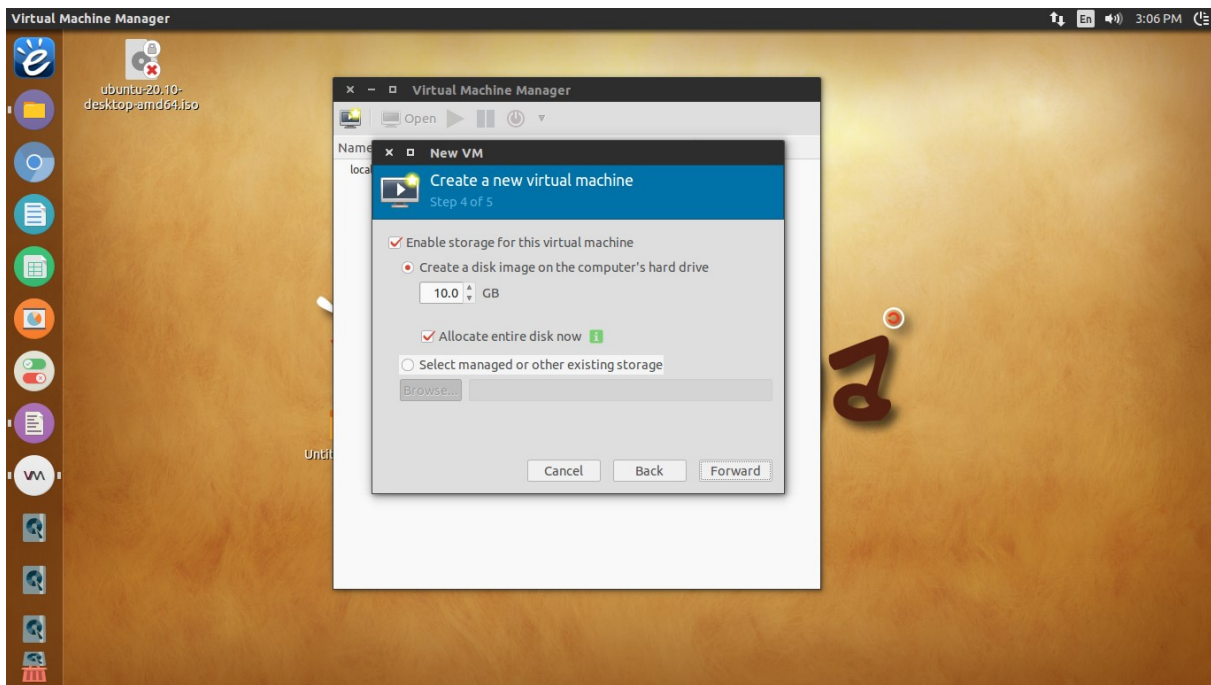
choose iso file.
click on open.



click on forward.



choose the amount of RAM and number of CPUs.



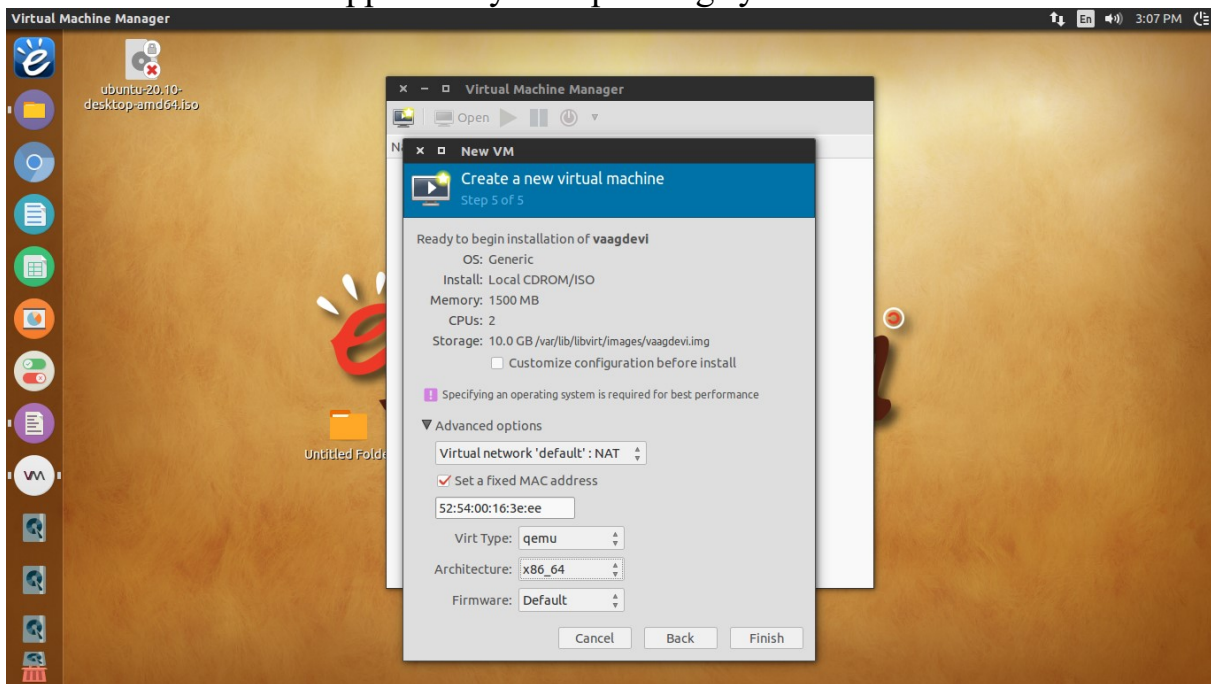
choose the amount of hard disk size to be allocated for new virtual machine.

click on forward.

click on advanced options.

choose qemu from “virt Type” drop down menu.

choose Architecture supported by the operating system.



click on finish.

choose ubuntu/install ubuntu.

3) Study and implementation of Storage as a Service

1. **Aim:** To study and implementation of Storage as a Service

2. **Objectives:** From this experiment, the student will be able to

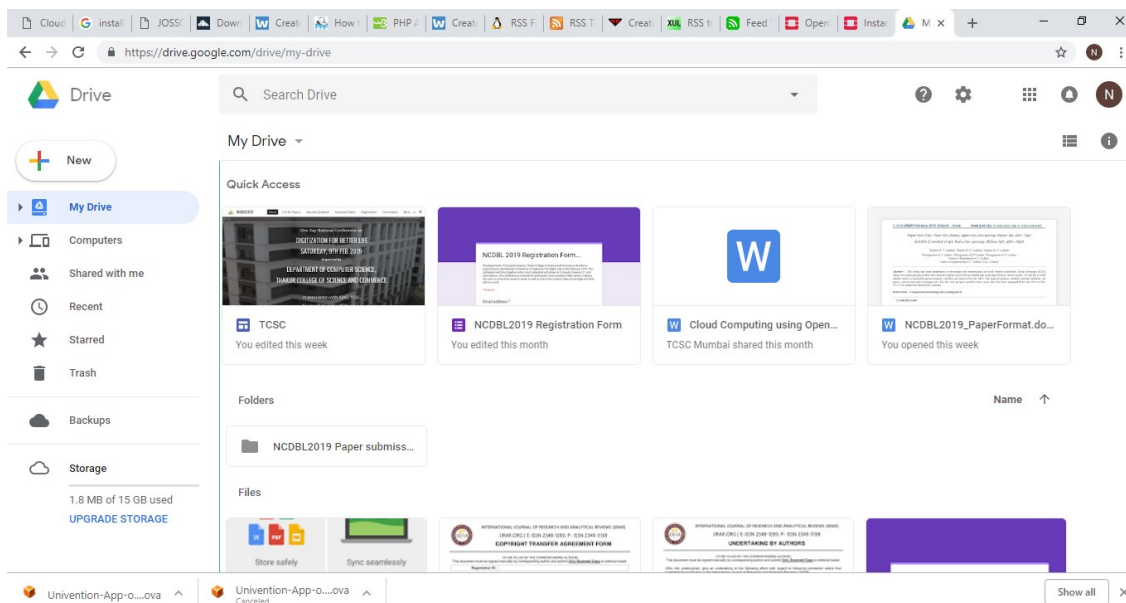
- Understand use of cloud as Platform, Storage as a services.
- Learn the efficient tools to implement the technique

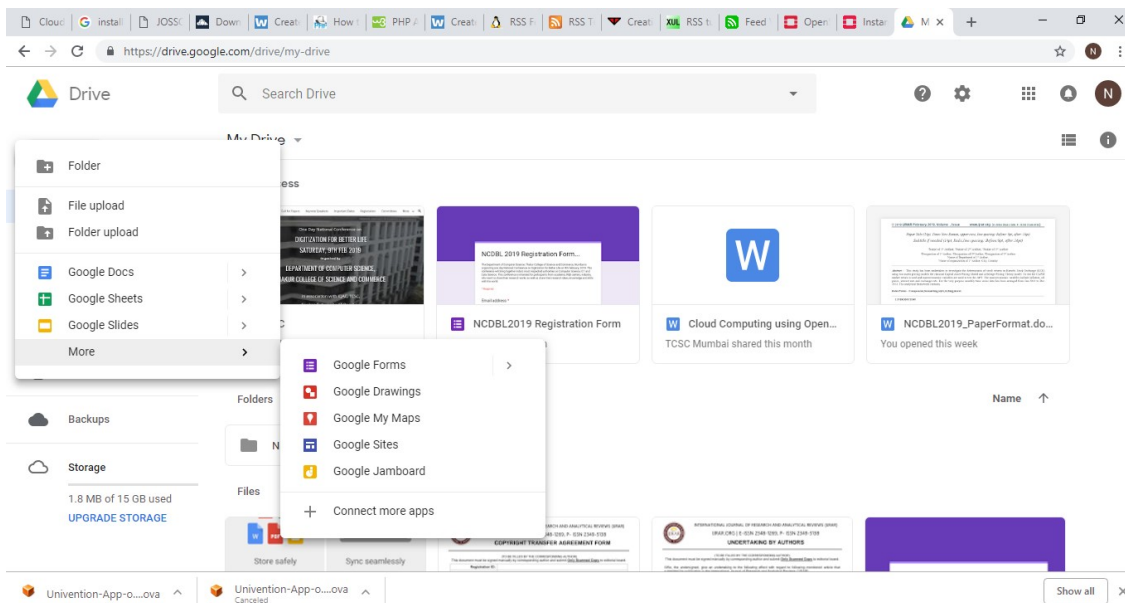
3. **Outcomes:** The learner will be able to

4. **Hardware / Software Required:**

5. **Theory:**

6. **Result:**





7. Conclusion:

Google Docs provide an efficient way for storage of data. It fits well in Storage as a service (SaaS). It has varied options to create documents, presentations and also spreadsheets. It saves documents automatically after a few seconds and can be shared anywhere on the Internet at the click of a button.

4) Write a program for web feed.

RSS - Really Simple Syndication

Concept: Web feed and RSS

Objective: this lab is to understand the concept of form and control validation

Scope: Write a program for web feed

Technology: XML / PHP, HTML

<https://www.w3schools.in/php/php-rss-feed/>

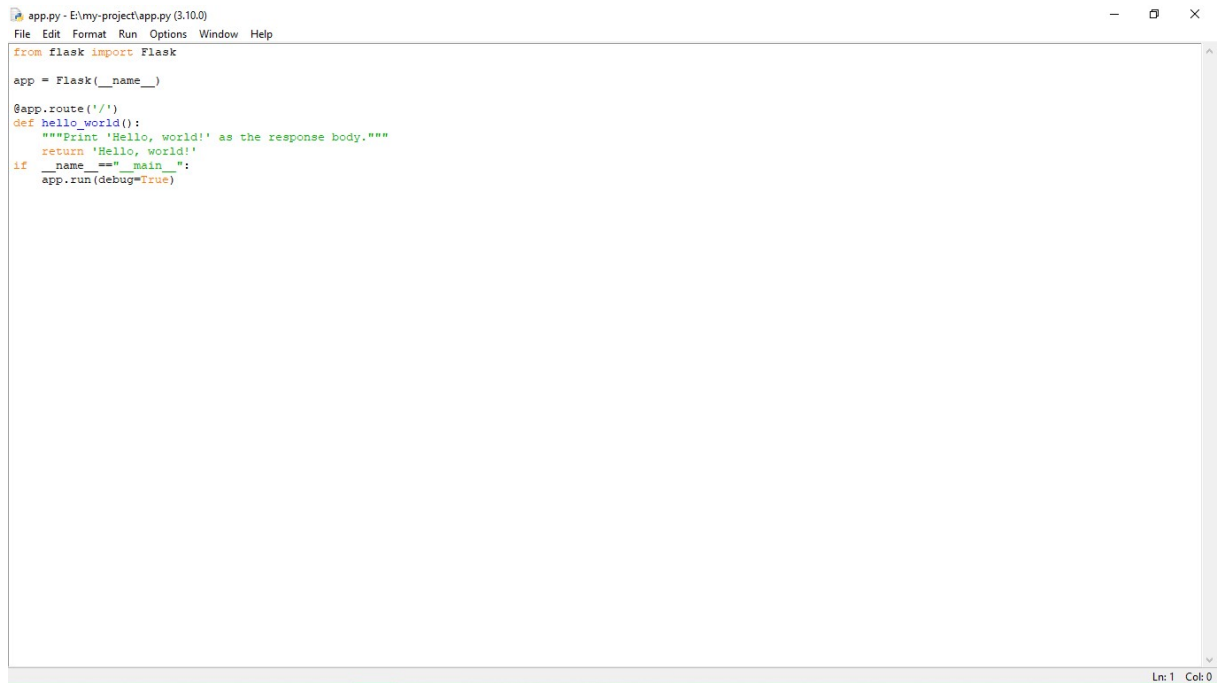
RSS technology is used by millions of users around the world to get the latest information from their favorite websites.

<https://www.xul.fr/en-xml-rss.html>

Building and Using an RSS Feed

5) Create and Deploy a simple hello world app using python in cloud.

1. Create a folder
2. Write source code and save it.



The screenshot shows a code editor window titled 'app.py - E:\my-project\app.py (3.10.0)'. The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code is as follows:

```
from flask import Flask

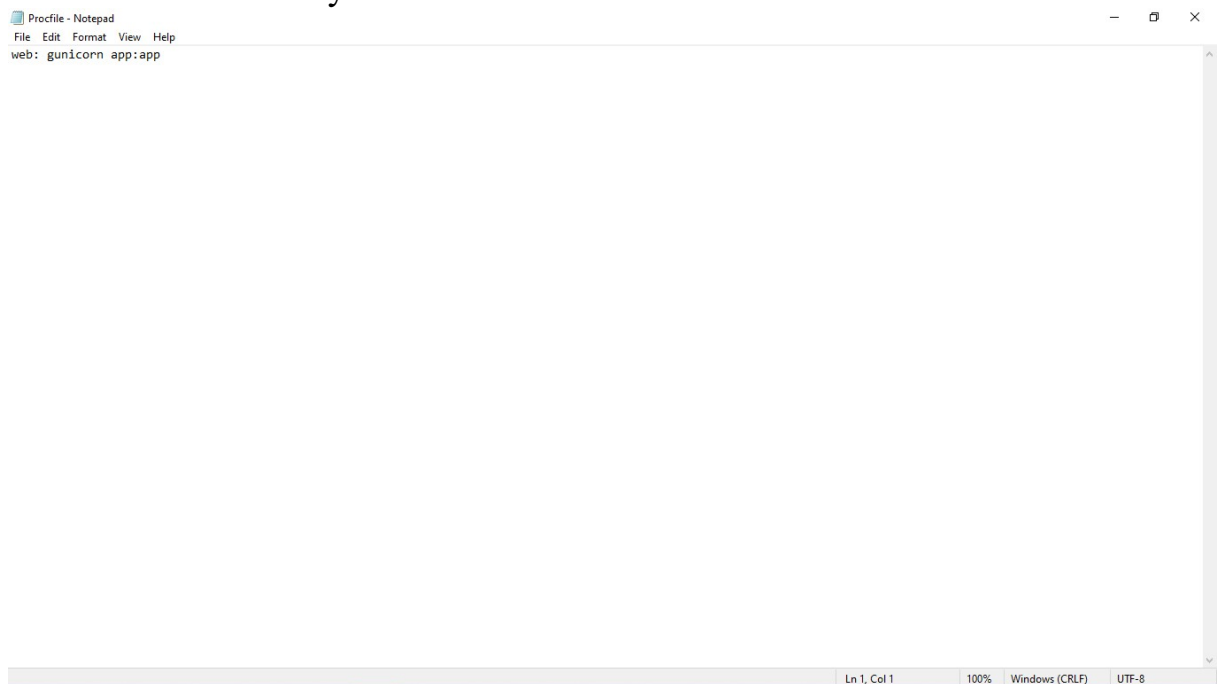
app = Flask(__name__)

@app.route('/')
def hello_world():
    """Print 'Hello, world!' as the response body."""
    return 'Hello, world!'

if __name__ == "__main__":
    app.run(debug=True)
```

The status bar at the bottom right indicates 'Ln: 1 Col: 0'.

3. Write a Procfile
- Procfile is a file which tells the heroku to what to do.
It does not contain any file extension.

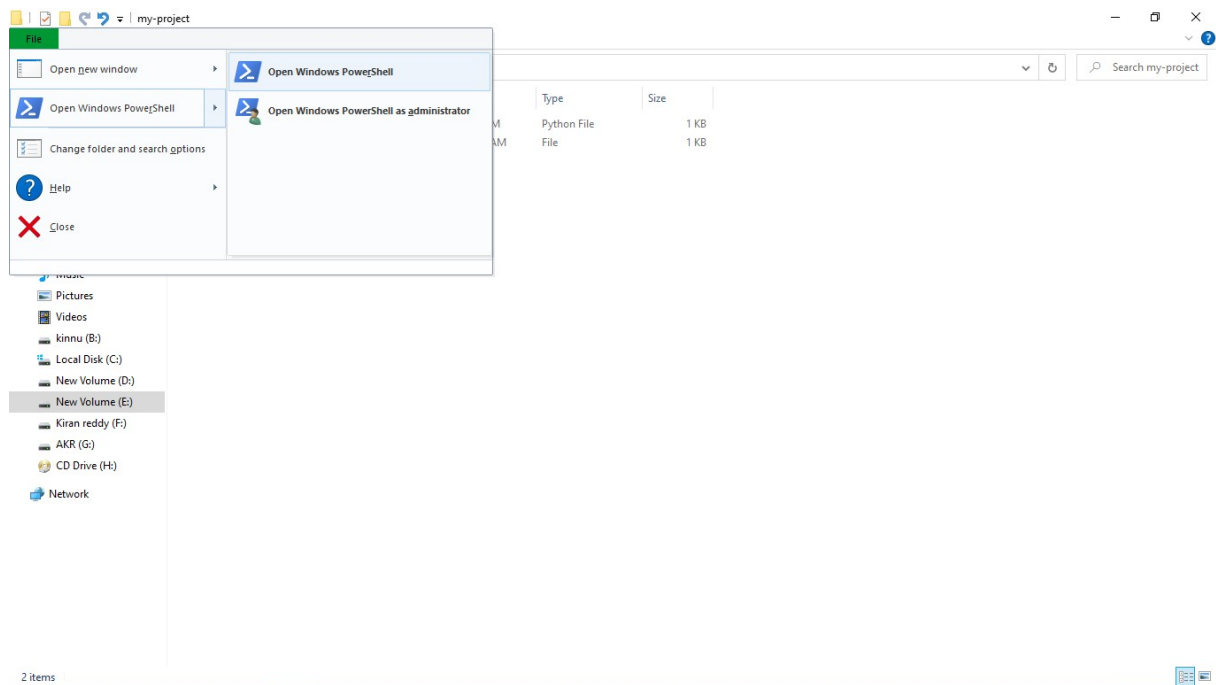


The screenshot shows a Notepad window titled 'Procfile - Notepad'. The menu bar includes File, Edit, Format, View, and Help. The content of the file is:

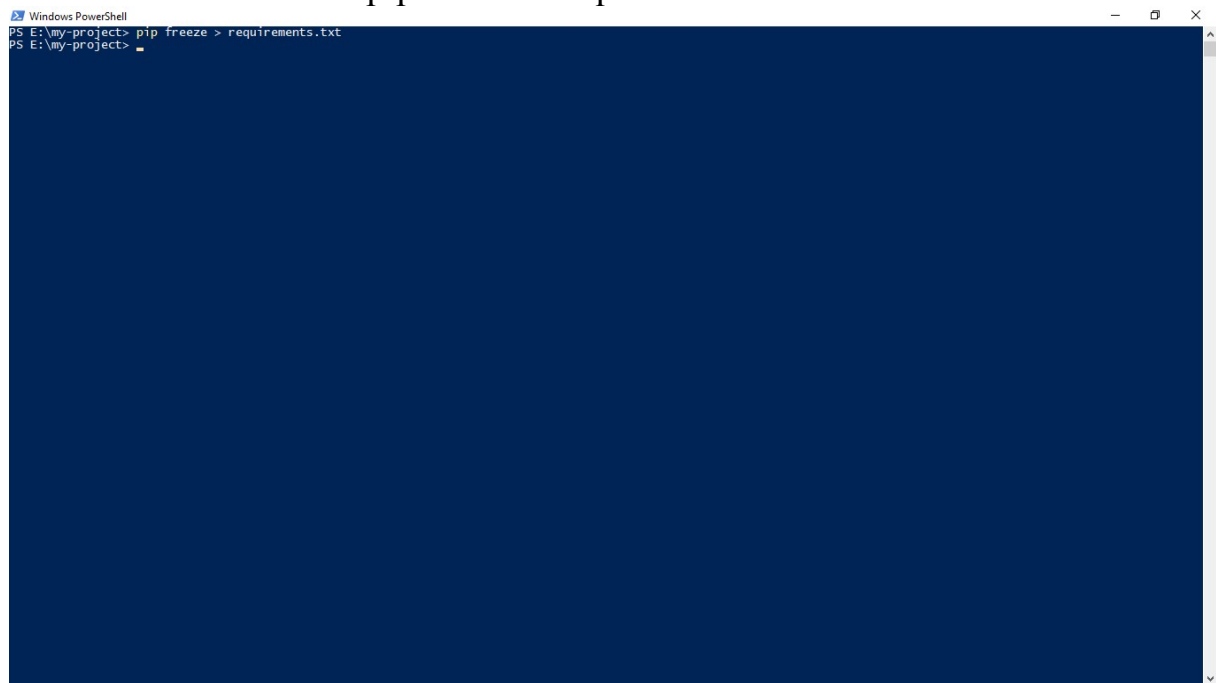
```
web: gunicorn app:app
```

The status bar at the bottom right indicates 'Ln 1, Col 1', '100%', 'Windows (CRLF)', and 'UTF-8'.

4. Create a requirements file using following steps
Open command prompt(cmd) or windows powershell.



Execute the command `pip freeze > requirements.txt`



5. Intiate git repository by using
`>git init`


```
Windows PowerShell
PS E:\my-project> pip freeze > requirements.txt
PS E:\my-project> git init
Initialized empty Git repository in E:/my-project/.git/
PS E:\my-project>
```

6. Add all files to git repository by using
> git add .

```
Windows PowerShell
PS E:\my-project> pip freeze > requirements.txt
PS E:\my-project> git init
Initialized empty Git repository in E:/my-project/.git/
PS E:\my-project> git add .
PS E:\my-project>
```

7. give a message to know what we have done which is called commit. By using
> git commit -m "intial commit"

```
Windows PowerShell
PS E:\my-project> pip freeze > requirements.txt
PS E:\my-project> git init
Initialized empty Git repository in E:/my-project/.git/
PS E:\my-project> git add
PS E:\my-project> git commit -m "initial commit"
[master (root-commit) e82b572] initial commit
3 files changed, 11 insertions(+)
create mode 100644 Profile
create mode 100644 app.py
create mode 100644 requirements.txt
PS E:\my-project>
```

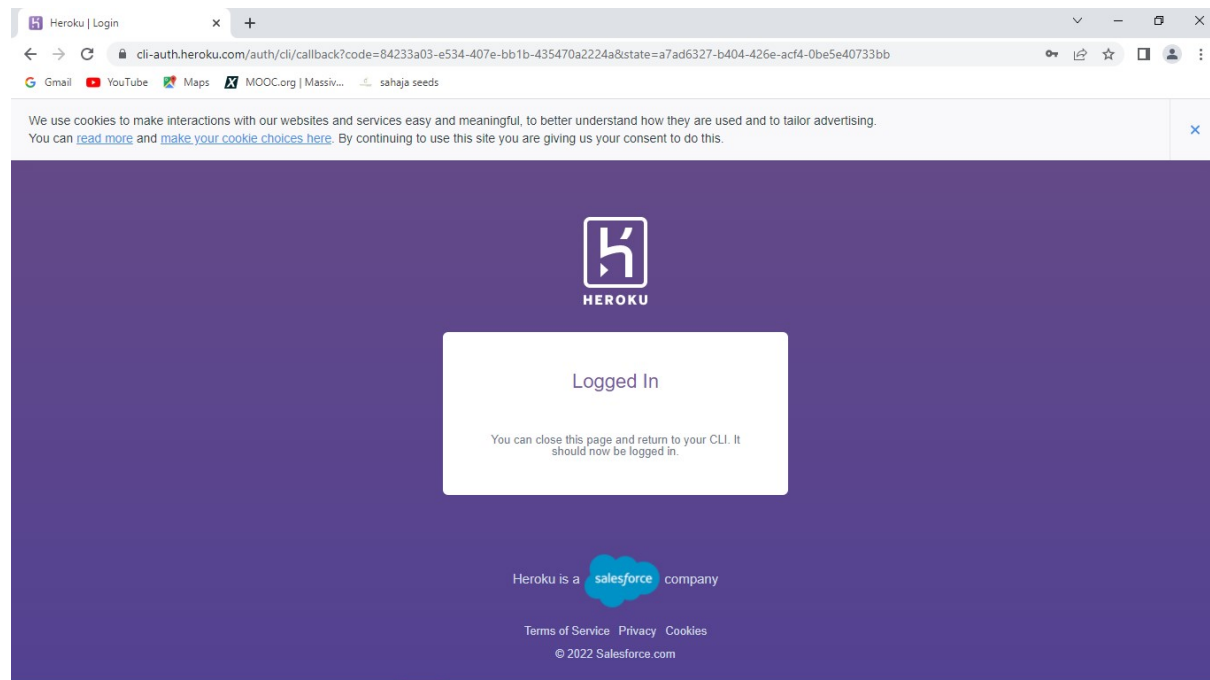
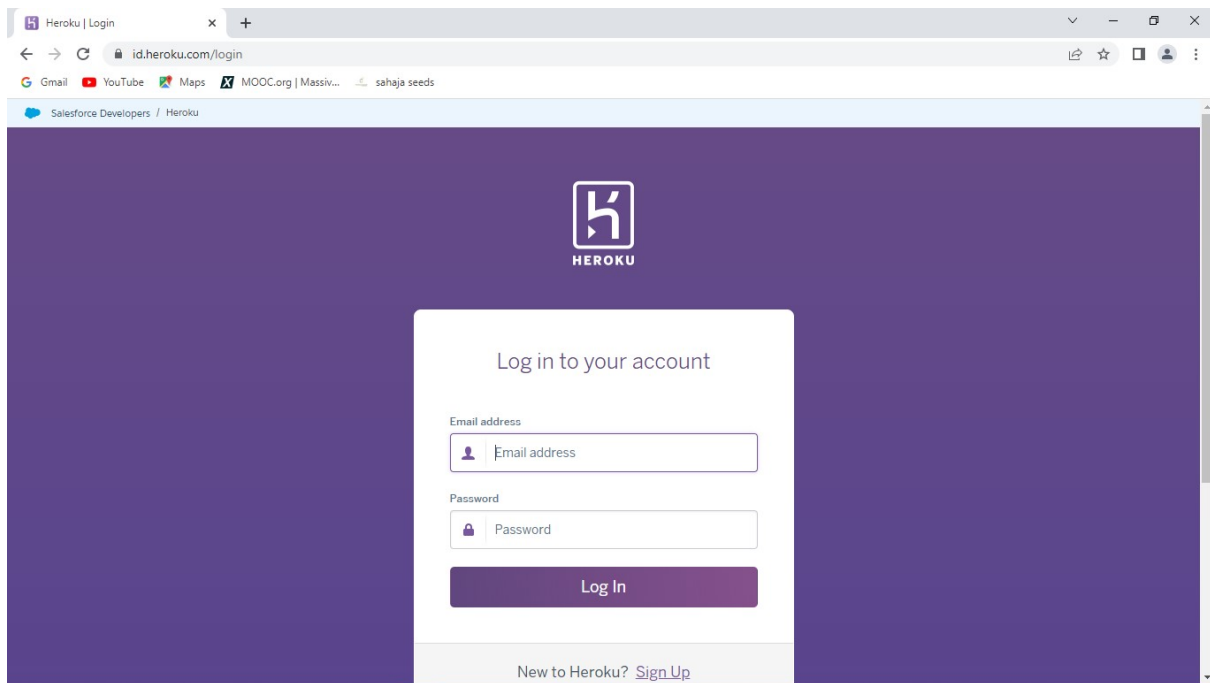
Process of adding files to git repository has been finished.

8. log in to heroku by using

> heroku login

```
Windows PowerShell
PS E:\my-project> heroku login
> Warning: heroku update available from 7.53.0 to 7.60.1.
heroku: Press any key to open up the browser to login or q to exit:
```

Press any key to continue to login in default browser
Click on login and Enter email address and password.



9. Create a web application using

> heroku create appname

Appname should be unique, if it is already taken you need choose another one

```
Windows PowerShell
PS E:\my-project> heroku create my-app2022
* Warning: heroku update available from 7.53.0 to 7.60.1.
Creating my-app2022... done
https://my-app2022.herokuapp.com/ | https://git.heroku.com/my-app2022.git
PS E:\my-project>
```

10. push all the files in repository to the master branch.

> git push heroku master

It will install all the requirements mentioned in requirements.txt file and deploy's the app in the cloud.

```
Windows PowerShell
PS E:\my-project> heroku create my-app2022
* Warning: heroku update available from 7.53.0 to 7.60.1.
Creating my-app2022... done
https://my-app2022.herokuapp.com/ | https://git.heroku.com/my-app2022.git
PS E:\my-project> git push heroku master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 851 bytes | 425.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-20 stack
remote: -----> Determining which buildpack to use for this app
remote: -----> Python app detected
remote: -----> No Python version was specified. Using the buildpack default: python-3.10.4
remote: -----> To use a different version, see: https://devcenter.heroku.com/articles/python-runtimes
remote: -----> Installing python-3.10.4
remote: -----> Installing pip 22.0.4, setuptools 60.10.0 and wheel 0.37.1
remote: ----->
```

```
Windows PowerShell
remote: Collecting MarkupSafe==2.1.1
remote:   Downloading MarkupSafe-2.1.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (25 kB)
remote: Collecting platformdirs==2.5.1
remote:   Downloading platformdirs-2.5.1-py3-none-any.whl (14 kB)
remote: Collecting psycpg2==2.9.3
remote:   Downloading psycpg2-2.9.3.tar.gz (380 kB)
remote:   Preparing metadata (setup.py): started
remote:   Preparing metadata (setup.py): finished with status 'done'
remote: Collecting requests==2.27.1
remote:   Downloading requests-2.27.1-py2.py3-none-any.whl (63 kB)
remote: Collecting six==1.16.0
remote:   Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
remote: Collecting sqlparse==0.4.2
remote:   Downloading sqlparse-0.4.2-py3-none-any.whl (42 kB)
remote: Collecting tzdata==2022.1
remote:   Downloading tzdata-2022.1-py2.py3-none-any.whl (339 kB)
remote: Collecting urllib3==1.26.9
remote:   Downloading urllib3-1.26.9-py2.py3-none-any.whl (138 kB)
remote: Collecting virtualenv==20.14.1
remote:   Downloading virtualenv-20.14.1-py2.py3-none-any.whl (8.8 MB)
remote: Collecting Werkzeug==2.1.1
remote:   Downloading Werkzeug-2.1.1-py3-none-any.whl (224 kB)
remote: Collecting whitenoise==6.0.0
remote:   Downloading whitenoise-6.0.0-py3-none-any.whl (19 kB)
remote: Building wheels for collected packages: psycpg2
remote:   Building wheel for psycpg2 (setup.py): started
remote:   Building wheel for psycpg2 (setup.py): finished with status 'done'
remote:   Created wheel for psycpg2: filename=psycpg2-2.9.3-cp310-cp310-linux_x86_64.whl size=586483 sha256=83dd032460422f0fb2dac1edee9f8036326c369c2d1c6be93be
remote:   Stored in directory: /tmp/pip-ephem-wheel-cache-skcknq3a/wheels/81/b6/3d/091aad3e8919ea76c84c2674b02ce3ab52de882e091c39249e
remote: Successfully built psycpg2
remote: Installing collected packages: dj-database-url, distlib, certifi, whitenoise, Werkzeug, urllib3, tzdata, sqlparse, six, psycpg2, platformdirs, MarkupSafe, itsdangerous, idna, gunicorn, filelock, colorama, click, charset-normalizer, asgiref, virtualenv, requests, Jinja2, Django, Flask, django-heroku
remote: Successfully installed Django-4.0.3 Flask-2.1.1 Jinja2-3.1.1 MarkupSafe-2.1.1 Werkzeug-2.1.1 asgiref-3.5.0 certifi-2021.10.8 charset-normalizer-2.0.12 click-8.1.2 colorama-0.4.4 dj-database-url-0.5.0 django-heroku-0.3.1 filelock-3.6.0 gunicorn-20.1.0 idna-3.3 itsdangerous-2.1.2 platformdirs-2.5.1 psycpg2-2.9.3 requests-2.27.1 six-1.16.0 sqlparse-0.4.2 tzdata-2022.1 urllib3-1.26.9 virtualenv-20.14.1 whitenoise-6.0.0
remote: -----> Skipping Django collectstatic since no manage.py file found.
remote: -----> Discovering process types
remote:   Procfile declares types -> web
remote: -----> Compressing...
remote:   Done: 79.3M
remote: -----> Launching...
remote:   Released v3
remote:   https://my-app2022.herokuapp.com/ deployed to Heroku
remote: Verifying deploy... done.
To https://git.heroku.com/my-app2022.git
 * [new branch] master -> master
PS E:\my-project>
```

Check the application in the highlighted URL.



The application is displayed perfectly.