DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DSEU DWARKA CAMPUS,

Sector 9, Dwarka, New Delhi

2021-2024

SYSTEM ANALYSIS & DESIGN

LAB FILE

SUBMITTED BY:

ANIKET

(10621084)

SUBMITTED TO:

SHIVANI ANAND

(Faculty)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DSEU DWARKA CAMPUS,

Sector 9, Dwarka, New Delhi

2021-2024

SYSTEM ANALYSIS & DESIGN

LAB FILE

SUBMITTED BY:

ROHIT

(10621432)

SUBMITTED TO:

SHIVANI ANAND

(Faculty)

Practicals

S. no.	Topic	Date
1.	to install virtual box / VMware workstation with different OS (Window/Linux) on exiting windows	
2.	Install C/Python compiler in VM created using virtual box and execute simple C/python programs	
3.	To install Google App Engine.	
4.	To launch the Web application by using GAE Launch.	
5.	Write the procedure to transfer file from a virtual machine to other machine	
6.		

Practical - 1

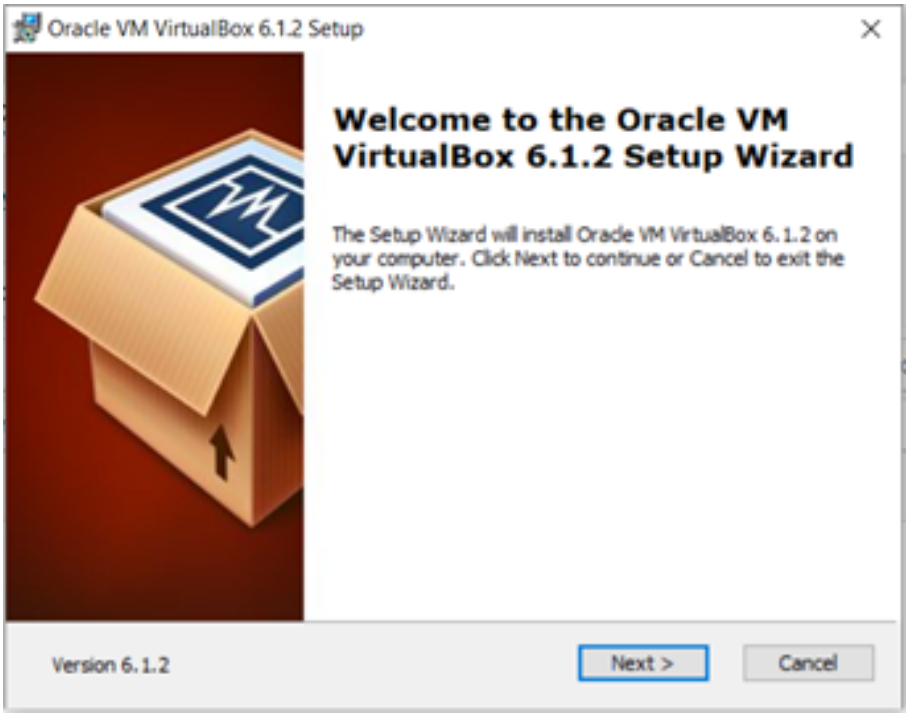
Aim: **to install virtual box / VMware workstation with different OS (Window/Linux) on exiting windows**

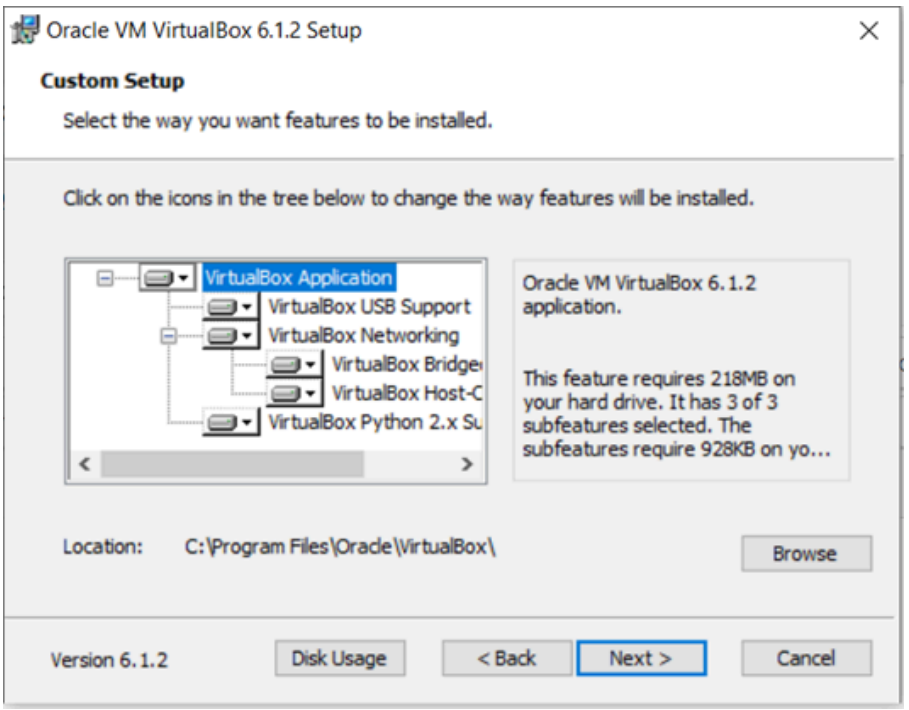
Step 1: Download VirtualBox:

1. Go to the official VirtualBox website: https://www.virtualbox.org/
2. Click on the "Downloads" link.
3. Download the version of VirtualBox that corresponds to your host operating system (in this case, Windows).

Step 2: Install VirtualBox:

1. Once the download is complete, locate the installer file (usually in your Downloads folder) and double-click it to start the installation process.
2. Follow the on-screen instructions to install VirtualBox. You can leave most settings as default unless you have specific preferences.

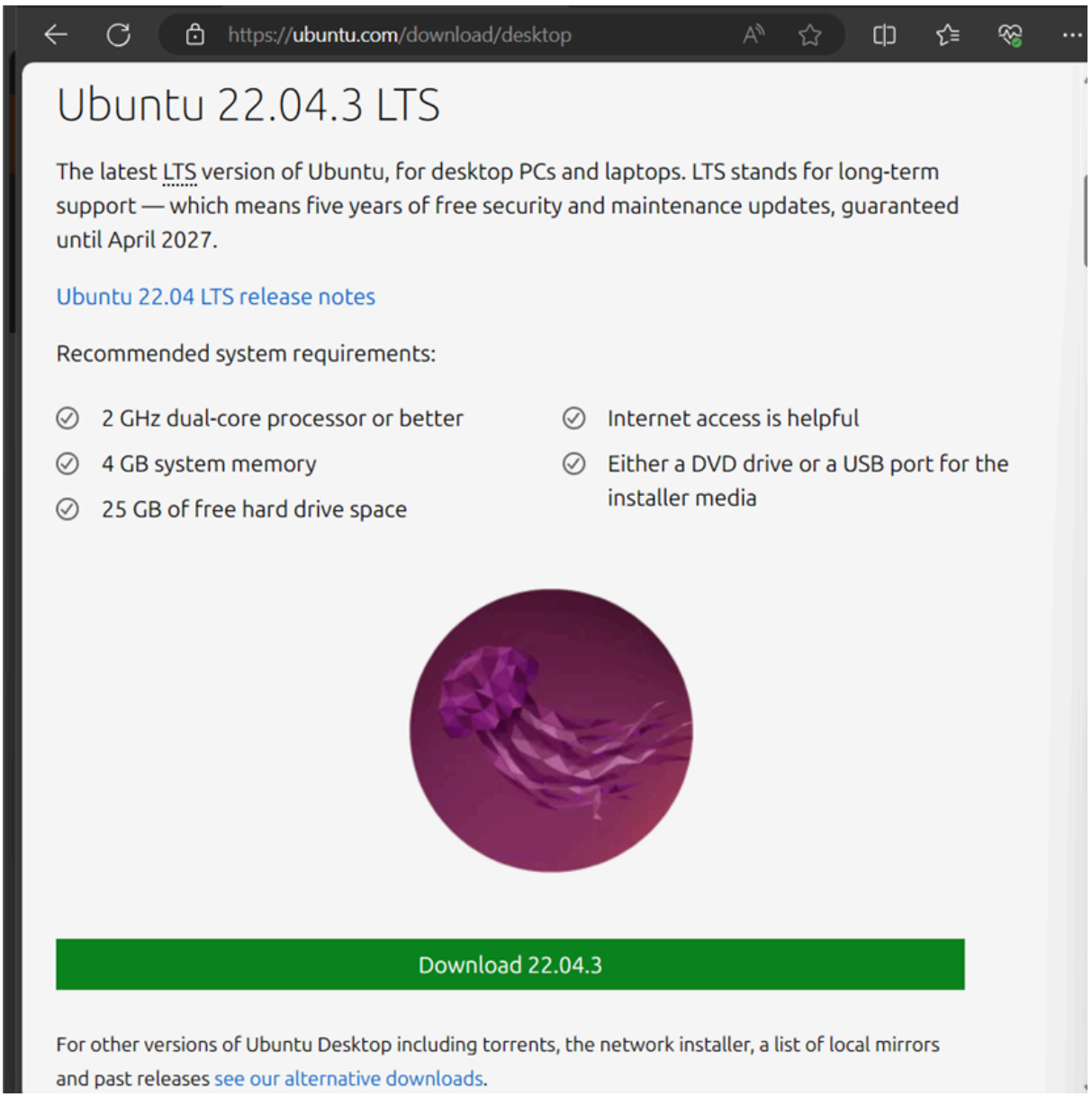




Step 3: Download a Linux ISO:

1. Decide on which Linux distribution you want to install in your virtual machine (e.g., Ubuntu, CentOS, Debian).

2. Go to the official website of the chosen Linux distribution and download the ISO file for the version you want to use.



Step 4: Create a New Virtual Machine:

1. Open VirtualBox after installation.
2. Click on the "New" button in the toolbar.
3. Enter a name for your virtual machine (e.g., "Ubuntu VM").
4. Select the type of operating system (Linux) and the version (e.g., Ubuntu 64-bit).
5. Click "Next."
6. Allocate memory (RAM) to your virtual machine. It's recommended to allocate at least 1GB, but you can adjust this based on your system resources.
7. Click "Next."
8. Choose "Create a virtual hard disk now" and click "Create."

9. Select the hard disk file type. The default (VDI) should be fine.

10. Choose whether to dynamically allocate the virtual hard disk space or to allocate a fixed amount. Dynamic allocation is usually preferred unless you have specific requirements.

11. Choose the size of the virtual hard disk. The default size should be sufficient for most purposes.

12. Click "Create."

Step 5: Configure Virtual Machine Settings:

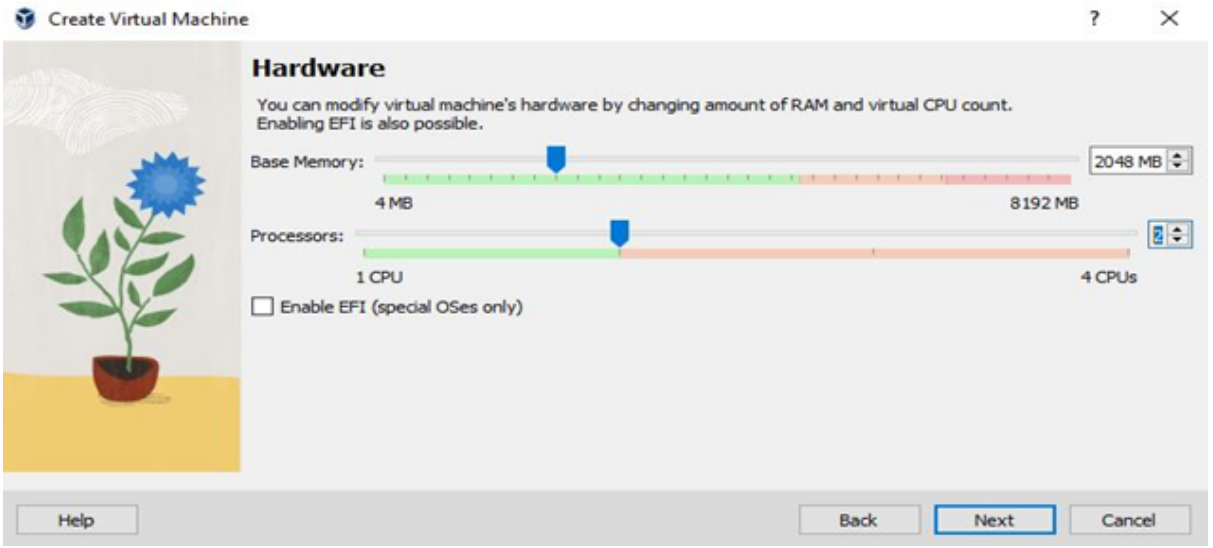
1. Select the virtual machine you just created from the list on the left side.

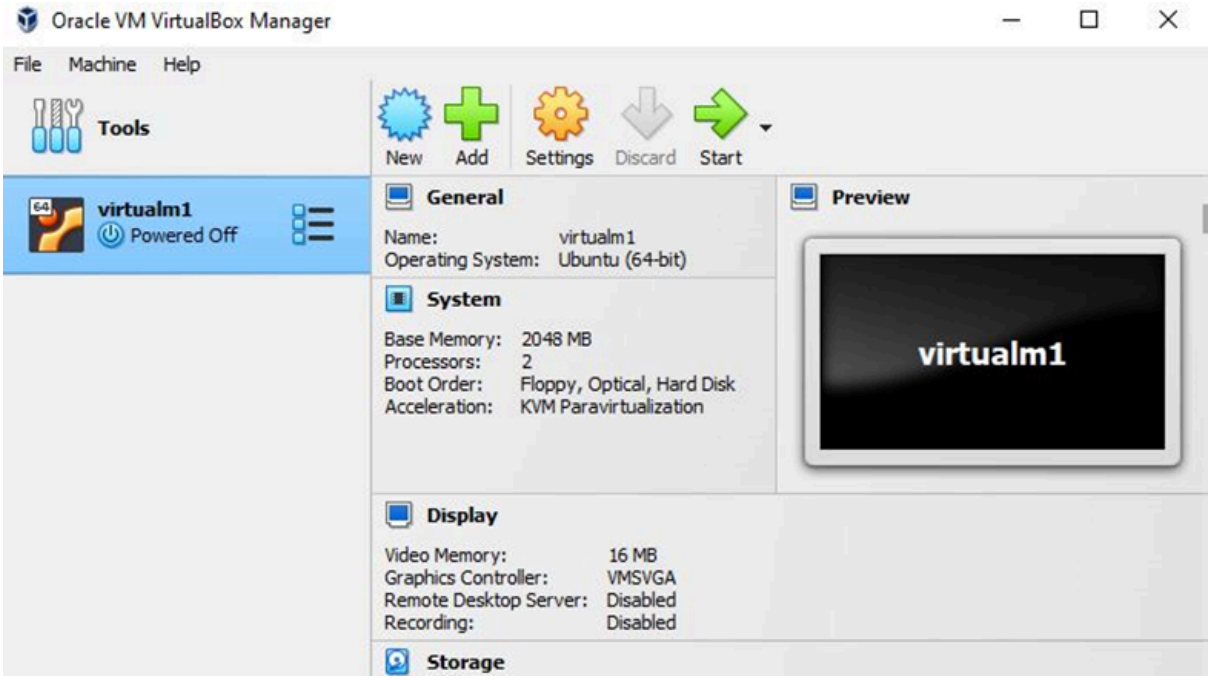
2. Click on the "Settings" button.

3. In the settings window, go through each section to configure options such as storage, network, display, etc.

4. Under "Storage," click on the empty disk icon next to "Controller: IDE" and select the Linux ISO file you downloaded earlier.

5. Click "OK" to save the settings.





Step 6: Install Linux:

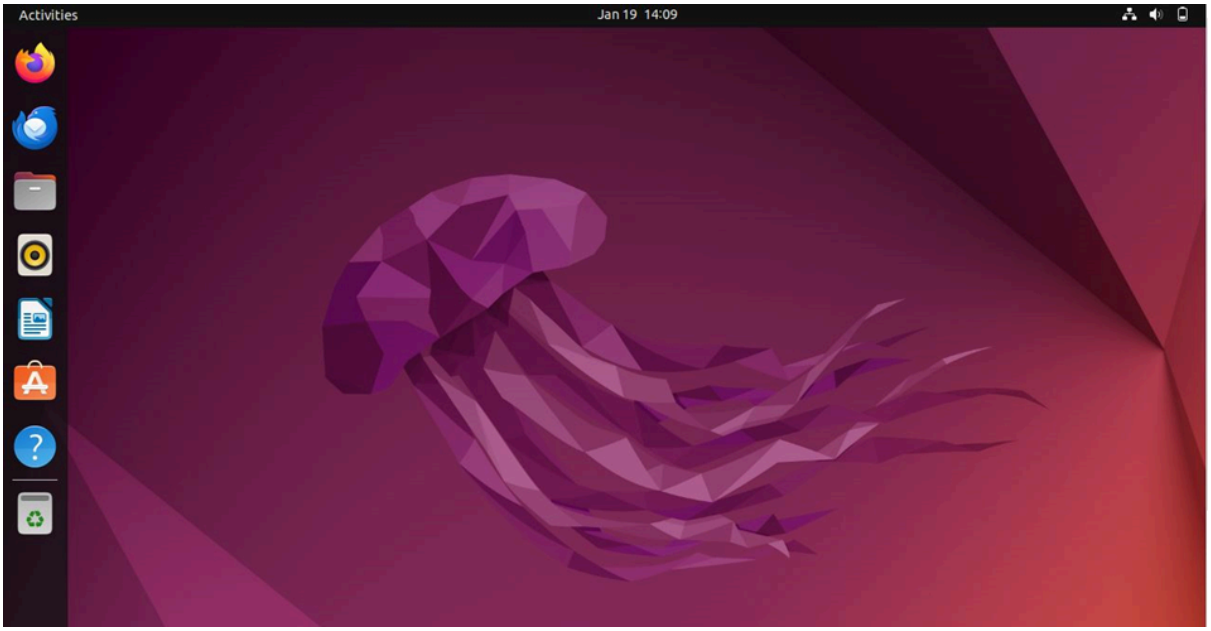
1. Start the virtual machine by clicking on the "Start" button in VirtualBox.

2. The virtual machine will boot from the Linux ISO file you attached. Follow the on-screen instructions to install Linux within the virtual machine.

3. Once the installation is complete, the virtual machine will restart.

4. You may need to install VirtualBox Guest Additions inside the virtual machine for better integration and performance. This is usually found under the "Devices" menu in the virtual machine window.

Final Preview:



Practical - 2

Aim: **Install C/Python compiler in VM created using virtual box and execute simple C/python programs**

Step 1: Install C Compiler (GCC)

1. Once the Linux distribution is installed and running in the virtual machine, open a terminal.

2. Install GCC by running: **&Sudo apt install gcc**

```
gaurav@gaurav-VirtualBox:~$ sudo apt install gcc
[sudo] password for gaurav:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gcc is already the newest version (4:11.2.0-1ubuntu1).
gcc set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 198 not upgraded.
gaurav@gaurav-VirtualBox:~$ gcc  --version
gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

gaurav@gaurav-VirtualBox:~$
```

Step 2: Install Python

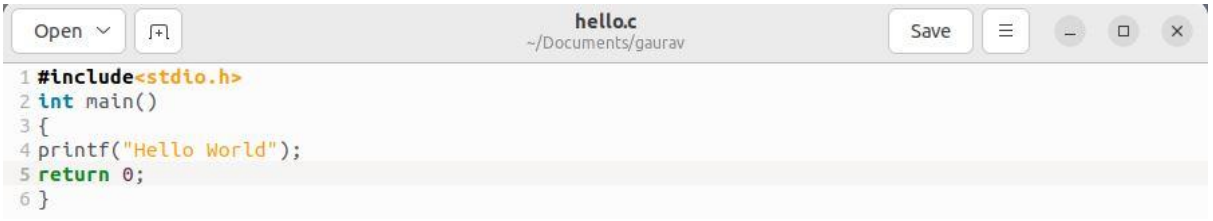
Python usually comes pre-installed with most Linux distributions, but you can ensure you have the latest version by running: **&sudo apt install python3**

```
gaurav@gaurav-VirtualBox:~$ sudo apt install python3
[sudo] password for gaurav:
Sorry, try again.
[sudo] password for gaurav:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.10.6-1~22.04).
0 upgraded, 0 newly installed, 0 to remove and 198 not upgraded.
```

This command will install Python 3.x, which is the recommended version.

Step 3: Write and Execute Simple C/Python Programs

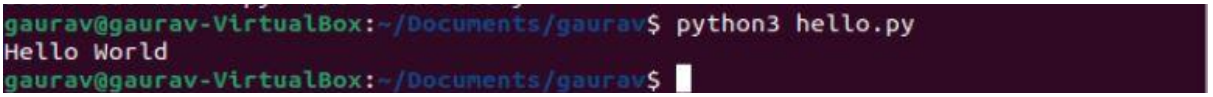
1. Open a text editor (e.g., Nano, Vim, or Gedit) to write your C and Python programs.
2. Write your C program (e.g., `hello.c`):



3. Save the file.
4. Compile the C program by running:5. Run the compiled C program:

6. Write your Python program (e.g., `hello.py`):

7. Save the file.
8. Run the Python program:



Practical - 3

Aim: **To install Google App Engine.**

Materials Needed:

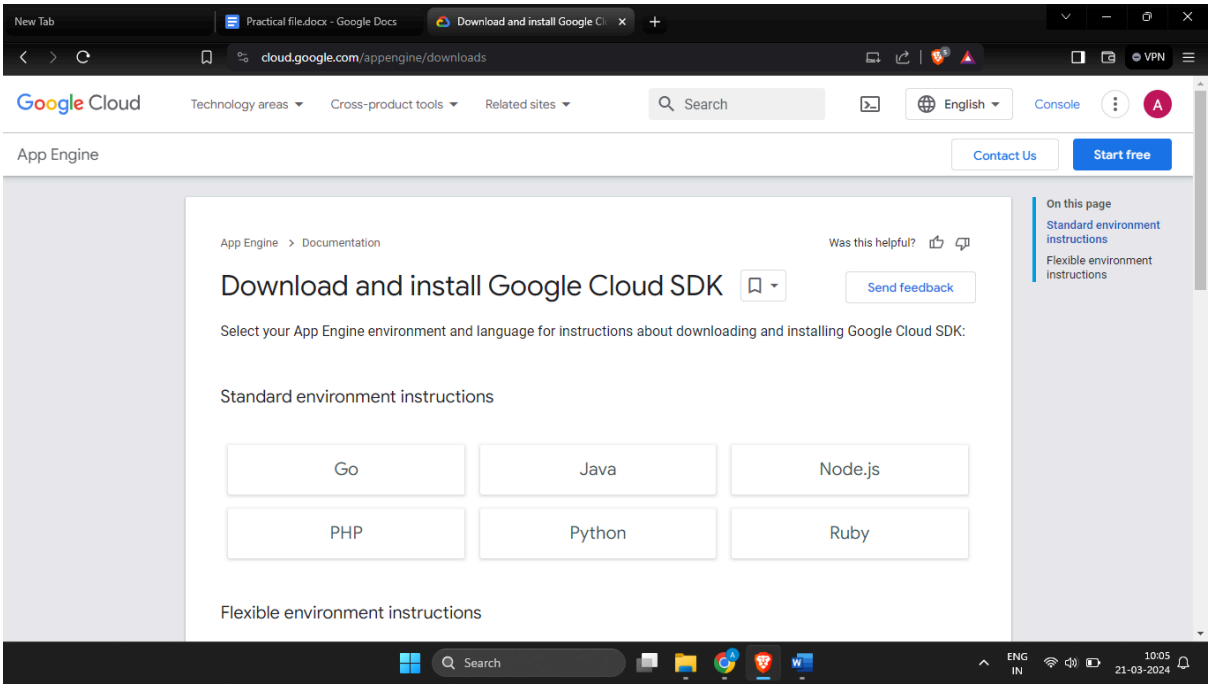
1. Computer with internet access
2. Python installed (version 2.7 or 3.7 recommended)
3. Google account

Theory:

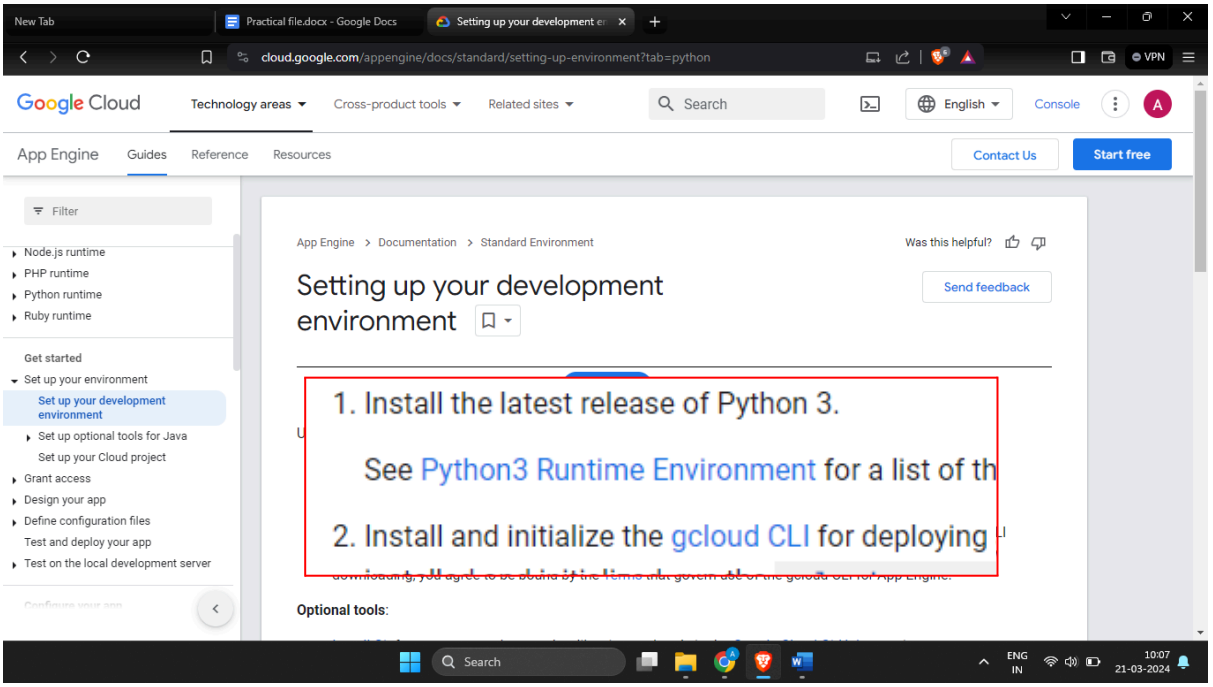
App Engine is a fully managed, serverless platform for developing and hosting web applications at scale. You can choose from several popular languages, libraries, and frameworks to develop your apps, and then let App Engine take care of provisioning servers and scaling your app instances based on demand. Overall, users appreciate Google App Engine for its reliability, scalability, ease of use, language flexibility, fast response times from the support team, cost-effectiveness, and seamless integration with other Google products

Step-by-Step Procedure:

Step 1: Go to cloud.google.com/appengine/downloads and click Python

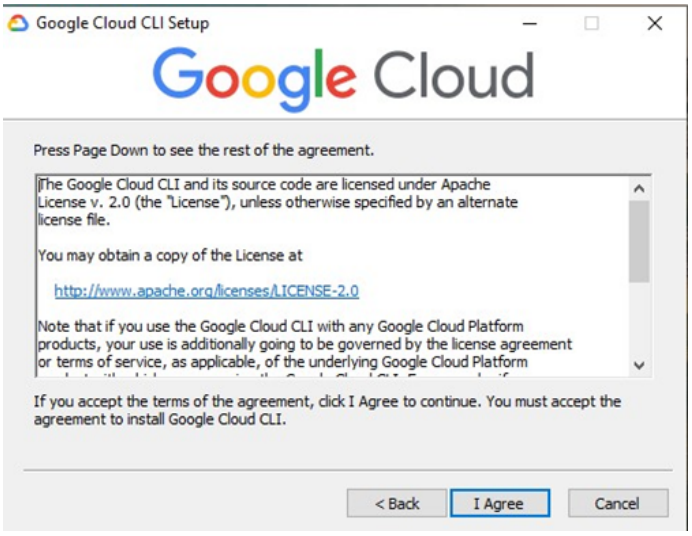


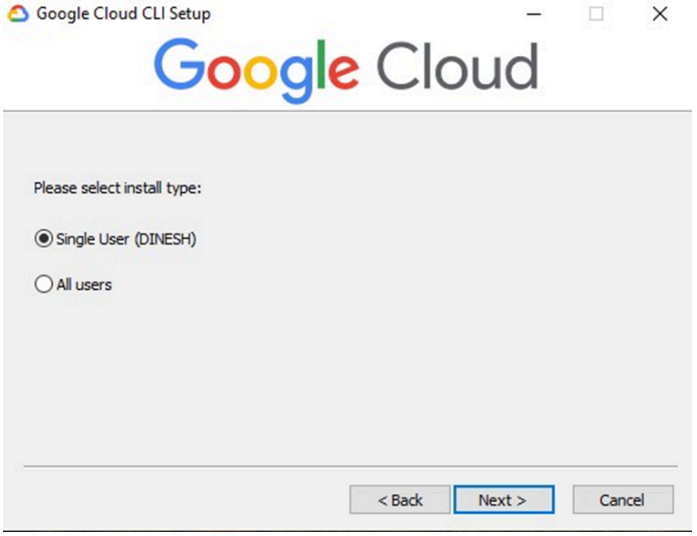
Step 2: Select setting up your environment development .Click install and initialize the gcloud CLI

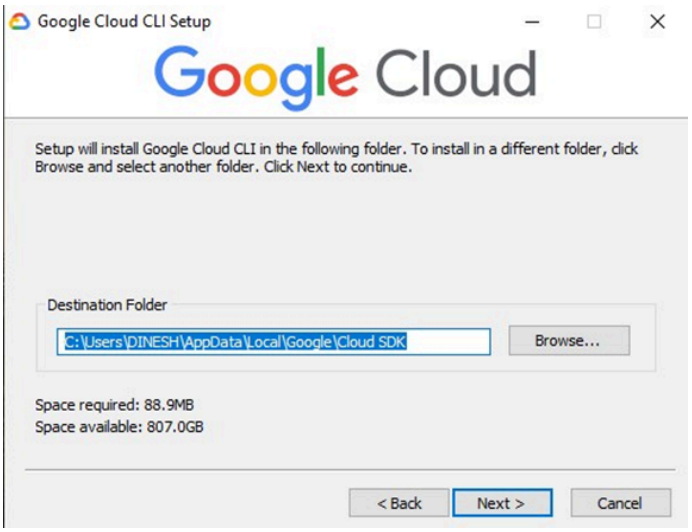


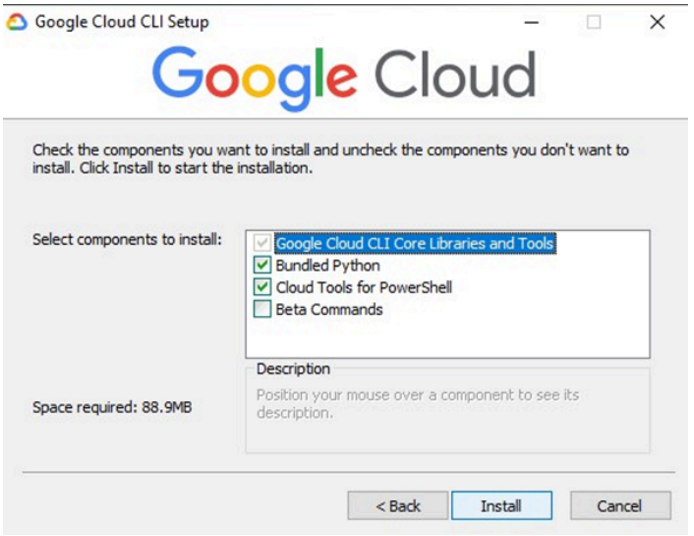
Step 3: Download CLI installer and install it

Step 4: Click Next -> Click “I Agree”-> Select “single user” -> Click finish

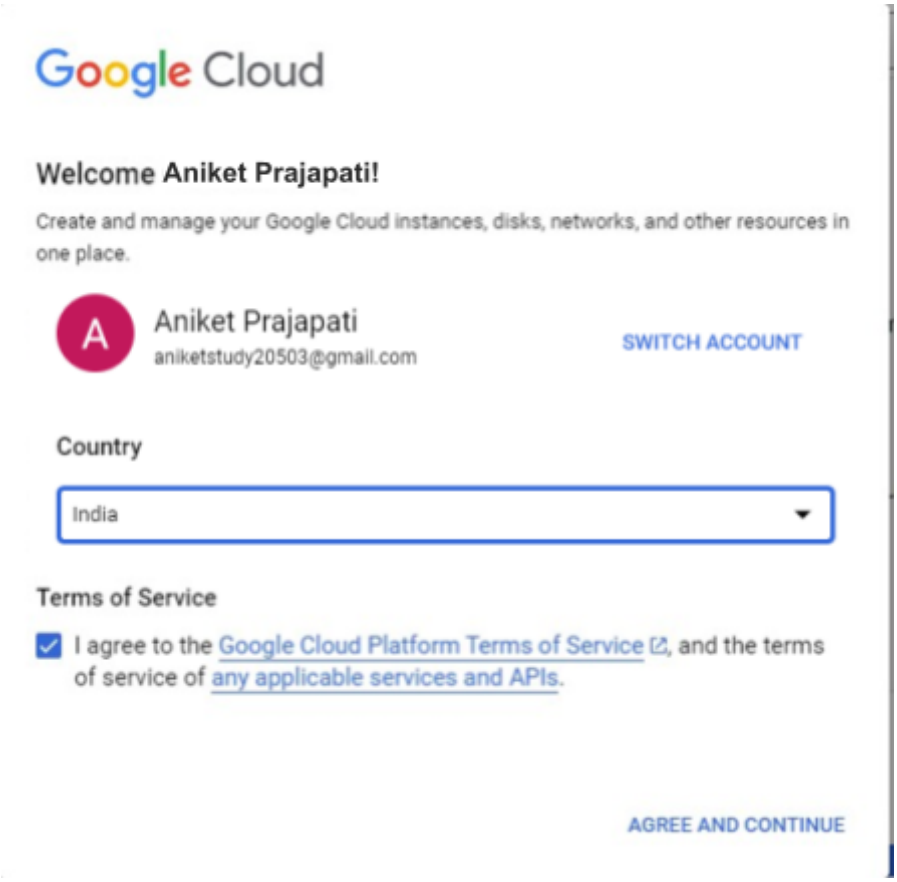




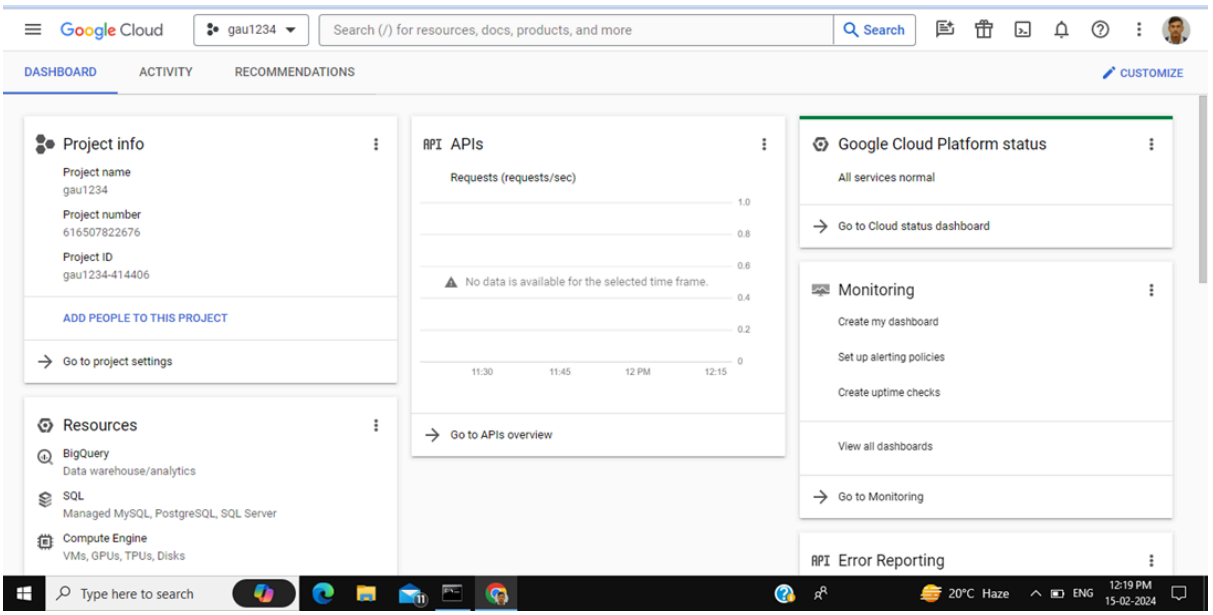




Step 5: Once successfully installed cmd line in login with your google account.



Result:

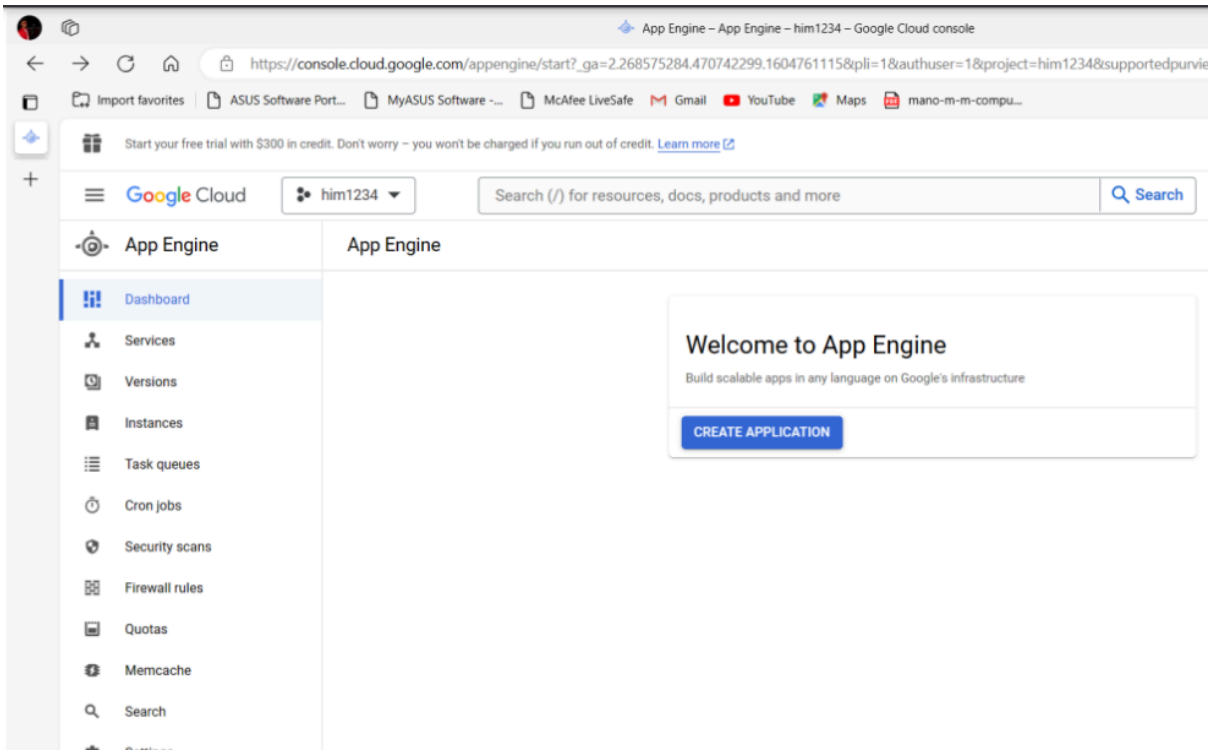


Google Engine is successfully installed

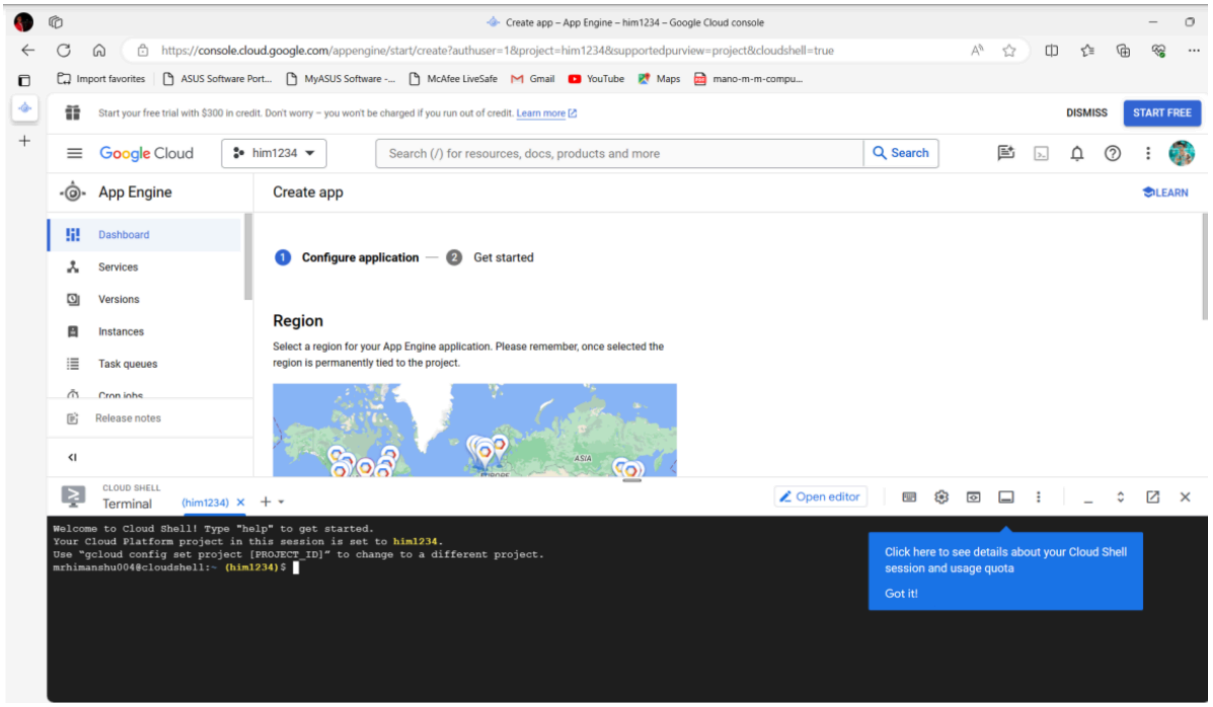
Practical - 4

Aim: **- To launch the Web application by using GAE Launch.**

Step 1: Go to GCloud App Engine Dashboard & Create Application



Step 2: Select Python and click next -> Open cloud shell



Step 3: run cmd

> git clone\ https://github.com/googleCloudPlatform/pyhton-docs-samples

```
mrhimanshu004@cloudshell:~ (him1234)$ git clone \
> https://github.com/googlecloudplatform/python-docs-samples
Cloning into 'python-docs-samples'...
remote: Enumerating objects: 109067, done.
remote: Counting objects: 100% (705/705), done.
remote: Compressing objects: 100% (453/453), done.
remote: Total 109067 (delta 361), reused 511 (delta 239), pack-reused 108362
Receiving objects: 100% (109067/109067), 223.24 MiB | 16.00 MiB/s, done.
Resolving deltas: 100% (64187/64187), done.
Updating files: 100% (4899/4899), done.
```

Step 4: Install requirement & run app

>$ pip install -r requirement.txt

```
(him1234)$ pip install -r requirements.txt
```

```
Collecting Flask==3.0.0
  Downloading flask-3.0.0-py3-none-any.whl (99 kB)
     |████████████████████████████████| 99 kB 1.9 MB/s
```

```
mrhimanshu004@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_world (him1234)$ pip install -r requirements.txt
Collecting Flask==3.0.0
  Downloading flask-3.0.0-py3-none-any.whl (99 kB)
     |████████████████████████████████| 99 kB 1.9 MB/s
Requirement already satisfied: Jinja2>=3.1.2 in /usr/local/lib/python3.9/dist-packages (from Flask==3.0.0->-r requirements.txt (line 1)) (3.1.3)
Requirement already satisfied: itsdangerous>=2.1.2 in /usr/local/lib/python3.9/dist-packages (from Flask==3.0.0->-r requirements.txt (line 1)) (2.1.2)
Requirement already satisfied: click>=8.1.3 in /usr/local/lib/python3.9/dist-packages (from Flask==3.0.0->-r requirements.txt (line 1)) (8.1.7)
Requirement already satisfied: Werkzeug>=3.0.0 in /usr/local/lib/python3.9/dist-packages (from Flask==3.0.0->-r requirements.txt (line 1)) (3.0.1)
Requirement already satisfied: importlib-metadata>=3.6.0 in /usr/local/lib/python3.9/dist-packages (from Flask==3.0.0->-r requirements.txt (line 1)) (7.0.1)
Requirement already satisfied: blinker>=1.6.2 in /usr/local/lib/python3.9/dist-packages (from Flask==3.0.0->-r requirements.txt (line 1)) (1.7.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.9/dist-packages (from importlib-metadata>=3.6.0->Flask==3.0.0->-r requirements.txt (line 1)) (3.17
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-packages (from Jinja2>=3.1.2->Flask==3.0.0->-r requirements.txt (line 1)) (2.1.5)
Installing collected packages: Flask
  WARNING: The script flask is installed in '/home/mrhimanshu004/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed Flask-3.0.0
mrhimanshu004@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_world (him1234)$
```

Step 5: Create & deploy app

> pyhton main.py

```
(him1234)$ python main.py
```

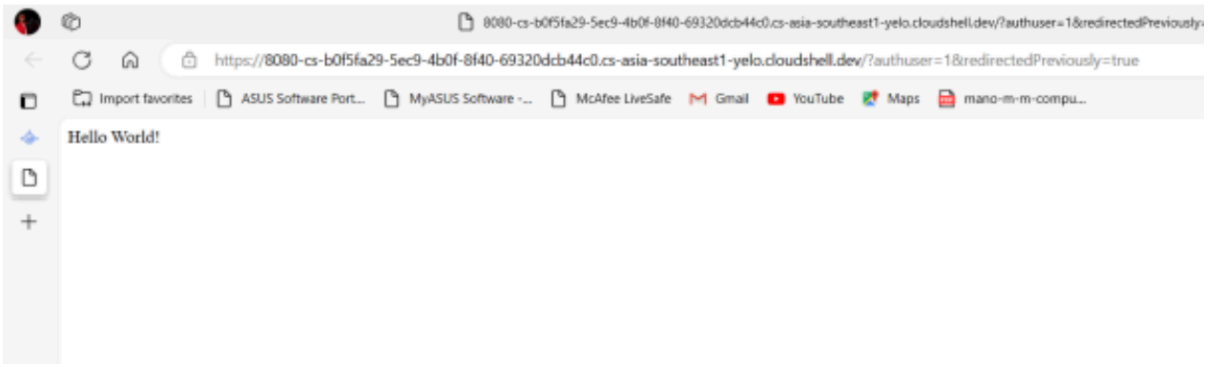
```
mrhimanshu004@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_world (him1234)$ python main.py
 * Serving Flask app 'main'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead
 * Running on http://127.0.0.1:8080
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 112-730-169
gcloud app create
gcloud app deploy app.yaml\
--project him1234
127.0.0.1 - - [29/Feb/2024 06:23:48] "GET /?authuser=1&redirectedPreviously=true HTTP/1.1" 200 -
127.0.0.1 - - [29/Feb/2024 06:23:48] "GET /favicon.ico HTTP/1.1" 404 -
```

Step 6: Click preview on port 8080

>gcloud app create

>gcloud app deploy app.ymal --project

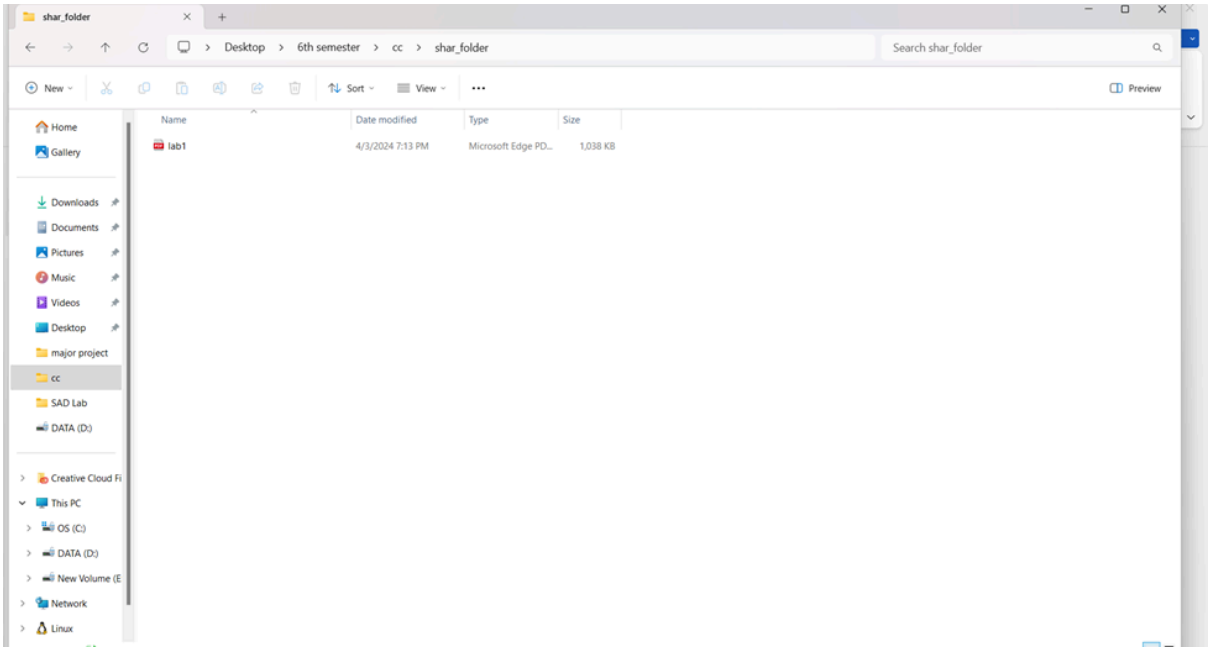
```
mrhimanshu004@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_world (him1234)$ python main.py
 * Serving Flask app 'main'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead
 * Running on http://127.0.0.1:8080
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 112-730-169
gcloud app create
gcloud app deploy app.yaml\
--project him1234
127.0.0.1 - - [29/Feb/2024 06:23:48] "GET /?authuser=1&redirectedPreviously=true HTTP/1.1" 200 -
127.0.0.1 - - [29/Feb/2024 06:23:48] "GET /favicon.ico HTTP/1.1" 404 -
```



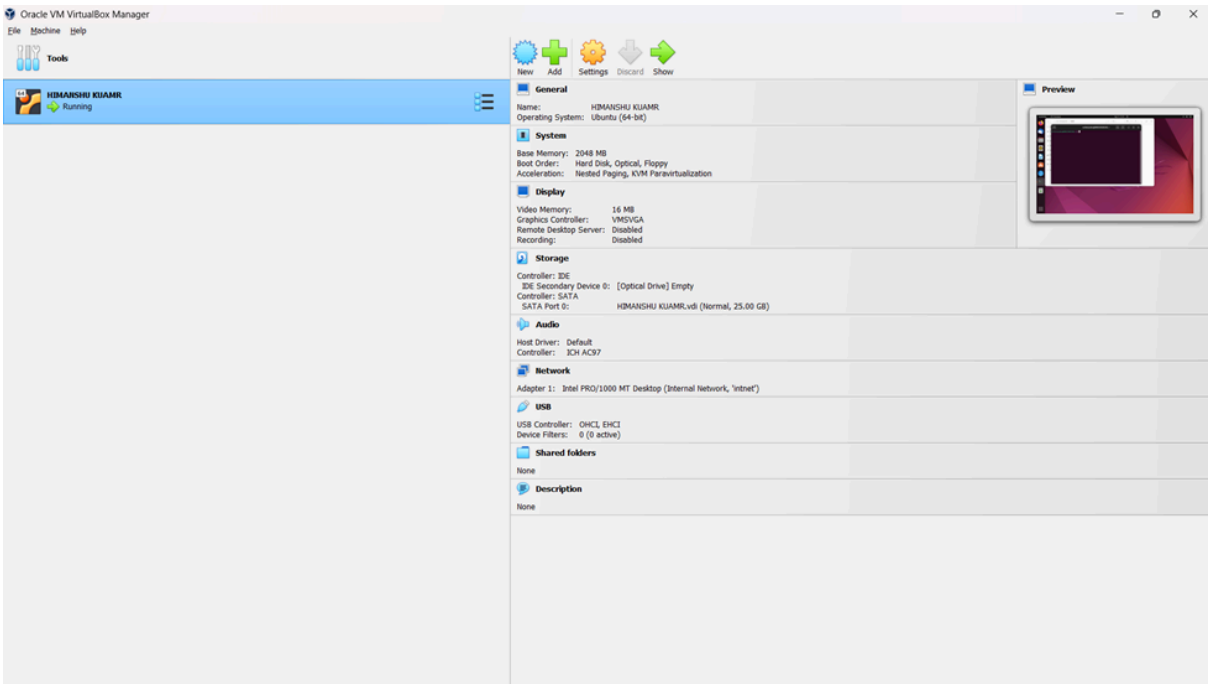
Practical - 5

Aim: **- Write the procedure to transfer file from a virtual machine to other machine**

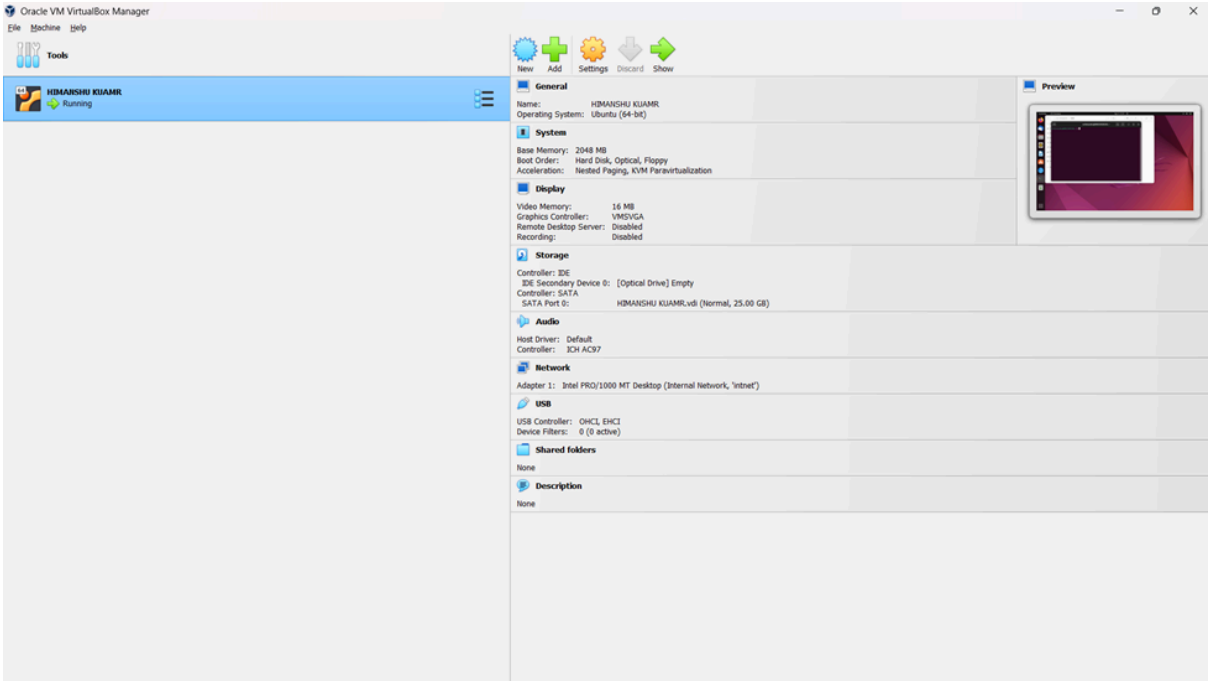
Step 1: Put files that are to be shared in Separate folder (on host machine)



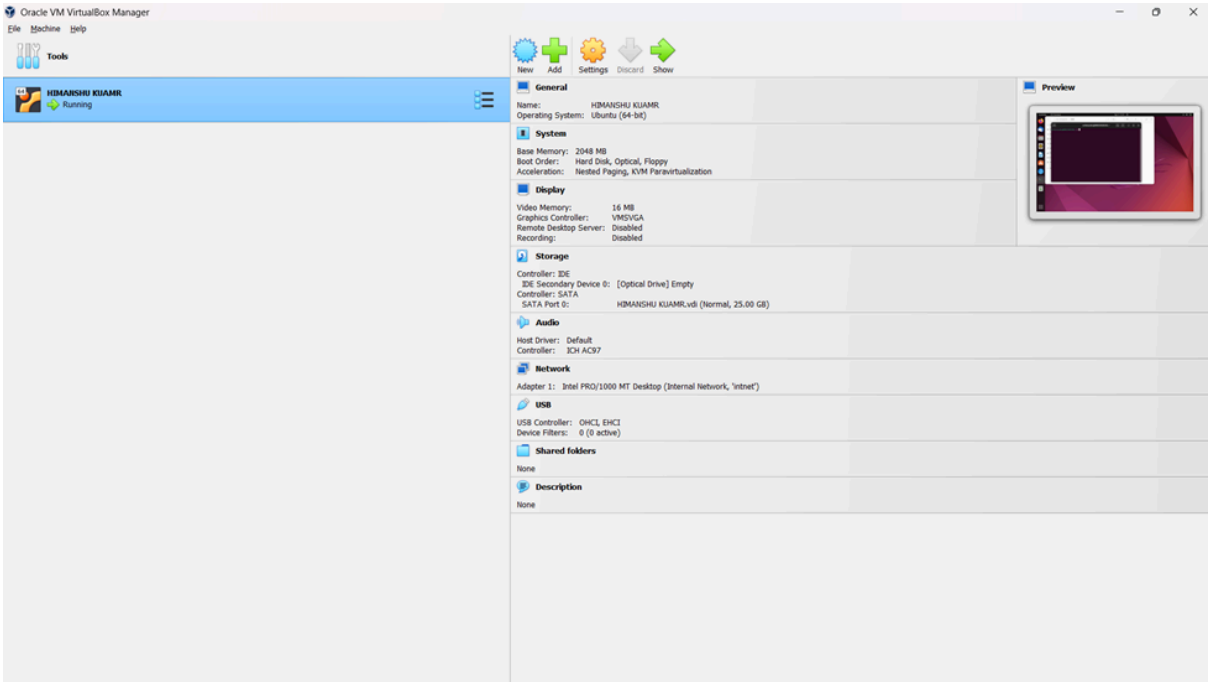
Step 2: Open VM launch Ubuntu



Step 3 : open file manager – go to “other locations”



Step 4 : open folder name “mnt”



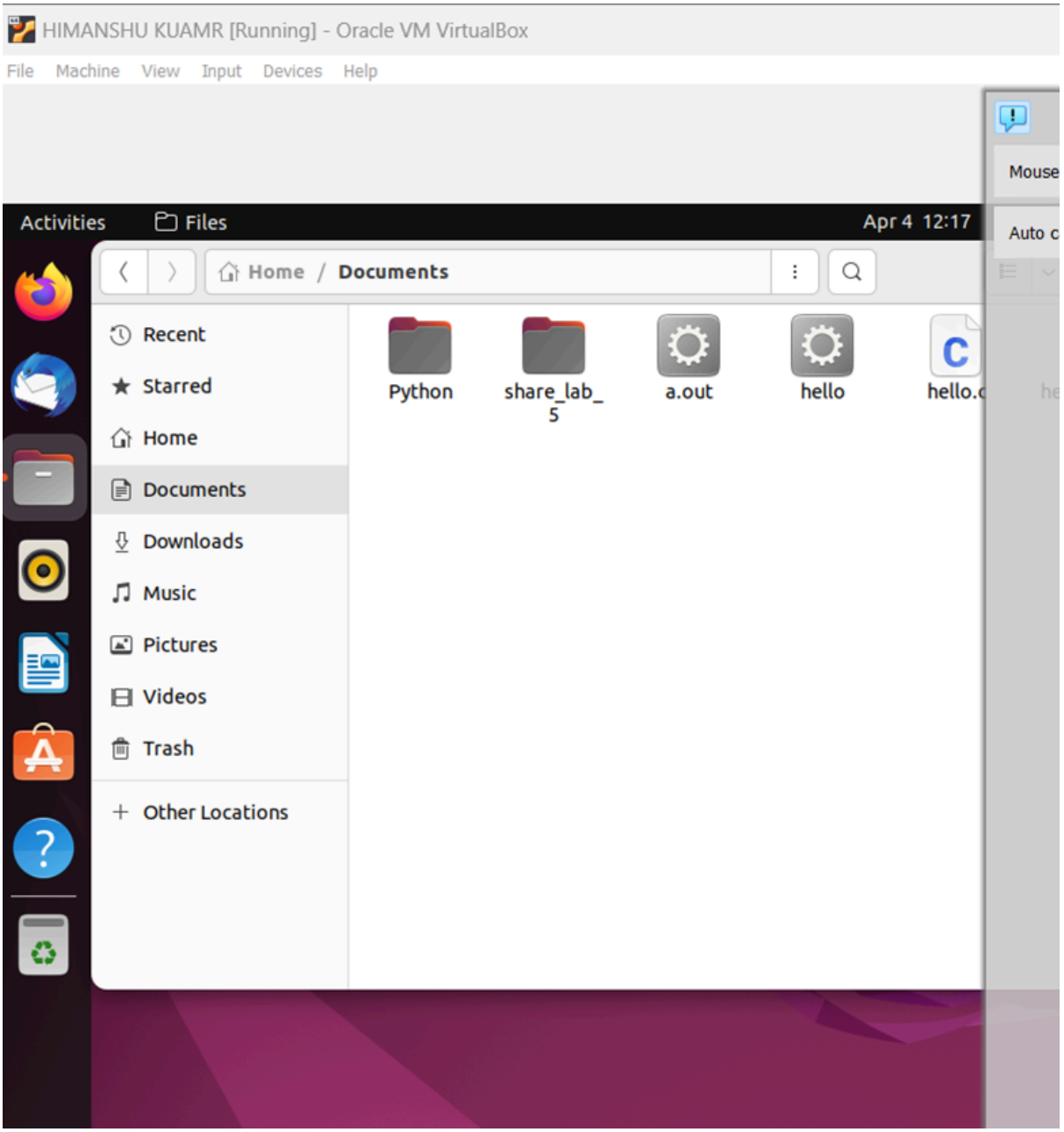
Step 5 : open folder “hgfs”.

Step 6 : Right click VM name & select properties , go to options lab

Step 7 : Select shared folders, change “Always enabled “

Step 8 : Select folder/file that has to be shared . sudo mount -t vboXsf [share name] (path to /mnt/hgfs) ~/../..mnt/hgfs

Step 9 : Check Read only & finish .



Practical - 6

Aim: **- Case Study of Sample cloud Services**

What is AWS:

Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud, offering over 200 fully featured services from data centers globally. Millions of customers—including the fastest-growing startups, largest enterprises, and leading government agencies—are using AWS to lower costs, become more agile, and innovate faster.

History of AWS:

1. Inception (2002-2006):

- AWS began as an internal project at Amazon.com to improve the company's infrastructure and IT operations.
- In 2002, Amazon's CEO, Jeff Bezos, mandated that the company's teams should begin exposing its functionalities as services.
- The initial services included Simple Queue Service (SQS) for messaging and Simple Storage Service (S3) for storage.

2. Launch of AWS (2006-2008):

- AWS was officially launched to the public in 2006, offering cloud computing services to external customers.
- Amazon Elastic Compute Cloud (EC2) was introduced, allowing users to rent virtual computers on which to run their own applications.
- Other services like Amazon Simple DB (a NoSQL database service) and Amazon CloudFront (a content delivery network) were also introduced during this period.

3. Rapid Expansion and Innovation (2009-2013):

- AWS experienced rapid growth and continued to expand its service offerings.
- New services such as Amazon Relational Database Service (RDS) for managed relational databases and Amazon Virtual Private Cloud (VPC) for isolated cloud resources were introduced.
- AWS launched regions and availability zones globally, expanding its infrastructure to multiple geographic locations to better serve customers worldwide.
- The launch of AWS Marketplace in 2012 allowed third-party vendors to sell their software and services directly to AWS customers.

4. Enterprise Adoption and Dominance (2014-2018):

- AWS became increasingly popular among enterprises, startups, and developers due to its reliability, scalability, and cost-effectiveness.
- The introduction of services like AWS Lambda (serverless computing) and Amazon Aurora (a fully managed relational database) further solidified AWS's position as a leading cloud provider.
- AWS continued to innovate with machine learning and artificial intelligence services such as Amazon Sage Maker and Amazon Rekognition.
- Major customers like Netflix, Airbnb, and NASA relied on AWS for their cloud infrastructure needs.

5. Continued Growth and Expansion (2019-2022):

- AWS continued to grow rapidly, maintaining its position as the market leader in cloud computing.
- The launch of services like AWS Outposts (hybrid cloud solution) and AWS Wavelength (edge computing) demonstrated AWS's commitment to meeting diverse customer requirements.

- AWS expanded its presence in industries such as healthcare, finance, and government, offering specialized solutions and compliance certifications.
- In 2021, AWS introduced Graviton2, its custom-designed ARM-based processor, aimed at providing cost-effective and high-performance computing for various workloads.

6. Recent Developments (2022-2024):

- Up until my last update in January 2022, AWS continued its trajectory of growth and innovation. However, specific developments during this period would require more current information beyond my last training data.

Application of AWS services

1. Web Hosting and Content Delivery: AWS offers EC2 for hosting, S3 for scalable storage, and CloudFront for CDN services, ensuring efficient content delivery worldwide.

2. Data Storage and Management: AWS provides S3 for object storage, EBS for block storage, and Glacier for long-term archival, catering to various data storage needs with scalability and cost-effectiveness.

3. Big Data Analytics: AWS's EMR, Redshift, and Athena handle big data processing and analytics, empowering organizations with real-time insights and decision-making capabilities.

4. Machine Learning and AI: AWS offers SageMaker, Rekognition, and Comprehend for building, training, and deploying ML models, analyzing images/videos, and automating tasks for enhanced productivity.

5. Application Development and Deployment: AWS's Lambda, Elastic Beanstalk, and CodePipeline streamline application development and deployment with scalability, automation, and cost optimization features.

6. IoT: AWS IoT services like IoT Core and Greengrass facilitate secure IoT device connectivity, management, and analysis, benefiting industries like manufacturing, healthcare, and smart cities.

7. Enterprise IT Infrastructure: AWS supports enterprises with virtual servers, storage, networking, security, and management tools, enabling modernization, agility, scalability, and cost-efficiency.

Set of Product of AWS:

Compute

1. Amazon EC2 (Elastic Compute Cloud): Virtual servers in the cloud for running applications.
2. AWS Lambda: Serverless compute service for running code without provisioning or managing servers.
3. Amazon ECS (Elastic Container Service): Container management service for running Docker containers.
4. AWS Batch: Fully managed batch processing at any scale.
5. AWS Elastic Beanstalk: Easy-to-use service for deploying and scaling web applications and services.

Storage

1. Amazon S3 (Simple Storage Service): Object storage for storing and retrieving any amount of data.
2. Amazon EBS (Elastic Block Store): Block storage volumes for EC2 instances.
3. Amazon Glacier: Low-cost storage for long-term archiving and backup.
4. AWS Storage Gateway: Hybrid storage service that connects on-premises environments with AWS storage services.

Database

1. Amazon RDS (Relational Database Service): Managed relational database service for MySQL, PostgreSQL, Oracle, SQL Server, and MariaDB.
2. Amazon DynamoDB: Fully managed NoSQL database service.
3. Amazon Redshift: Fully managed data warehouse for analytics.
4. Amazon DocumentDB: Fully managed document database service compatible with MongoDB.

Networking

1. Amazon VPC (Virtual Private Cloud): Isolated virtual network for launching AWS resources.
2. Amazon Route 53: Scalable domain name system (DNS) web service.
3. AWS Direct Connect: Dedicated network connection between on-premises and AWS.

Machine Learning and AI

1. Amazon SageMaker: Fully managed service for building, training, and deploying machine learning models.
2. Amazon Rekognition: Deep learning-based image and video analysis.
3. Amazon Comprehend: Natural language processing service for extracting insights and relationships from text.
4. AWS DeepLens: Deep learning-enabled video camera for developers.

Analytics

1. Amazon EMR (Elastic MapReduce): Managed big data processing service using Apache Hadoop and Spark.
2. Amazon Athena: Serverless interactive query service for analyzing data in S3 using standard SQL.
3. Amazon Kinesis: Real-time data streaming and processing service.

Management and Governance

1. AWS Management Console: Web-based interface for managing AWS services.
2. AWS CloudFormation: Infrastructure as code service for automating resource provisioning.
3. AWS Config: Service for assessing, auditing, and evaluating AWS resource configurations.
4. AWS Organizations: Policy-based management for multiple AWS accounts.

Security and Identity

1. AWS IAM (Identity and Access Management): Service for managing user access and permissions.
2. Amazon Inspector: Automated security assessment service.
3. AWS Key Management Service (KMS): Managed service for creating and controlling encryption keys.

IoT (Internet of Things)

1. AWS IoT Core: Managed cloud service for connecting IoT devices to the cloud.
2. AWS IoT Greengrass: Software that extends AWS IoT functionality to devices.
3. Amazon FreeRTOS: Real-time operating system for microcontrollers.

Developer Tools

1. AWS CodeDeploy: Automated code deployment service.
2. AWS CodePipeline: Continuous integration and continuous delivery (CI/CD) service.
3. AWS CodeCommit: Fully managed source control service.

Pay as-you Go feature in AWS;

1. Usage-Based Billing: With pay-as-you-go pricing, customers are charged based on their actual usage of AWS services, such as compute instances, storage, databases, network traffic, and other resources. AWS tracks usage metrics for each service, and customers are billed accordingly on a per-hour, per-minute, or per-request basis, depending on the service.
2. No Upfront Costs: Unlike traditional IT infrastructure, where customers have to make upfront investments in hardware, software licenses, and maintenance, AWS does not require any upfront costs. Customers can get started with AWS services immediately without any initial investment, reducing financial barriers to entry.
3. Scalability and Flexibility: Pay-as-you-go pricing allows customers to scale their usage of AWS services up or down dynamically based on demand. Customers can provision additional resources when needed to handle spikes in traffic or workload, and scale down when demand decreases, optimizing costs and resource utilization.
4. Cost Control and Management: AWS provides tools and features to help customers monitor, analyze, and optimize their usage and costs. Customers can use services like AWS Cost Explorer, AWS Budgets, and AWS Trusted Advisor to track spending, set budget limits, identify cost-saving opportunities, and optimize resource usage to minimize costs.
5. No Long-Term Commitments: With pay-as-you-go pricing, customers are not locked into long-term contracts or commitments. They have the flexibility to use AWS services on a month-to-month basis and can stop or modify their usage at any time without penalties or termination fees.
6. Elasticity and On-Demand Provisioning: AWS services are designed to be elastic and on-demand, allowing customers to quickly provision and de-provision resources as needed. This elasticity enables customers to respond rapidly to changing business requirements and scale their infrastructure in real-time to meet demand fluctuations.

Advantage and Disadvantage of AWS

Advantages:

1. Scalability: AWS provides scalable resources that can grow or shrink based on demand. This flexibility allows businesses to efficiently handle fluctuating workloads without over-provisioning or under-provisioning resources.
2. Cost-Effective: AWS operates on a pay-as-you-go pricing model, allowing businesses to pay only for the resources they consume. This can significantly reduce upfront costs and minimize wasted resources.

3. Global Infrastructure: AWS has a vast global infrastructure with data centers located in multiple regions worldwide. This enables businesses to deploy their applications closer to their users, reducing latency and improving performance.
4. Security: AWS offers robust security measures to protect data, including encryption, identity and access management, and compliance certifications. Additionally, AWS provides tools and services to help businesses secure their applications and infrastructure.
5. Reliability: AWS provides high availability and reliability through redundancy and fault-tolerant architectures. Services like Amazon S3 and Amazon RDS are designed to offer durability and availability even in the face of hardware failures.
6. Flexibility: AWS offers a wide range of services spanning computing, storage, databases, machine learning, analytics, and more. This allows businesses to build and deploy diverse applications with ease.

Disadvantages:

1. Complexity: The breadth and depth of AWS services can be overwhelming, especially for beginners. Managing and optimizing these services require expertise, which may pose a challenge for some organizations.
2. Vendor Lock-In: Adopting AWS services may result in vendor lock-in, making it difficult to migrate to other cloud providers in the future. This can limit flexibility and increase dependency on AWS.
3. Cost Management: While AWS offers cost-effective pricing, managing costs can be complex, especially as usage scales up. Without proper monitoring and optimization, costs can quickly escalate.
4. Performance Variability: Despite AWS's global infrastructure, performance can vary based on factors such as geographical location, network congestion,

---

Account Creation of AWS: