

Unit-3

Python for Amazon Web Services:

Python is a programming language that is widely used in web applications, software development, data science, and machine learning (ML).

Developers use Python because it is efficient and easy to learn and can run on many different platforms.

Python software is free to download, integrates well with all types of systems, and increases development speed.

BENEFITS OF PYTHON

Benefits of Python include:

- Developers can easily read and understand a Python program because it has basic, English-like syntax.
- Python makes developers more productive because they can write a Python program using fewer lines of code compared to many other languages.
- Python has a large standard library that contains reusable codes for almost any task. As a result, developers do not have to write code from scratch.
- Developers can easily use Python with other popular programming languages such as Java, C, and C++.
- The active Python community includes millions of supportive developers around the globe. If you face an issue, you can get quick support from the community.
- Plenty of helpful resources are available on the internet if you want to learn Python. For example, you can easily find videos, tutorials, documentation, and developer guides.
- Python is portable across different computer operating systems such as Windows, macOS, Linux, and Unix.

PYTHON USED

The Python language has several use cases in application development, including the following examples:

SERVER-SIDE WEB DEVELOPMENT

Server-side web development includes the complex backend functions that websites perform to display information to the user. For example, websites must interact with databases, talk to other websites, and protect data when sending it over the network.

Python is useful for writing server-side code because it offers many libraries that consist of prewritten code for complex backend functions. Developers also use a wide range of Python frameworks that provide all the necessary tools to build web applications faster and more easily. For example, developers can create the skeleton web application in seconds because they don't need to write it from scratch. They can then test it using the framework's testing tools, without depending on external testing tools.

AUTOMATION WITH PYTHON SCRIPTS

A scripting language is a programming language that automates tasks that humans normally perform. Programmers widely use Python scripts to automate many day-to-day tasks such as the following:

- Renaming a large number of files at once
- Converting a file to another file type
- Removing duplicate words in a text file
- Performing basic mathematical operations
- Sending email messages
- Downloading content
- Performing basic log analysis
- Finding errors in multiple files

DATA SCIENCE AND MACHINE LEARNING

[Data science](#) is extracting valuable knowledge from data, and [machine learning \(ML\)](#) teaches computers to automatically learn from the data and make accurate predictions. Data scientists use Python for data science tasks such as the following:

- Fixing and removing incorrect data, which is known as data cleaning
- Extracting and selecting various features of data
- [Data labeling](#), which is adding meaningful names for the data
- Finding different statistics from data
- Visualizing data by using charts and graphs such as line charts, bar graphs, histograms, and pie charts

Data scientists use Python ML libraries to train ML models and build classifiers that accurately classify data. People in different fields use Python-based classifiers to do classification tasks such as image, text, and network traffic

classification; speech recognition; and facial recognition. Data scientists also use Python for deep learning, an advanced ML technique.

SOFTWARE DEVELOPMENT

Software developers often use Python for different development tasks and software applications such as the following:

- Keeping track of bugs in the software code
- Automatically building the software
- Handling software project management
- Developing software prototypes
- Developing desktop applications using Graphical User Interface (GUI) libraries
- Developing simple text-based games to more complex video games

SOFTWARE TEST AUTOMATION

Software testing is the process of checking whether the actual results from the software match the expected results to ensure that the software is error-free.

- Developers use Python unit test frameworks, such as unittest, Robot, and PyUnit, to test the functions they write.
- Software testers use Python to write test cases for various test scenarios. For example, they use it to test the user interface of a web application, multiple software components, and new features.

Developers can use several tools to automatically run test scripts. These tools are known as Continuous Integration/Continuous Deployment (CI/CD) tools. Software testers and developers use CI/CD tools such as Travis CI and Jenkins to automate tests. The CI/CD tool automatically runs the Python test scripts and reports the test results whenever developers introduce new code changes.

HISTORY OF PYTHON

Guido Van Rossum, a computer programmer in the Netherlands, created Python. He started it in 1989 at Centrum Wiskunde & Informatica (CWI), initially as a hobby project to stay busy during Christmastime. The name for the language was inspired by the BBC TV show Monty Python's Flying Circus because Guido Van Rossum was a big fan of the show.

HISTORY OF PYTHON RELEASES

- Guido Van Rossum published the first version of the Python code (version 0.9.0) in 1991. It already included good features such as some data types and functions for error handling.
- Python 1.0 was released in 1994 with new functions to easily process a list of data, such as map, filter, and reduce.
- Python 2.0 was released on October 16, 2000, with new useful features for programmers, such as support for Unicode characters and a shorter way to loop through a list.
- On December 3, 2008, Python 3.0 was released. It included features such as the print function and more support for number division and error handling.

FEATURES OF PYTHON

Following features of the Python programming language make it unique:

AN INTERPRETED LANGUAGE

Python is an interpreted language, which means it directly runs the code line by line. If there are errors in the program code, it will stop running. Therefore, programmers can quickly find errors in the code.

AN EASY-TO-USE LANGUAGE

Python uses English-like words. Unlike other programming languages, Python doesn't use curly brackets. Instead, it uses indentation.

A DYNAMICALLY TYPED LANGUAGE

Programmers do not have to declare variable types when writing code because Python determines them at runtime. Because of this, you can write Python programs more quickly.

A HIGH-LEVEL LANGUAGE

Python is closer to human languages than some other programming languages. Therefore, programmers do not have to worry about its underlying functionalities such as architecture and memory management.

AN OBJECT-ORIENTED LANGUAGE

Python considers everything to be an object, but it also supports other types of programming such as structured and functional programming.

PYTHON LIBRARIES

A library is a collection of frequently used codes that developers can include in their Python programs to avoid writing code from scratch. By default, Python comes with the Standard Library, which contains a lot of reusable functions. In addition, more than 137,000 Python libraries are available for various applications, including web development, data science, and machine learning (ML).

POPULAR PYTHON LIBRARIES

Matplotlib

Developers use Matplotlib to plot data in high-quality two- and three-dimensional (2D and 3D) graphics. It is often used in scientific applications. With Matplotlib, you can visualize data by displaying it in different charts such as bar charts and line charts. You can also plot multiple charts at once, and the graphics are portable across all platforms.

Pandas

Pandas provides optimized and flexible data structures that you can use to manipulate time-series data and structured data, such as tables and arrays. For example, you can use Pandas to read, write, merge, filter, and group data. Many people use it for data science, data analysis, and ML tasks.

NumPy

NumPy is a popular library that developers use to easily create and manage arrays, manipulate logical shapes, and perform linear algebra operations. NumPy supports integration with many languages such as C and C++.

Requests

The Requests library provides useful functions that are required for web development. You can use it to send HTTP requests, add headers, add URL parameters, add data, and perform many more tasks when communicating with web applications.

OpenCV-Python

OpenCV-Python is a library that developers use to process images for computer vision applications. It provides many functions for image processing tasks such as reading and writing images simultaneously, building a 3D environment from a 2D one, and capturing and analyzing images from video.

Keras

Keras is Python's deep neural network library with excellent support for data processing, visualization and much more. Keras supports many neural networks. It has a modular structure that offers flexibility in writing innovation applications.

PYTHON FRAMEWORKS

A Python framework is a collection of packages and modules. A module is a set of related code, and a package is a set of modules. Developers can use Python frameworks to build Python applications more quickly because they do not have to worry about low-level details such as how communications happen in the web application or how Python will make the program faster. Python has two types of frameworks:

- A full-stack framework includes almost everything that is required to build a large application.
- A microframework is a basic framework that provides minimal functionalities for building simple Python applications. It also provides extensions if applications need more sophisticated functions.

THE MOST POPULAR PYTHON FRAMEWORKS

Developers can use multiple Python frameworks to make their development efficient, including the following frameworks:

Django

Django is one of the most widely used full-stack Python web frameworks for developing large-scale web applications. It provides several useful features, including a web server for development and testing, a template engine to build the website frontend, and various security mechanisms.

Flask

Flask is a micro-framework for developing small web applications. Its features include strong community support, well-written documentation, a template engine, unit testing, and a built-in web server. It also provides extensions for validation support, database mapping layers, and web security.

TurboGears

TurboGears is a framework designed to build web applications faster and easier. These are some of its popular features:

- A specific database table structure
- Tools for creating and managing projects
- A template engine to build the databases
- A template engine to build the frontend
- Mechanisms to handle web security

Apache MXNet

Apache MXNet is a fast, flexible, and scalable deep learning framework that developers use to build research prototypes and deep learning applications. It supports multiple programming languages, including Java, C++, R, and Perl. It provides a rich set of tools and libraries to support development. For example, you can find an interactive machine learning (ML) book, computer vision toolkits, and deep learning models for Natural Language Processing (NLP), which processes natural language such as text and speech.

PyTorch

PyTorch is a framework for ML that has been built on top of the Torch library, which is another open-source ML library. Developers use it for applications such as NLP, robotics, and computer vision, finding meaningful information in images and videos. They also use it to run those applications in CPUs and GPUs.

PYTHON IDEs?

An integrated development environment (IDE) is software that gives developers the tools they need to write, edit, test, and debug code in one place.

THE MOST POPULAR PYTHON IDEs?

PyCharm

JetBrains, a Czech company that develops software tools, created PyCharm. It has a free community edition that is suitable for small Python applications and a paid professional edition that is suitable for building large-scale Python applications, with the following full set of features:

- Automatic code completion and code inspection
- Error handling and quick fixes
- Code cleaning without changing the functionality
- Support for web application frameworks such as Django and Flask
- Support for other programming languages, such as JavaScript, CoffeeScript, TypeScript, AngularJS, and Node
- Scientific tools and libraries such as Matplotlib and NumPy
- Ability to run, debug, test, and deploy applications in remote virtual machines
- A debugger to find errors in the code, a profiler to identify performance issues in the code, and a test runner for running unit tests
- Support for databases

IDLE

Integrated Development and Learning Environment (IDLE) is the Python IDE installed by default. It has been developed only with Python using the Tkinter GUI toolkit and offers the following features:

- Works across many operating systems such as Windows, Unix, and macOS
- Provides a shell window to run commands and display the output
- Offers a multiple-window text editor that provides code syntax highlighting and automatic code completion
- Has its own debugger

Spyder

Spyder is an open-source IDE that many scientists and data analysts use. It provides a comprehensive development experience with features for advanced data analysis, data visualization, and debugging. It also includes the following features:

- A rich code editor that supports multiple languages
- An interactive IPython console
- A basic debugger
- Scientific libraries such as Matplotlib, SciPy, and NumPy
- Ability to explore variables in the code
- Ability to view documentation in real time

Atom

Atom is a free editor developed by GitHub that supports coding in many programming languages, including Python. Using Atom, developers can directly work with GitHub, the website where you can save your code centrally. Atom offers the following features:

- Ability to use with many operating systems
- Easy installation or creation of new packages
- Faster automatic code completion
- Ability to search files and projects
- Easy customization of the interface

PYTHON SDKS

A software development kit (SDK) is a collection of software tools that developers can use to create software applications in a particular language. Most SDKs are specific to different hardware platforms and operating systems. Python SDKs include many tools such as libraries, code samples, and developer guides that developers find helpful when writing applications.

BOTO3 IN PYTHON

Boto3 is the AWS SDK for Python. You can use it to create, configure, and manage AWS services such as [Amazon Elastic Compute Cloud \(EC2\)](#), [Amazon Simple Storage Service \(S3\)](#), and [Amazon DynamoDB](#). Boto3 also provides two types of [APIs](#): low-level APIs and Resource APIs for developers.

AWS PyCHARM

The [AWS Toolkit for PyCharm](#) is the plug-in for the PyCharm IDE that makes it easier to create, debug, and deploy Python applications on AWS. Using the AWS Toolkit for PyCharm, developers can easily get started with Python development. It provides several useful features for developers, including start guides, step-through debugging, and IDE deployment.

PYTHON FOR AWS CLOUD9

Python for Amazon web services :

Python is a programming language that can be used to interact with Amazon Web Services (AWS). The AWS SDK for Python (Boto3) allows you to use Python code to interact with AWS services like: Amazon S3, Amazon EC2, Amazon DynamoDB.

You can use Boto3 to:

- Create an Amazon S3 bucket
- List your available buckets
- Delete a bucket
- Integrate your Python application, library, or script with AWS services

To run a Python script in AWS, you can:

- Create an S3 bucket
- Upload the Python script
- Grant

Python is a popular programming language for interacting with Amazon Web Services (AWS). You can use Python to develop applications and control your cloud-based infrastructure.

You can use the AWS SDK for Python (Boto3) to:

- Create, configure, and manage AWS services
- Create an Amazon S3 bucket
- List your available buckets
- Delete a bucket

You can use Python to interact with AWS services like:

- Amazon Elastic Compute Cloud (EC2)
- Amazon Simple Storage Service (S3)
- Amazon DynamoDB

You can add Python code to AWS by:

- Creating an EC2 instance
- Granting access to the .pem file in the local
- Connecting to the EC2 instance via CMD
- Creating a directory and uploading the Python script
- Installing Python 3 and downloading all the supporting Python packages in the root directory

Python is a leading language for cloud computing because of its:

- Ease of use
- Performance
- Open-source development
- Third-party integrations
- Popularity among developers

How to run Python code in an AWS Cloud9 development environment.

RESULT IN CHARGES TO YOUR AWS ACCOUNT . THESE INCLUDE POSSIBLE CHARGES FOR SERVICES SUCH AS AMAZON ELASTIC COMPUTE CLOUD (AMAZON EC2) AND AMAZON SIMPLE STORAGE SERVICE (AMAZON S3). FOR MORE INFORMATION , SEE [AMAZON EC2 PRICING](#) AND [AMAZON S3 PRICING](#).

PREREQUISITES

Before you use this tutorial, be sure to meet the following requirements.

- **You have an AWS Cloud9 EC2 development environment**
This tutorial assumes that you have an EC2 environment, and that the environment is connected to an Amazon EC2 instance running Amazon Linux or Ubuntu Server. See [Creating an EC2 Environment](#) for details.
If you have a different type of environment or operating system, you might need to adapt this tutorial's instructions.
- **You have opened the AWS Cloud9 IDE for that environment**
When you open an environment, AWS Cloud9 opens the IDE for that environment in your web browser. See [Opening an environment in AWS Cloud9](#) for details.

STEP 1: INSTALL PYTHON

1. In a terminal session in the AWS Cloud9 IDE, confirm whether Python is already installed by running the **python3 --version** command. (To start a new terminal session, on the menu bar choose **Window, New Terminal**.) If Python is installed, skip ahead to [Step 2: Add code](#).
2. Run the **yum update** (for Amazon Linux) or **apt update** (for Ubuntu Server) command to help ensure the latest security updates and bug fixes are installed.
For Amazon Linux:

```
sudo yum -y update
```


For Ubuntu Server:

```
sudo apt update
```
3. Install Python by running the **install** command.
For Amazon Linux:

```
sudo yum -y install python3
```


For Ubuntu Server:

```
sudo apt-get install python3
```

STEP 2: ADD CODE

In the AWS Cloud9 IDE, create a file with the following content and save the file with the name `hello.py`. (To create a file, on the menu bar choose **File, New File**. To save the file, choose **File, Save**.)

STEP 3: RUN THE CODE

1. In the AWS Cloud9 IDE, on the menu bar choose **Run, Run Configurations, New Run Configuration**.
2. On the **[New] - Stopped** tab, enter `hello.py 5 9` for **Command**. In the code, 5 represents `sys.argv[1]`, and 9 represents `sys.argv[2]`.
3. Choose **Run** and compare your output.
4. Hello, World!
5. The sum of 2 and 3 is 5.
The sum of 5 and 9 is 14.

- By default, AWS Cloud9 automatically selects a runner for your code. To change the runner, choose **Runner**, and then choose **Python 2** or **Python 3**.
-

STEP 4: INSTALL AND CONFIGURE THE AWS SDK FOR PYTHON (BOTO3)

The AWS SDK for Python (Boto3) enables you to use Python code to interact with AWS services like Amazon S3. For example, you can use the SDK to create an Amazon S3 bucket, list your available buckets, and then delete the bucket you just created.

INSTALL PIP

In the AWS Cloud9 IDE, confirm whether pip is already installed for the active version of Python by running the **python -m pip --version** command. If pip is installed, skip to the next section.

To install pip, run the following commands. Because sudo is in a different environment from your user, you must specify the version of Python to use if it differs from the current aliased version.

INSTALL THE AWS SDK FOR PYTHON (BOTO3)

After you install pip, install the AWS SDK for Python (Boto3) by running the **pip install** command.

SET UP CREDENTIALS IN YOUR ENVIRONMENT

Each time you use the AWS SDK for Python (Boto3) to call an AWS service, you must provide a set of credentials with the call. These credentials determine whether the SDK has the necessary permissions to make the call. If the credentials don't cover the necessary permissions, the call fails.

To store your credentials within the environment, follow the instructions in [Calling AWS services from an environment in AWS Cloud9](#), and then return to this topic.

For additional information, see [Credentials](#) in the AWS SDK for Python (Boto3).

STEP 5: ADD AWS SDK CODE

Add code that uses Amazon S3 to create a bucket, list your available buckets, and optionally delete the bucket you just created.

In the AWS Cloud9 IDE, create a file with the following content and save the file with the name `s3.py`.

STEP 6: RUN THE AWS SDK CODE

- On the menu bar, choose **Run, Run Configurations, New Run Configuration**.
 - For **Command**, enter `s3.py my-test-bucket us-west-2`, where `my-test-bucket` is the name of the bucket to create, and `us-west-2` is the ID of the AWS Region where your bucket is created. By default, your bucket is deleted before the script exits. To keep your bucket, add `--keep_bucket` to your command. For a list of AWS Region IDs, see [Amazon Simple Storage Service Endpoints and Quotas](#) in the *AWS General Reference*.
Note: Amazon S3 bucket names must be unique across AWS—not just your AWS account.
 - Choose **Run**, and compare your output.
-

STEP 7: CLEAN UP

To prevent ongoing charges to your AWS account after you're done with this tutorial, delete the AWS Cloud9 environment. For instructions, see [Deleting an environment in AWS Cloud9](#).

AUTOMATE PYTHON FLASK DEPLOYMENT TO THE AWS CLOUD

introducing additional AWS services that demonstrate how to manage and deploy an application in the AWS Cloud. The updated workflow includes the following additions: Continuous Integration / Continuous Deployment (CI / CD) pipeline, applications tests, Blue / Green application deployment pattern, and updated Terraform modules. This additional functionality enables developers to automatically test and deploy new versions of the Flask application. Using [Python Flask](#), [Terraform](#), and [Docker](#) in conjunction with several AWS services, you configure a CI/CD pipeline for deploying a Python Flask application.

After going through this exercise, readers will know how to use AWS services and open source tools for managing an open source application in the AWS Cloud.

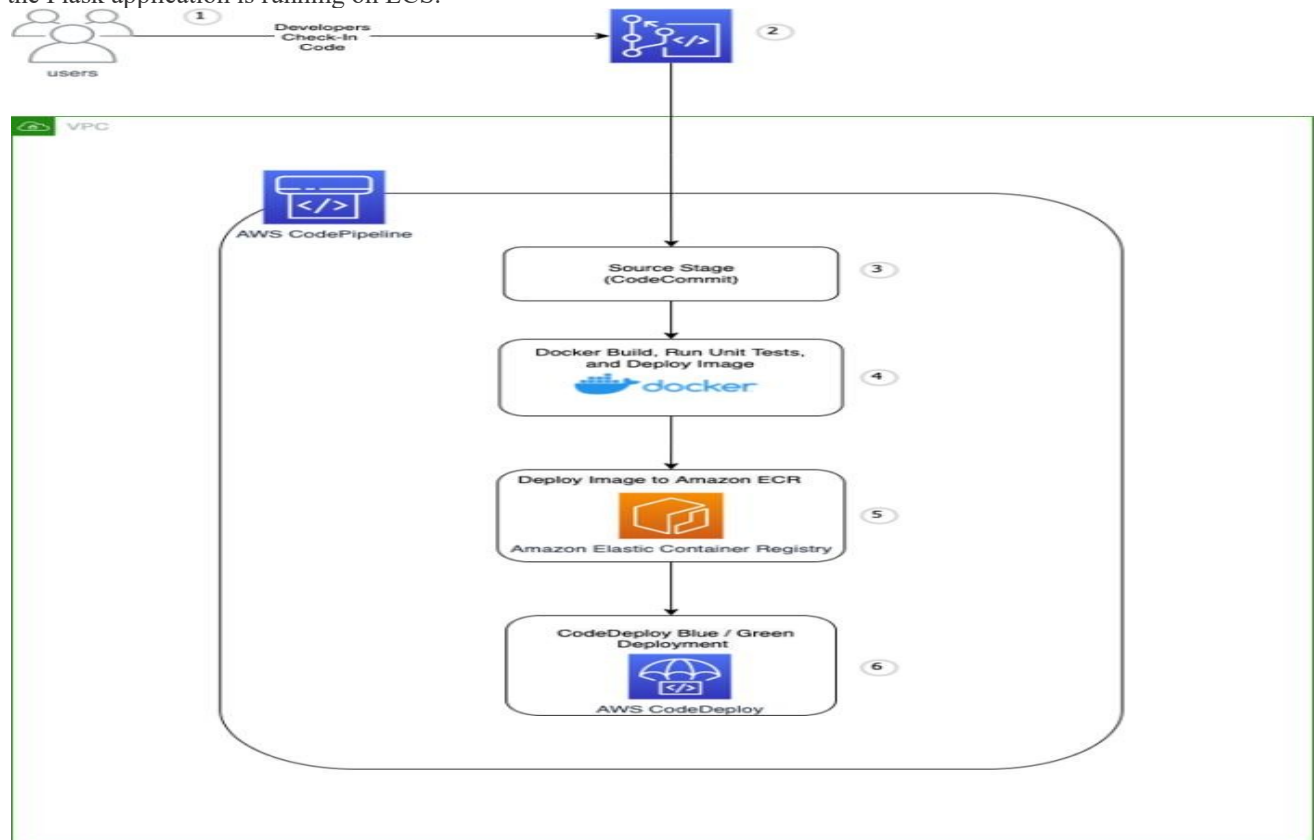
PREREQUISITES

- An AWS account with local credentials properly configured (typically under `~/.aws/credentials`).
- The latest version of the [AWS Command Line Interface \(AWS CLI\)](#). For more information, refer to the documentation for [installing, updating, and uninstalling the AWS CLI](#).
- [Terraform](#) 14.7+ installed on local workstation.
- [Docker](#) Desktop 3.1.0+ installed on local workstation.

- A [Git client](#) to clone the source code provided and a GitHub repository.
- [Python6+](#) installed on local workstation.
- [Boto3](#) installed on local workstation.

SOLUTION OVERVIEW

There is a sample Python Flask application that is deployed by [AWS CodePipeline](#). The pipeline uses [Docker](#) to build and deploy the image to the [Amazon Elastic Container Registry \(Amazon ECR\)](#). Unit tests are run during the Docker image build. Next, the [appspec](#) file is updated to point to the latest [ECS task Definition](#). [AWS CodeDeploy](#) kicks off a [Blue / Green deployment](#) and deploys the latest Docker image to Elastic Container Service running on EC2 instances. Once traffic is shifted to the new version of the application, CodeDeploy completes successfully and the latest version of the Flask application is running on ECS.



The pipeline deploys the Flask application:

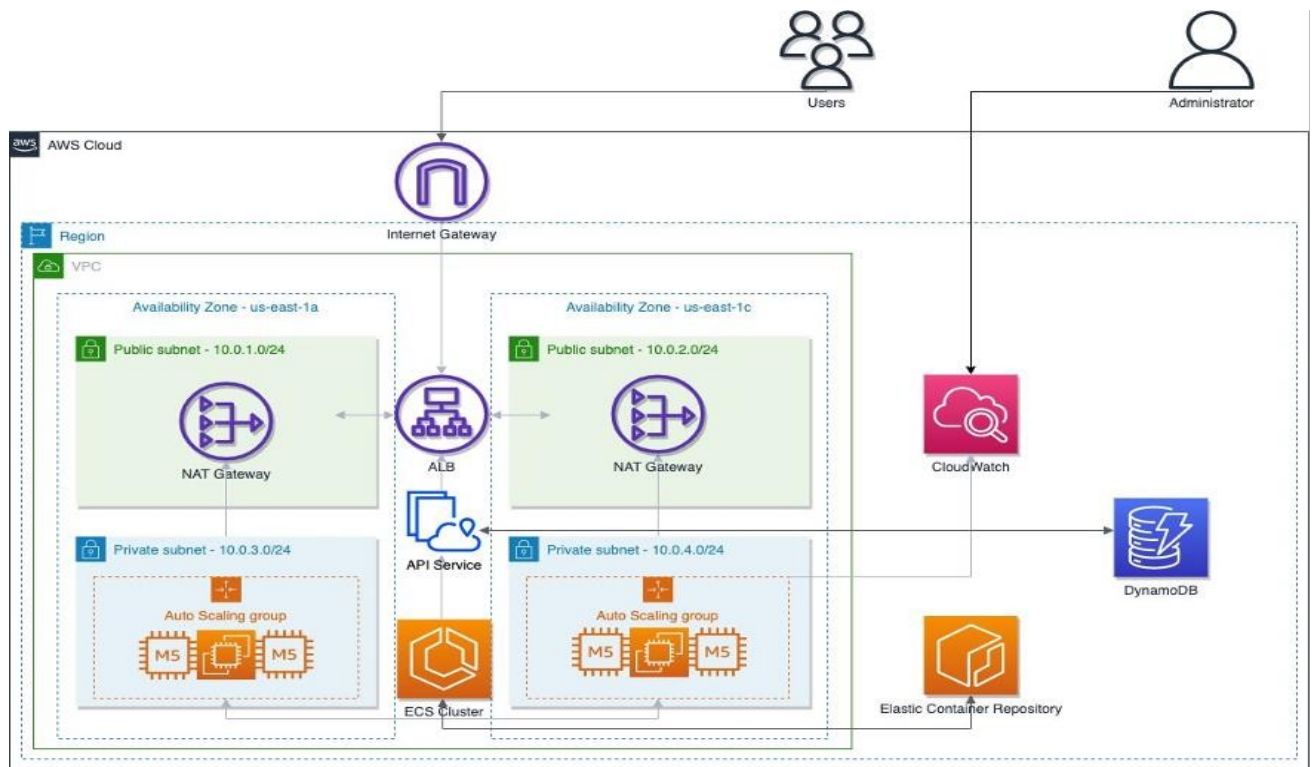
1. The developer pushes code to the main branch.
2. An update to the main branch invokes the pipeline.
3. The pipeline clones the AWS CodeCommit repository.
4. Docker builds the container image, runs unit tests, and assigns tags.
5. Docker pushes the image to Amazon ECR.
6. CodeDeploy Blue / Green deployment with the latest version of Flask app deployed to ECS

APPLICATION OVERVIEW

Python Flask is the foundation for the rest API. Python Flask is a micro framework for building web applications. The Flask application has a back end database of Amazon DynamoDB. The Flask API defines two routes. The first route maps to `/`, and the second maps to `/v1/bestmusic/90s/artist`. These are the only paths recognized by the application. If you enter any other URL when accessing the API, you will receive an error message. You can define specific error responses in the API routes. For example, referencing the Python functions in the `get_artist` method, “*Artist does not exist*” is the response returned when a users requests an artist that is not present in the DynamoDB table (musicTable).

The `create_artist` method posts an artist and song to your DynamoDB.

Unit and functional tests are also included as part of this workflow. [PyUnit](#) is used for the unit tests. The below diagram depicts the application architecture.



You now configure and deploy the AWS services:

1. Clone the demo code:

```
git clone https://github.com/aws-samples/deploy-python-flask-microservices-to-aws-using-open-source-tools-part2
```

2. Change directory into the *terraform/* directory:

```
cd terraform/
```

3. Create the CodeCommit repository:

```
make configure-repo
```

You should see output similar to the following:

```
INFO: Creating CodeCommit repository.
```

```
{ repositoryMetadata: { accountId: 01234567890, repositoryId: 1b807cbf-c184-4e9c-823b-3262973b39cc,
  repositoryName: FlaskDemoRepo-01234567890, repositoryDescription: Flask demo application repo.,
  lastModifiedDate: 2022-05-12T11:40:09.129000-04:00, creationDate: 2022-05-12T11:40:09.129000-04:00,
  cloneUrlHttp: https://git-codecommit.us-east-1.amazonaws.com/v1/repos/FlaskDemoRepo-01234567890,
  cloneUrlSsh: ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/FlaskDemoRepo-01234567890, Arn:
  arn:aws:codecommit:us-east-1:1234567890:FlaskDemoRepo-01234567890 } }
```

```
INFO: Successfully created CodeCommit repository.
```

```
INFO: The CodeCommit HTTP clone URL is https://git-codecommit.us-east-1.amazonaws.com/v1/repos/FlaskDemoRepo-01234567890
```

4. Clone the AWS CodeCommit repository, you may be prompted for your CodeCommit username / password:

```
make clone
```

The output should look like the following:

```
Cloning into 'FlaskDemoRepo-01234567890'...
```

```
warning: You appear to have cloned an empty repository.
```

5. Configure the CodeCommit repository:

```
make upload-codecommit
```

The output should look like the following:

```
cd /home/ssm-user/private-rest-api/terraform/FlaskDemoRepo-01234567890 && \
```

```
git checkout -b main && \
```

```
cd /home/ssm-user/private-rest-api && \
```

```
tar czf demo-code.tar.gz app Dockerfile appspec.yml buildspec.yml && \
```

```
tar -tvf demo-code.tar.gz && \
```

```
mv /home/ssm-user/private-rest-api/demo-code.tar.gz /home/ssm-user/private-rest-api/terraform/FlaskDemoRepo-01234567890 && \
```

```
cd /home/ssm-user/private-rest-api/terraform/FlaskDemoRepo-01234567890 && \
```

```

tar -xvf /home/ssm-user/private-rest-api/terraform/FlaskDemoRepo-01234567890/demo-code.tar.gz && \
rm /home/ssm-user/private-rest-api/terraform/FlaskDemoRepo-01234567890/demo-code.tar.gz && \
git add . && \
git commit -m "Configuring repo." && \
git push -u origin main
Switched to a new branch 'main'
drwxr-xr-x ssm-user/ssm-user 0 2022-07-13 00:47 app/
drwxr-xr-x ssm-user/ssm-user 0 2022-07-13 00:47 app/tests/
drwxr-xr-x ssm-user/ssm-user 0 2022-07-12 19:25 app/tests/lambda-tests/
-rw-r--r-- ssm-user/ssm-user 1970 2022-07-12 19:25 app/tests/lambda-tests/functional_test.py
-rw-r--r-- ssm-user/ssm-user 918 2022-07-12 19:25 app/tests/lambda-tests/functional_test.zip
-rw-r--r-- ssm-user/ssm-user 194 2022-07-13 00:47 app/tests/test.py
-rw-r--r-- ssm-user/ssm-user 1319 2022-07-13 00:47 app/app.py
-rw-r--r-- ssm-user/ssm-user 399 2022-07-12 19:25 Dockerfile
-rw-r--r-- ssm-user/ssm-user 291 2022-07-12 19:25 appspec.yaml
-rw-r--r-- ssm-user/ssm-user 957 2022-07-12 19:25 buildspec.yml
app/
app/tests/
app/tests/lambda-tests/
app/tests/lambda-tests/functional_test.py
app/tests/lambda-tests/functional_test.zip
app/tests/test.py
app/app.py
Dockerfile
appspec.yaml
buildspec.yml
[main (root-commit) 3e37158] Configuring repo.
7 files changed, 179 insertions(+)
create mode 100644 Dockerfile
create mode 100644 app/app.py
create mode 100644 app/tests/lambda-tests/functional_test.py
create mode 100644 app/tests/lambda-tests/functional_test.zip
create mode 100644 app/tests/test.py
create mode 100644 appspec.yaml
create mode 100644 buildspec.yml
Username for 'https://git-codecommit.us-east-1.amazonaws.com': awsjoe-at-01234567890
Password for 'https://awsjoe-at-01234567890@git-codecommit.us-east-1.amazonaws.com':
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 2 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (12/12), 3.73 KiB | 3.73 MiB/s, done.
Total 12 (delta 0), reused 0 (delta 0), pack-reused 0
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/FlaskDemoRepo-01234567890
* [new branch]    main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

```

6. Deploy the AWS Services:

```
make deploy-infra
```

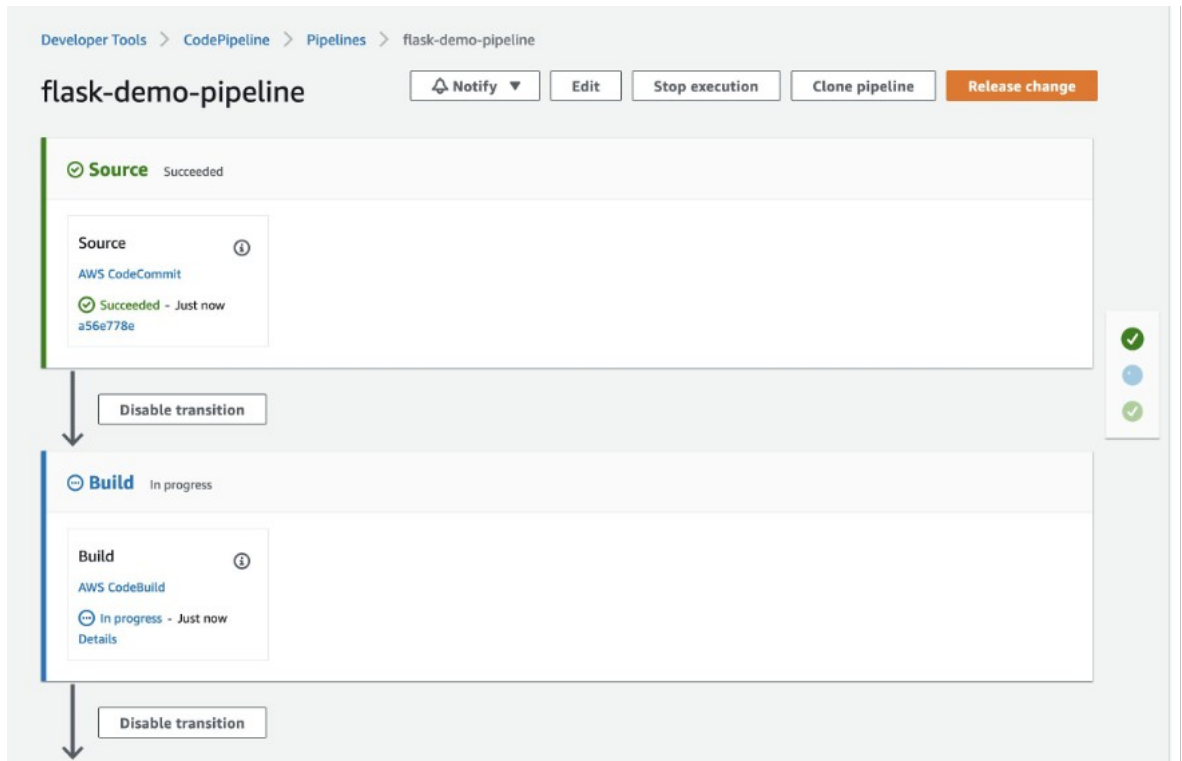
This will result in great deal of output ending in:

Apply complete! Resources: 118 added, 1 changed, 0 destroyed.

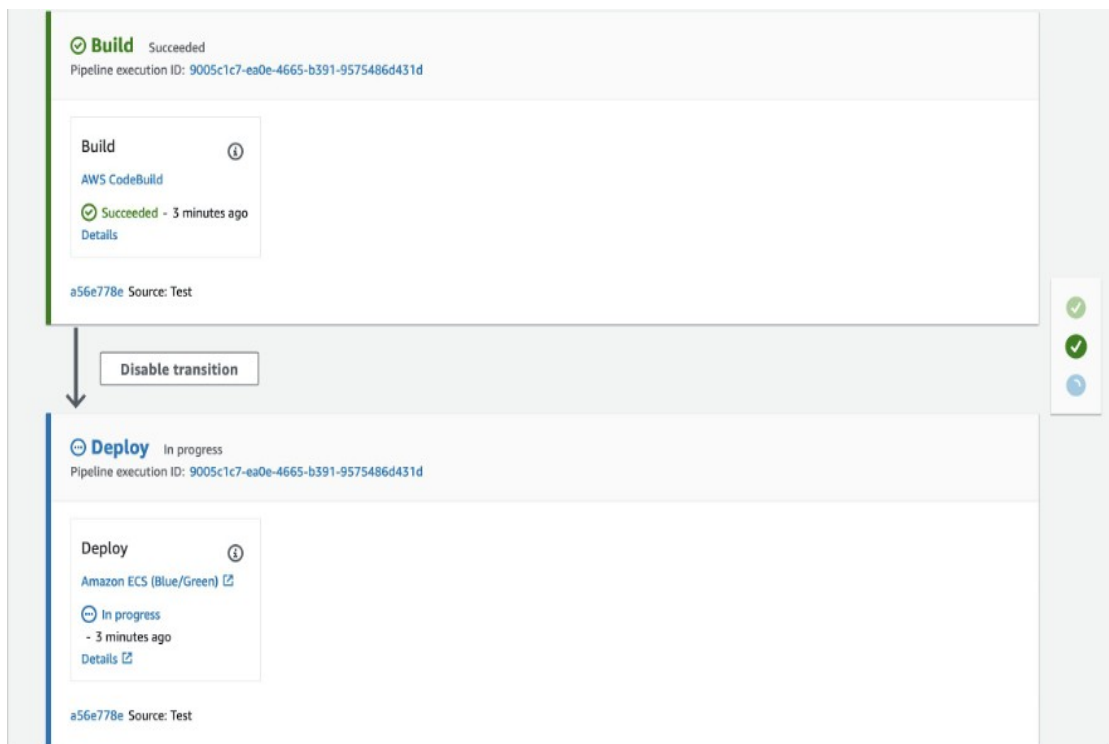
Outputs:

```
aws_lb_dns_name = "ecsalb-123456789.us-east-1.elb.amazonaws.com"
```

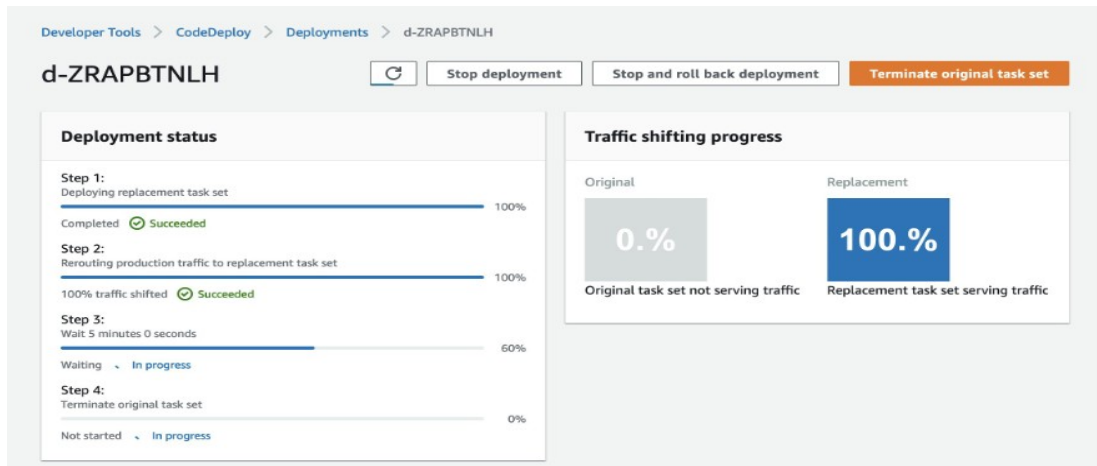
7. In the AWS Console open *CodePipeline* → *Pipelines* and select *flask-demo-pipeline*:



8. Scroll down to the *Deploy* phase and click *Details*:



9. Wait for the deployment to complete, it should take around 8 minutes:



10. Back in your terminal, run the following command to list the Terraform outputs:

terraform output

You should see output similar to the following:

aws_lb_dns_name = "ecsalb-123456789.us-east-1.elb.amazonaws.com"

11. Run the following command using the `aws_lb_dns_name` from the previous step to test that the application is functioning properly:

curl http://ecsalb-123456789.us-east-1.elb.amazonaws.com/

You should see output similar to the following:

Hello World!

PYTHON APP DEVELOPMENT ON GOOGLE CLOUD PLATFORM (GCP)

Google Cloud Platform (GCP) supports Python 3. The supported versions of Python are:

- Python 3: Preferred version, 3.5 to 3.8
- Python 2: 2.7.9 or later

The Windows version of Cloud SDK comes bundled with Python 3 and Python 2. To use Cloud SDK, your operating system must be able to run a supported version of Python.

GCP has the tools Python developers need to build cloud-native applications. You can build apps quicker with SDKs and in-IDE assistance. You can scale your apps as big, or small, as you need on Cloud Run, GKE, or Anthos.

To write Python code in the Google cloud shell, you can:

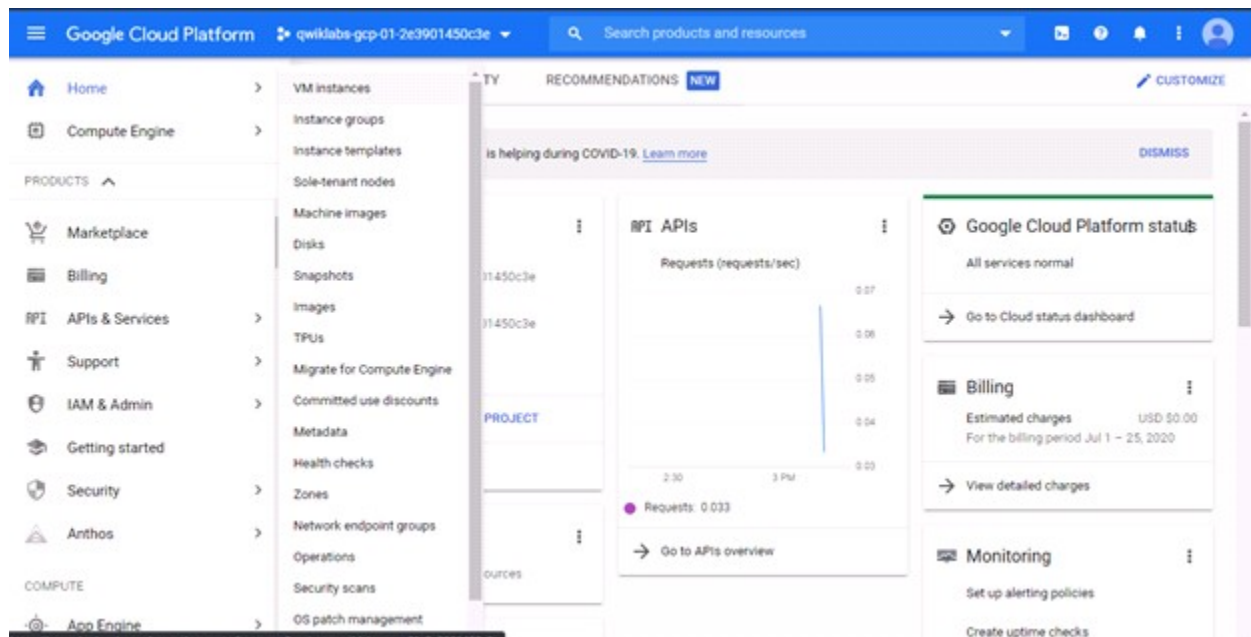
1. Select Files and Launch code editor.
2. Select File New folder and give the folder a name.
3. Create three files:
 1. app gamma: Contains configuration data.
 2. requirements TX text txt: Specifies any additional libraries needed to deploy the application.
 3. main dot pi: Contains the main body of the Python code for the web app.
 4. Import the logging library.

Google Cloud Platform supports Python 3. If you are a python developer and you are still new in Google Cloud Platform, this article will guide you step by step. Please keep on reading

Knowing the basics Google Cloud Platform

Google Cloud Console, Cloud Shell, Google Command Line, Google Editor, Virtual Machine Instance, Google Compute Engine, Google API.

Google Cloud Console: An interface basis on the web to run the google cloud platform.



Console Google Cloud Platform

Cloud Shell: a Debian-based virtual machine and as the terminal environment for you to manage GCP projects and resources easily by typing the google command line.

Google Command Line

You can utilize this instrument to perform numerous platform tasks in scripts, or command lines, or another automation.

Reference :

<https://cloud.google.com/sdk/gcloud/reference> and <https://cloud.google.com/sdk/docs/cheatsheet>

Google Cloud Editor

<https://cloud.google.com/shell/docs/viewing-and-editing-files>

Virtual Machine Instance

A virtualized operating system or application environment to implement software in the cloud system that can be run on Google infrastructure.

Google Compute Engine

A Google service on the google cloud platform permits clients to dispatch virtual machines on request.

Google API

This tool is the way for you to interact on the Google Cloud Platform by enabling the APIs and Services on the google cloud console

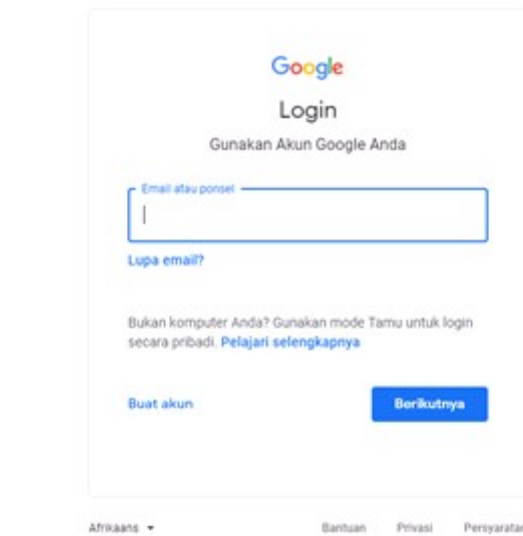
Set Up Python Environment

Prepare the Python environment for your local machine

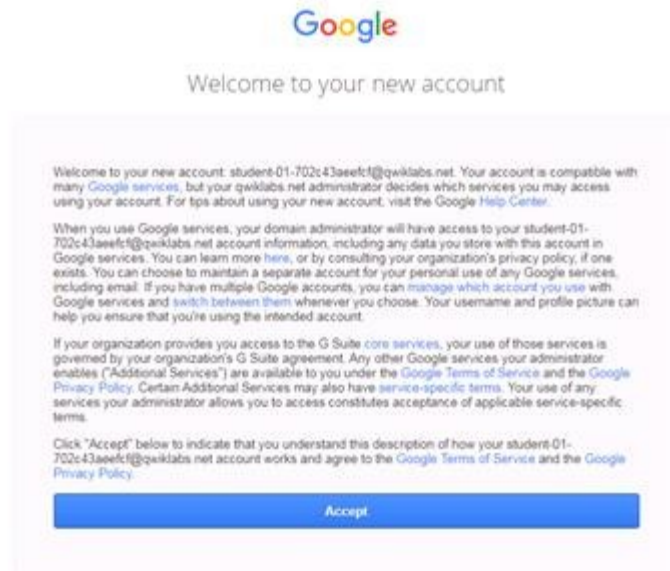
- Install the latest version of Python.
- Use `venv` to isolate dependencies.
- Install an editor (optional): Sublime Text, Atom, Pycharm
- Install the Cloud SDK (optional).
- Install the Cloud Client Libraries for Python (optional).
- Install other useful tools.

Step by step to Run App Dev with python on GCP

You must have an account on the GCP first. You should register for a new account if you are a new GCP user. If you already had it, log in to your GCP account.

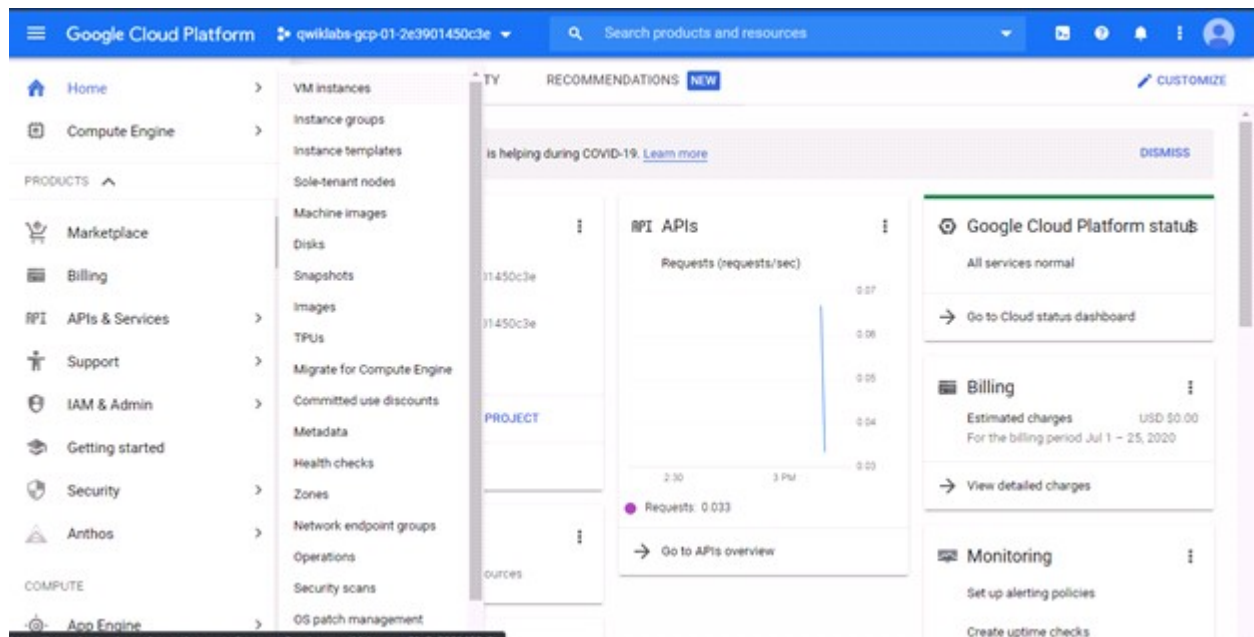
The image shows the Google Login page. At the top is the Google logo, followed by the word "Login" and the instruction "Gunakan Akun Google Anda". Below this is a text input field labeled "Email atau ponsel" with a cursor inside. To the left of the input field is a link "Lupa email?". Below the input field is a paragraph: "Bukan komputer Anda? Gunakan mode Tamu untuk login secara pribadi. [Pelajari selengkapnya](#)". At the bottom left is a link "Buat akun" and at the bottom right is a blue button labeled "Berikutnya". At the very bottom, there are links for "Afrikaans", "Bantuan", "Privasi", and "Persyaratan".

Login to GCP Account

The image shows the "Welcome to your new account" page. At the top is the Google logo, followed by the text "Welcome to your new account". Below this is a large text block containing the following information: "Welcome to your new account: student-01-702c43aeefcf@qwiklabs.net. Your account is compatible with many Google services, but your qwiklabs.net administrator decides which services you may access using your account. For tips about using your new account, visit the [Google Help Center](#)." "When you use Google services, your domain administrator will have access to your student-01-702c43aeefcf@qwiklabs.net account information, including any data you store with this account in Google services. You can learn more [here](#), or by consulting your organization's privacy policy, if one exists. You can choose to maintain a separate account for your personal use of any Google services, including email. If you have multiple Google accounts, you can [manage which account you use](#) with Google services and [switch between them](#) whenever you choose. Your username and profile picture can help you ensure that you're using the intended account." "If your organization provides you access to the G Suite [core services](#), your use of those services is governed by your organization's G Suite agreement. Any other Google services your administrator enables ("Additional Services") are available to you under the [Google Terms of Service](#) and the [Google Privacy Policy](#). Certain Additional Services may also have [service-specific terms](#). Your use of any services your administrator allows you to access constitutes acceptance of applicable service-specific terms." "Click 'Accept' below to indicate that you understand this description of how your student-01-702c43aeefcf@qwiklabs.net account works and agree to the [Google Terms of Service](#) and the [Google Privacy Policy](#)." At the bottom is a blue button labeled "Accept".

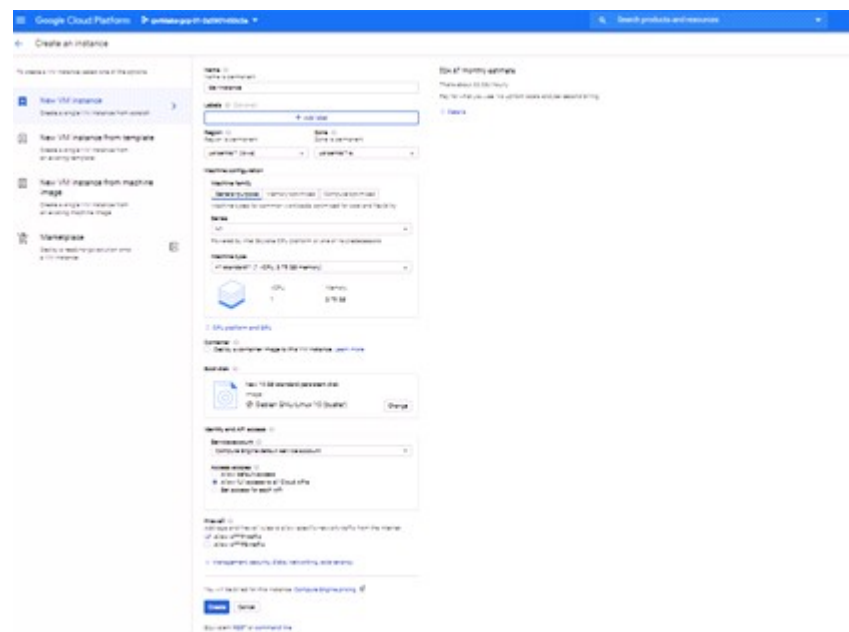
Welcome To GCP Account

After that, You need to create the virtual machine instance with google compute engine first. Here are the steps to create VM instance on Google Console :



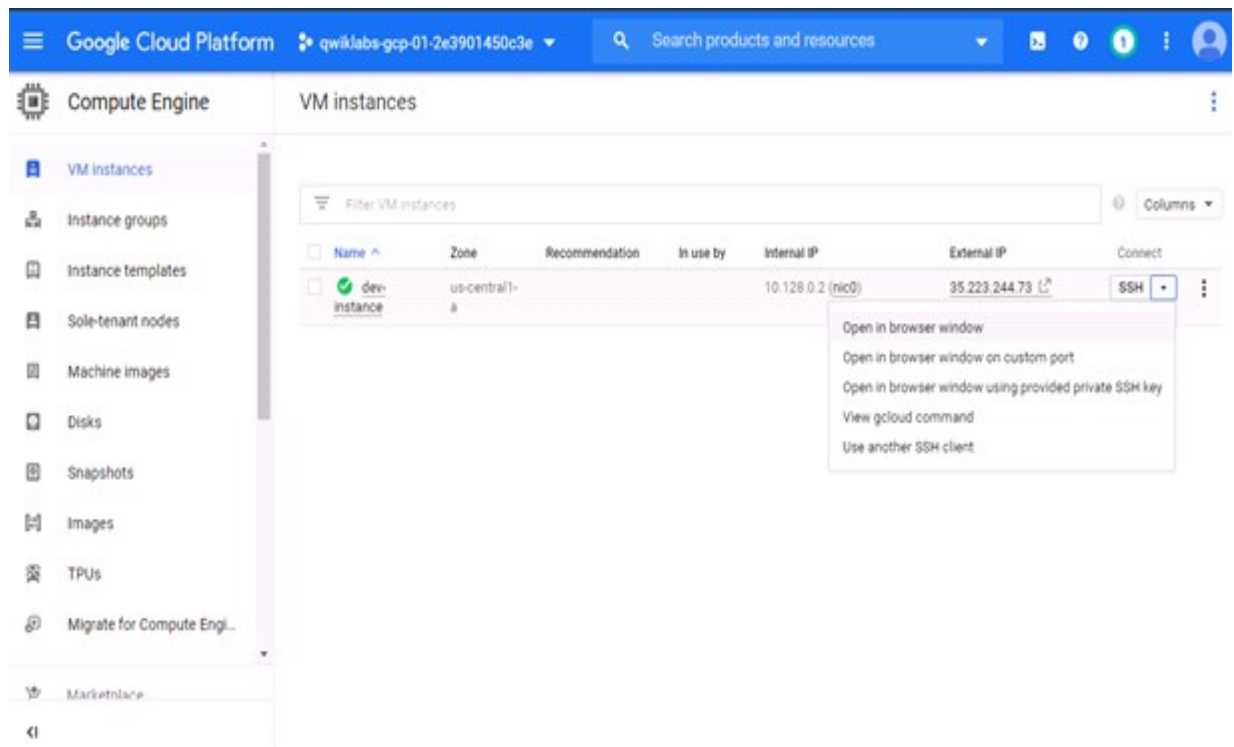
Console Google Cloud Platform

1. On the Google Cloud Console, Choose Compute Engine, VM Instance then click create
2. Create the instance: Type The name, select the Region and the Zone
3. Allow full access to all Cloud APIs in the Identity and API section.
4. Enable Allow HTTP traffic in the firewall.
5. Create



Create VM Instance on GCP

6. Click The instance you just created on the VM instance page
7. Click SSH to open the browser window



VM Instance Page
Still in SSH Window, Now Install Software.

```

student-01-702c43aeef@dev-instance: ~ - Google Chrome
ssh.cloud.google.com/projects/qwiklabs-gcp-01-2e3901450c3e/zones/us-central1-a/instances/dev-instance?useAdminProxy=true&aut...

Setting up libcc1-0:amd64 (8.3.0-6) ...
Setting up liblocale-gettext-perl (1.07-3+b4) ...
Setting up liblsan0:amd64 (8.3.0-6) ...
Setting up libitm1:amd64 (8.3.0-6) ...
Setting up libalgorithm-merge-perl (0.08-3) ...
Setting up binutils-x86-64-linux-gnu (2.31.1-16) ...
Setting up libtsan0:amd64 (8.3.0-6) ...
Setting up python3-distutils (3.7.3-1) ...
Setting up dh-python (3.20190308) ...
Setting up manpages-dev (4.16-2) ...
Setting up python3-setuptools (40.8.0-1) ...
Setting up binutils (2.31.1-16) ...
Setting up dpkg-dev (1.19.7) ...
Setting up libgcc-8-dev:amd64 (8.3.0-6) ...
Setting up cpp (4:8.3.0-1) ...
Setting up libstdc++-8-dev:amd64 (8.3.0-6) ...
Setting up gcc-8 (8.3.0-6) ...
Setting up gcc (4:8.3.0-1) ...
Setting up libexpat1-dev:amd64 (2.2.6-2+deb10u1) ...
Setting up g++-8 (8.3.0-6) ...
Setting up libpython3.7-dev:amd64 (3.7.3-2+deb10u1) ...
Setting up python3.7-dev (3.7.3-2+deb10u1) ...
Setting up g++ (4:8.3.0-1) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
Setting up build-essential (12.6) ...
Setting up libpython3-dev:amd64 (3.7.3-1) ...
Setting up python3-dev (3.7.3-1) ...
Processing triggers for man-db (2.8.5-2) ...
Processing triggers for libc-bin (2.28-10) ...
student-01-702c43aeef@dev-instance:~$ curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
% Total % Received % Xferd Average Speed Time Time Current
100 1825k 100 1825k 0 0 7243k 0 --:--:-- --:--:-- --:--:-- 7243k
student-01-702c43aeef@dev-instance:~$ sudo python3 get-pip.py
Collecting pip
  Downloading pip-20.1.1-py2.py3-none-any.whl (1.5 MB)
    | 1.5 MB 3.4 MB/s
Collecting wheel
  Downloading wheel-0.34.2-py2.py3-none-any.whl (26 kB)
Installing collected packages: pip, wheel
Successfully installed pip-20.1.1 wheel-0.34.2
student-01-702c43aeef@dev-instance:~$

```

SSH Window

Here is a step by step guidance for you:

1. Update The Debian Package list: Sudo apt-get update
2. Install Git: Sudo apt-get install git
3. Install python :

sudo apt-get install python3-setuptools python3-dev build-essential

4. Install pip

curl <https://bootstrap.pypa.io/get-pip.py> -o get-pip.py

Sudo python3 get-pip.py

SSH Window

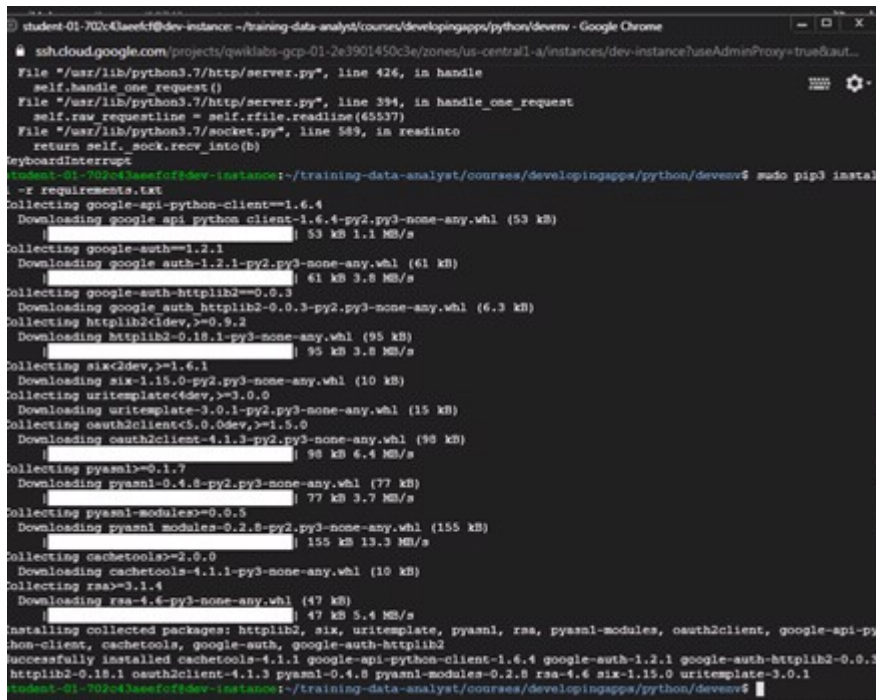
Verify Python Installation

1. python3 — version
2. pip3 — version
3. git clone <https://github.com/GoogleCloudPlatform/training-data-analyst> (This is a sample file from Google Cloud Platform)
4. cd ~/training-data-analyst/courses/developingapps/python/devenv/
5. sudo python3 server.py

Then Configure your Virtual Machine so you can run the Application

Navigation menu ==> **Compute Engine** ==> **Virtual Instances**, and click on the **External IP address** for the instance that you just created it before.

1. Return to the SSH window, and stop the application by pressing **Ctrl+c**.
2. sudo pip3 install -r requirements.txt
3. python3 list-gce-instances.py <PROJECT_ID> — zone=<YOUR_VM_ZONE>



SSH Window

CONTAINERIZED PYTHON WEB APP ON AZURE WITH MONGO DB

You can use Python with Microsoft Azure to perform activities such as:

- Remote debugging on Windows, Linux, and Mac OS
- Cluster debugging

You can install Python on an Azure VM by:

1. RDP to the Windows VM
2. Install Python
3. Download the Python packages locally and upload them to the Azure VM

You can trigger a Python script on an Azure Windows VM by:

1. Using Azure Event Grid or Azure Service Bus to send the event or message
2. Using Azure Automation or Azure Virtual Machine Extension to execute the script

You can create a free account with Microsoft Azure to get started with 12 months of free services, 40+ services that are always free, and USD200 in credit.

Azure Functions Python Worker supports Python versions 3.6, 3.7, 3.8, and 3.9. You can check your Python interpreter version by:

- `py --version` in Windows
- `python3 --version` in Unix-like systems

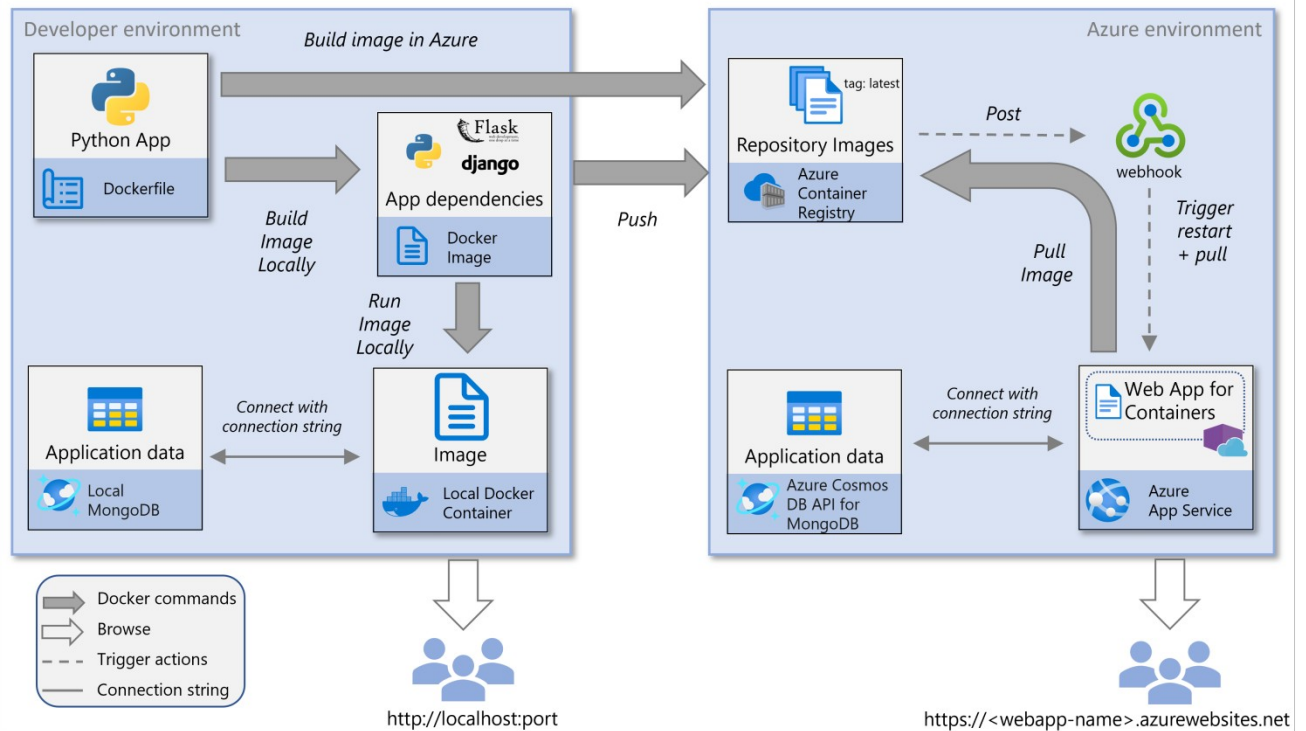
to containerize a Python web app and deploy it to Azure. The single container web app is hosted in [Azure App Service](#) and uses [MongoDB for Azure Cosmos DB](#) to store data. App Service [Web App for Containers](#) allows you to focus on composing your containers without worrying about managing and maintaining an underlying container orchestrator. When building web apps, Azure App Service is a good option for taking your first steps with containers. For more information about using containers in Azure, see [Comparing Azure container options](#).

- Build and run a [Docker](#) container locally. *This step is optional.*
- Build a [Docker](#) container image directly in Azure.
- Configure an App Service to create a web app based on the Docker container image.

Following this tutorial, you'll have the basis for Continuous Integration (CI) and Continuous Deployment (CD) of a Python web app to Azure.

SERVICE OVERVIEW

The service diagram supporting this tutorial shows two environments (developer environment and Azure) and the different Azure services used in the tutorial.



The components supporting this tutorial and shown in the diagram above are:

- [Azure App Service](#)
 - The underlying App Service functionality that enables containerization is Web App for Containers. Azure App Service uses the [Docker](#) container technology to host both built-in images and custom images. In this tutorial, you'll build an image from Python code and deploy it to Web App for Containers.
 - Web App for Containers uses a webhook in the registry to get notified of new images. A push of a new image to the repository triggers App Service to pull the image and restart.
- [Azure Container Registry](#)
 - Azure Container Registry enables you to work with Docker images and its components in Azure. It provides a registry that's close to your deployments in Azure and that gives you control over access, making it possible to use your Azure Active Directory groups and permissions.
 - In this tutorial, the registry source is Azure Container Registry, but you can also use Docker Hub or a private registry with minor modifications.
- [Azure Cosmos DB for MongoDB](#)
 - The Azure Cosmos DB for MongoDB is a NoSQL database used in this tutorial to store data.
 - Access to Azure Cosmos DB resource is via a connection string, which is passed as an environment variable to the containerized app.

AUTHENTICATION

In this tutorial, you'll build a Docker image (either locally or directly in Azure) and deploy it to Azure App Service. The App Service pulls the container image from an Azure Container Registry repository.

The App Service uses [managed identity](#) to pull images from Azure Container Registry. Managed identity allows you to grant permissions to the web app so that it can access other Azure resources without the need to specify credentials. Specifically, this tutorial uses a system assigned managed identity. Managed identity is configured during setup of App Service to use a registry container image.

The tutorial sample web app uses MongoDB to store data. The sample code connects to Azure Cosmos DB via a connection string.

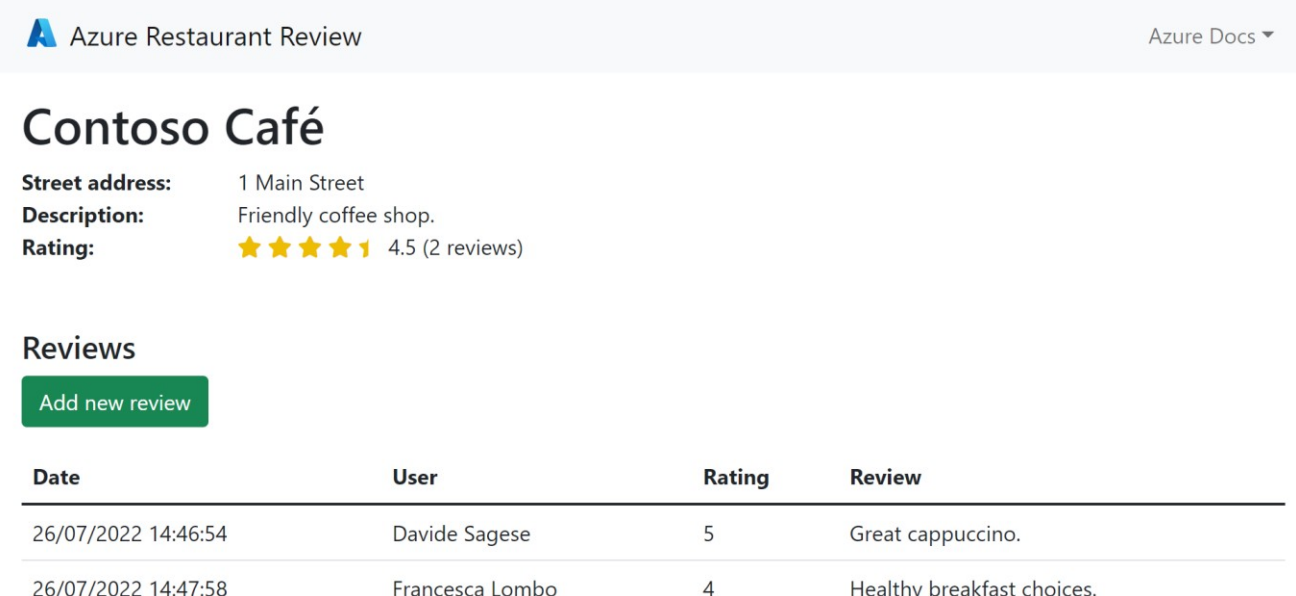
PREREQUISITES

To complete this tutorial, you'll need:

- An Azure account where you can create:
 - [Azure Container Registry](#)
 - [Azure App Service](#)
 - [MongoDB for Azure Cosmos DB](#) (or access to equivalent). To create an Azure Cosmos DB for MongoDB database, you can use the steps for [Azure portal](#), [Azure CLI](#), [PowerShell](#), or [VS Code](#). The sample tutorial requires that you specify a MongoDB connection string, a database name, and a collection name.
- [Visual Studio Code](#) or [Azure CLI](#), depending on what tool you'll use.
 - For Visual Studio Code, you'll need the [Docker extension](#) and [Azure App Service extension](#).
- Python packages:
 - [PyMongo](#) for connecting to MongoDB.
 - [Flask](#) or [Django](#) as a web framework.
- [Docker](#) installed locally if you want to run container locally.

SAMPLE APP

The Python sample app is a restaurant review app that saves restaurant and review data in MongoDB. For an example of a web app using PostgreSQL, see [Deploy a Python web app to Azure with managed identity](#). At the end of the tutorial, you'll have a restaurant review app deployed and running in Azure that looks like the screenshot below.



The screenshot shows a web application titled "Azure Restaurant Review" with a header bar containing the Azure logo and "Azure Docs" link. The main content area features "Contoso Café" with details: "Street address: 1 Main Street", "Description: Friendly coffee shop.", and "Rating: ★★★★★ 4.5 (2 reviews)". Below this is a "Reviews" section with a green "Add new review" button. A table lists two reviews: one by Davide Sageese with a rating of 5 and review "Great cappuccino.", and another by Francesca Lombo with a rating of 4 and review "Healthy breakfast choices."

1. CLONE OR DOWNLOAD THE SAMPLE APP
2. BUILD A DOCKER IMAGE
3. SET UP MONGODB
4. RUN THE IMAGE LOCALLY IN A CONTAINER

Python for MapReduce:

MapReduce is a programming model for processing and generating large data sets with a parallel distributed algorithm on a cluster. The MapReduce programming style was inspired by the functional programming constructs map and reduce.

In Python, MapReduce can be implemented using the Hadoop Streaming API. The Hadoop Streaming API allows you to write MapReduce programs in Python and run them on a Hadoop cluster.

To write a MapReduce program in Python, you need to create a mapper function and a reducer function. The mapper function takes a key-value pair as input and produces a list of key-value pairs as output. The reducer function takes a list of key-value pairs as input and produces a single key-value pair as output.

The following is an example of a MapReduce program in Python:

```
defmapper(key, value):
```



```
# Split the value into a list of words.
words = value.split()

# For each word, emit a key-value pair with the word as the key and 1 as the value.
for word in words:
    yield word, 1
```

```
def reducer(key, values):
    # Sum the values for each key.
    sum = 0
    for value in values:
        sum += value
```

```
# Emit a key-value pair with the key as the word and the sum as the value.
yield key, sum
Use code with caution.
Learn more
```

This program counts the number of times each word appears in a text file.

To run this program on a Hadoop cluster, you need to create a Hadoop job and specify the mapper and reducer functions. You can then submit the job to the Hadoop cluster.

MapReduce is a powerful tool for processing large data sets. It is a good choice for tasks that can be divided into independent tasks that can be performed in parallel.

When we deal with “BIG” data, as the name suggests dealing with a large amount of data is a daunting task. MapReduce is a built-in programming model in Apache Hadoop. It will parallel process your data on the cluster.

This article will look into how MapReduce works with an example dataset using Python.

MAPREDUCE : ANALYSE BIG DATA

[MapReduce](#) will **transform** the data using **Map** by dividing data into key/value pairs, getting the output from a map as an input, and **aggregating** data together by **Reduce**. MapReduce will deal with all your cluster failures.

HOW MAPREDUCE WORKS

To understand MapReduce, let's take a real-world example. You have [a dataset that consists of hotel reviews and ratings](#). Now you need to find out how many reviews each rating.

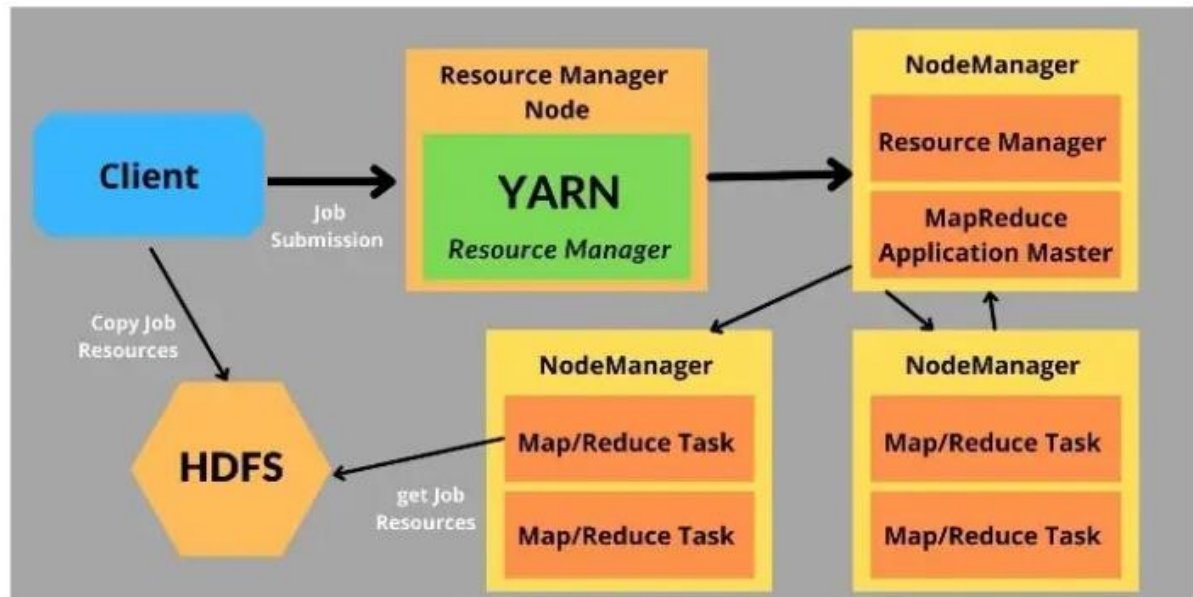
Hotel_Reviews

Review	Rating
nice hotel expensive parking got good deal stay hotel anniversary, arrived late evening took advice previous reviews did valet parking, check quick easy, little disappointed non-existent view room room clean ni	4
ok nothing special charge diamond member hilton decided chain shot 20th anniversary seattle, start booked suite paid extra website description not, suite bedroom bathroom standard hotel room, took printed	2
nice rooms not 4* experience hotel monaco seattle good hotel n't 4* level.positives large bathroom mediterranean suite comfortable bed pillowsattentive housekeeping staffnegatives ac unit malfunctioned stay	3
unique, great stay, wonderful time hotel monaco, location excellent short stroll main downtown shopping area, pet friendly room showed no signs animal hair smells, monaco suite sleeping area big striped curt	5
great stay great stay, went seahawk game awesome, downfall view building did n't complain, room huge staff helpful, booked hotels website seahawk package, no charge parking got voucher taxi, problem taxi	5
love monaco staff husband stayed hotel crazy weekend attending memorial service best friend husband celebrating 12th wedding anniversary, talk mixed emotions, booked suite hotel monte carlos, loaned bea	5
cozy stay rainy city, husband spent 7 nights monaco early january 2008. business trip chance come ride.we booked monte carlo suite proved comfortable longish stay, room 905 located street building, street no	5
excellent staff, housekeeping quality hotel chocked staff make feel home, experienced exceptional service desk staff concierge door men maid service needs work, maid failed tuck sheets foot bed instance soi	4
hotel stayed hotel monaco cruise, rooms generous decorated uniquely, hotel remodeled pacific bell building charm sturdiness, everytime walked bell men felt like coming home, secure, great single travelers, lo	5
excellent stayed hotel monaco past w/e delight, reception staff friendly professional room smart comfortable bed, particularly liked reception small dog received staff guests spoke loved, mild negative distance	5
poor value stayed monaco seattle july, nice hotel priced 100- 150 night not, hotel takes beating quotient, experience simply average, nothing exceptional paying 300+ n't ca n't terribly disappointed, wife stayed r	2
nice value seattle stayed 4 nights late 2007. looked comparable hilton marriott westin area points/miles n't gave monaco shot, pleasantly surprised nice room service quick tasty bed especially comfortable unlik	4
nice hotel good location hotel kimpton design whimsical vibe fun, staff young casual problem hotel busy stay friendly helpful, group reserved rooms gave connecting rooms fuss, not busy week.the rooms decer	4
nice hotel not nice staff hotel lovely staff quite rude, bellhop desk clerk going way make things difficult, waited forever check heavy bags no help getting through double doors room, worst desk clerk checking	3
great hotel night quick business trip, loved little touches like goldfish leopard print robe, complaint wifi complimentary not internet access business center, great location library service fabulous,	4
horrible customer service hotel stay february 3rd 4th 2007my friend picked hotel monaco appealing website online package included champagne late checkout 3 free valet gift spa weekend, friend checked roo	1
disappointed say anticipating stay hotel monaco based reviews seen tripadvisor, definitely disappointment, decor room hotel envisioned nice, housekeeping staff impressive extremely polite cheery helpful, desk	2
fantastic stay monaco seattle hotel monaco holds high standards kimpton hotel line, having stayed kimpton hotels cities easily say seattle hotel monaco best seen, service attentive prompt, based member kim	5
good choice hotel recommended sister, great location room nice, comfortable bed- quiet- staff helpful recommendations restaurants, pike market 4 block walk, stay,	5
hmmmm say really high hopes hotel monaco chose base girlfriend shopping trip seattle, stay say given competition seattle just okay, hotel lot nice features little things detract bedding super soft luxurious coi	3
service service service spent week g-friend labor day bumbershoot, gray line airporter drops corner hotel 10 person cab 28 total make sure flat rate town car 38. location central downtown street w. it _ Ç. é. qui	5
excellent stay, delightful surprise stay monaco, thoroughly enjoyed stay, room comfortable lovely amenities friendly staff, especially enjoyed hour indulgence, definitely come,	5
good value downtown hotel monaco seattle great option pricey area town, rooms w street going close 400 night, hotel monaco unique interesting cozy, bed/linens quality aveda products bathroom nice touch.ye	4
hotel monaco great location service hotel monaco centrally located provides excellent service, recently stayed 4/23-5/1, originally checked slight problem informed booked queen beds expedia room king bed, f	5
great location need internally upgrade advantage north end downtown seattle great restaurants nearby good prices, rooms need updated literally thought sleeping 1970 bed old pillows sheets, net result bad nig	2
n't mind noise place great, read reviews noise used hotel website book room, interesting site gives checkbox options 1 room farthest elevator 2 room higher floor 3 quiet room, getting quietest room possible ch	3
loved, stayed warwick overnight getaway enjoy christmas shopping, warwick exceeded expectations, staff wonderful extremely friendly room clean service lounge wonderful, came contact hotel friendly, wome	3
met expectations centrally located hotel blocks water popular nightlife shopping options belltown downtown, classify property star location, paid wedding rate not sure fares value proposition, room spacious lit	3
nice hotel husband stayed warwick 4 years ago liked hotel location did not hesitate book trip.we asked received non-smoking room king bed high floor, hoping hotel nice view space needle, check-in nice friend	4
good hotel not large hotel newly decorated, rooms good size clean, excellent location 2 blocks centre town 4 blocks market area.hotel resturant bustling place alot diners coming street, good meal there.overall	4
good choice seattle stayed night business booked company, overall satisfactory arrived late evening room 17th floor clean comfortable great view, bathroom little small clean equipped, good night sleep enjoyee	4
great location expensive parking warwick heart seattle easy walking distance pikes place monorail terminal downtown shops, stayed twice recently start finish tour pacific north west, staff helpful knowledgabl	4
noise airconditioner-a standard, arranged stay travel agency unfortunately warwick seattle hotel dissatisfaction trip, 3 night stay warwick changed 3 rooms, starting minute stay hotel personnel didn't make fee	1
good location poor cleanliness warwick hotel great location seattle, close shopping pike place seattle centre, really lets hotel cleanliness, bed linen torn curtains torn floor stains, hotel advertises having balcon	2
good place spending big bucks warwick plenty comfortable nice, nice view space needle room, n't expect balcony, just 18 inch ledge ironwork falling overboard, open patio door partially step view lake union ell	4
nice hotel trip seattle wanted stay downtown, good rate hotel decided stay warwick, clean stayed 17th floor excellent view space needle downtown, nice able open sliding door let cool air, bed comfy pillows no	4

Dataset Snapshot

So what you will do is you will go through the Rating columns and count how many reviews are there for 1,2,3,4,5. It is pretty easy to do if we do have a small amount of data but when it comes to big data it can be billions or trillions of data. Therefore we can use MapReduce.

WHAT'S HAPPENING UNDER THE HOOD



- Client node will submit MapReduce Jobs to The resource manager Hadoop YARN.
- Hadoop YARN resource managing and monitoring the clusters such as keeping track of available capacity of clusters, available clusters, etc. Hadoop Yarn will copy the needful data Hadoop Distribution File System(HDFS) in parallel.
- Next Node Manager will manage all the MapReduce jobs. MapReduce application master located in Node Manager will keep track of each of the Map and Reduce tasks and distribute it across the cluster with the help of YARN.
- Map and Reduce tasks connect with HDFS cluster to get needful data to process and output data.

LET'S GET STARTED

MapReduce is written in Java but capable of running g in different languages such as Ruby, Python, and C++. Here we are going to use Python with the MR job package.

We will count the number of reviews for each rating(1,2,3,4,5) in the [dataset](#).

Step 1: Transform raw data into key/value pairs in parallel.

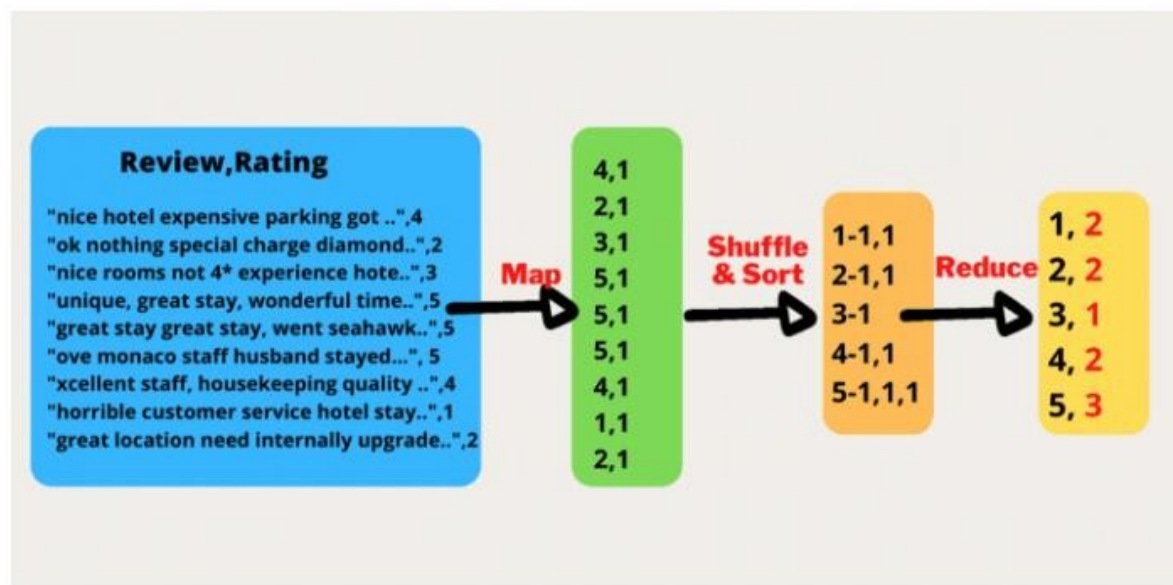
The mapper will get the data file and make the Rating the key and the values will be the reviews. We will add number 1 for reviews.

Step 2: Shuffle and sort by the MapReduce model.

The process of transferring mappers' intermediate output to the reducer is known as shuffling. It will collect all the reviews(number 1s) together with the individual key and it will sort them. it will get sorted by key.

Step3: Process the data using Reduce.

Reduce will count each value(number 1) for each key.



PREREQUISITES

1. Install [Python](#)
2. Install [Hadoop](#)
3. Install [MRJob](#)

pip install mrjoborpython setup.py test && python setup.py install

Here we are having a Job called '*NoRatings*' consisting of a *Mapper function*, a *Reducer function*, and a *definition*. A *Step function* is used to define our functions for mappers and reducers in our job.

Note: Used [yield](#) to return *statement in a function*.

Here is the full code which is saved as *NoRatings.py*

```
from mrjob.job import MRJob
from mrjob.step import MRStep
import csv#split by ,
columns = 'Review,Rating'.split(',')class NoRatings(MRJob):
    def steps(self):
        return[
            MRStep mapper=self.mapper_get_ratings,
            reducer=self.reducer_count_ratings
        ]
#Mapper function
def mapper_get_ratings(self, _, line):
    reader = csv.reader([line])
    for row in reader:
        zipped=zip(columns,row)
        diction=dict(zipped)
        ratings=diction['Rating']
        #outputing as key value pairs
        yield ratings, 1#Reducer function
def reducer_count_ratings(self, key, values):
    yield key, sum(values)if __name__ == "__main__":
    NoRatings.run()
```

- Verify whether the both dataset(Hotel_Reviews.csv) and the python file(*NoRatings.py*) are there in the directory.

ls
#or

hadoop fs -ls

- Run python script with the mrjob. We can run the *NoRatings.py* script locally or with Hadoop.

```
#run locally
#Hotel_Reviews.csv is the dataset.
#verify whether the both dataset(Hotel_Reviews.csv) and the python file(NoRatings.py) are there in current directory. using 'ls'
command.python NoRatings.py Hotel_Reviews.csv#run with hadoop#python [python file] -r hadoop --hadoop-streaming-jar
[The_path_of_Hadoop_Streaming_jar] [dataset]python NoRatings.py -r hadoop --hadoop-streaming-jar
/usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.7.1 jar Hotel_Reviews.csv
```

```
Running step 1 of 1...
Streaming final output from /tmp/No.root.20210821.155528.892551/output...
"1"      1421
"2"      1793
"3"      2184
"4"      6039
"5"      9054
"Rating"      1
```

Output

CONCLUSION

As you have seen MapReduce is necessary to deal with a large amount of data.

Python Packages of Interest:

top ten most important, useful and ubiquitous Python packages

1 NUMPY

You can do basic mathematical operations without any special Python packages. However, if you're going to do any kind of complex math, the NumPy package will make your coding life much easier.

NumPy provides tools to help build multi-dimensional arrays and perform calculations on the data stored in them. You can solve algebraic formulas, perform common statistical operations, and much more.

While NumPy is a valuable Python package for a variety of general-purpose programming tasks, it's particularly important if you want to do machine learning, since it provides part of the foundation for libraries like [TensorFlow](#).

2 PENDULUM

If you have at least a little Python programming experience, you probably know that you can use the datetime module to manage dates and times within an application.

While datetime is great for basic work along these lines, the Pendulum Python package makes it easier to do more complex coding involving dates and times. It's more intuitive to work with, and it manages time zones automatically. Best of all, Pendulum is designed to be a drop-in replacement for datetime. That means you can use it with code you've already written based on datetime. With only a few exceptions, Pendulum will work just as well, without the need to modify the code, while providing extra features not present in plain-old datetime.

3 PYTHON IMAGING LIBRARY

If your Python application interacts with images in any way, the Python imaging library, also known as PIL or Pillow, is a Python must-have. It makes it easy to write code that opens, modifies, and saves images in a variety of formats. If you're doing more advanced work with images (like image recognition, in which case OpenCV would be a good package to consider), Pillow won't cut it on its own. But for basic image importing, manipulation, and exporting, Pillow is your go-to solution.

4 MOVIEPY

MoviePy is to videos what Pillow is to images. It provides a range of functionality for common tasks associated with importing, modifying, and exporting video files. It also lets you do things like insert titles into videos or rotate videos 90 degrees (if for some reason you decide you want to do that).

Like Pillow, MoviePy is not intended as a tool for advanced data manipulation. If you're writing a video editing app, you'll probably also need to rely on OpenCV (which can work with videos as well as images) to provide the advanced functionality that MoviePy lacks. But for most standard tasks involving videos in Python code, MoviePy gets the job done quite well.

5 REQUESTS

Writing code that sends HTTP requests can be tricky, due in no small part to the fact that HTTP does not exactly format data in a way that is easy for humans to read.

The Requests Python package (motto: "HTTP for Humans") tackles this problem by automating many of the tedious tasks that you would otherwise need to perform in order to send HTTP requests from your application. It removes the need to add query strings, or do POST form encoding. It also keeps connections with HTTP servers alive automatically, eliminating the need to write a bunch of code for doing that.

In short, if your application sends any data over HTTP, Requests is a must-have package.

Recommended Reads

Want to develop a Python app with a Graphical User Interface (GUI)? There are a variety of packages designed to help you do that (indeed, we could make a top ten list of just Python GUI packages). But I think most Python developers would agree that Tkinter is the most important — and most commonly used — framework for creating GUIs. It binds Python to the TK GUI toolkit, which works on virtually every modern operating system.

Unless you have a strong preference for a different GUI toolkit, [Tkinter](#) is probably the best place to start when creating a Python GUI.

7 PYQT

The preceding sentence notwithstanding, PyQt, another Python package for building GUIs, is also a strong contender. It provides bindings to (you guessed it) the Qt toolkit, which is also cross-platform. It's intended for heavier-duty GUI programming than Tkinter. That means that PyQt may be overkill if you're building an app that has a pretty simple interface — say, just a window with some buttons and text fields — but it is a good tool if you want to build a complex, multi-dimensional GUI.

Recommended Reads

There is a long list of Python packages designed for working with complex data sets. But arguably, [Pandas](#) is the most important. Pandas helps you manipulate and analyze large sets of data without having to learn a specialized data-processing language like R.

Pandas has its limits in that it's not intended for advanced statistical modelling (in that case, you would want to learn R, or use a Python package like statsmodels). But if you need to do things like process time-series data or perform statistical analysis on a data set, Pandas has you covered.

9 PYWIN32

For Windows Python programming in particular, Pywin32 is a must-have package. It provides access to many of the native Windows API functions, allowing you to do things like interact with the Windows registry, use the Windows clipboard, and much more.

Pywin32 won't do you much good if you're building a cross-platform Python app, but Windows developers might find that they like it so much that they use it instead of native Windows tooling.

10 PYTEST

If you have a Python development project of any complexity, being able to perform testing on new code is essential. The Pytest package provides a variety of modules to help you do this. Whether it's a simple unit test or a more complex functional test, Pytest can help you write it.

PYTHON WEB FRAMEWORK :

Python Web framework is a collection of packages or modules that allow developers to write Web applications or services. With it, developers don't need to handle low-level details like protocols, sockets or process/thread management.

Python web framework will help you with:

- Interpreting requests (getting form parameters, handling cookies and sessions,...)
- Producing responses (presenting data as HTML or in other formats,...)
- Storing data persistently (and other things)

Now, let's look at the most useful and famous Python web framework to help you with Web development.

PYTHON FULL-STACK FRAMEWORKS

A full-stack framework in Python is one which attempts to provide a complete solution for applications. It attempts to supply components for each layer in the stack.

A. DJANGO

Django Python is a framework for perfectionists with deadlines. With it, you can build better Web apps in much less time, and in less code. Django is known for how it focusses on automating. It also believes in the DRY (Don't Repeat Yourself) principle.

Django was originally developed for content-management systems, but is now used for many kinds of web applications. This is because of its templating, automatic database generation, DB access layer, and automatic admin interface generation. It also provides a web server for development use.

Giant companies that use Django Python are- Instagram, Pinterest, Disqus, Mozilla, The Washington Times, and Bitbucket. In fact, when we think of the terms 'framework' and 'Python', the first thing that comes to our minds is Django.

Some of its features are-

Django helps developers avoid various common security mistakes.

With this framework, developers can take web applications from concept to launch within hours.

THE USER AUTHENTICATION SYSTEM OF THIS FRAMEWORK PROVIDES A SECURE WAY TO MANAGE USER ACCOUNTS AND PASSWORDS. TURBOGEARS



Python TurboGears — Python Web Framework

With TurboGears, you can create a database-driven, ready-to-extend application in just a few minutes.

It is an MVC web framework with ORM with real multi-database support and support for horizontal data partitioning. It also has a widget system to simplify the development of AJAX apps. You may additionally install its template engine Kajiki.

Learn: [How to Install Python on Windows](#)

TurboGears is a microframework and a full-stack solution. Its PyPI package is called tg.devtools.

C. WEB2PY



Python Web Framework — Python web2py

With web2py, you can develop, deploy, debug, test, administer the database, and maintain applications via the provided web interface. It has no configuration files, and you can even run it off a USB drive.

web2py uses the MVC built-in ticketing system to manage errors.

D. CUBIC WEB

CubicWeb is a semantic web application framework that features a query language and a selection+view mechanism. It also features multiple databases, security, workflows, and reusable components.

E. DJANGO -HOTSAUCE

Django-hotsauce is a general-purpose web toolkit that sits on top of Django and other frameworks. It is an interactive Pythonic API that will let you create scalable web applications using the WSGI 1.0 spec. It also provides native bindings for the Schevo DBMS, Durus, ZODB, and Authkit projects.

Learn: [7 Reasons Why Should I Learn Python in 2018](#)

F. GIOTTO

A strict MVC framework that strictly separates Model, View and Controller elements, Giotto makes sure that designers, Web developers, and sysadmins can work independently. It also includes controller modules that allow you to build applications on top of the web, irc or the command line. These are all the most popular Python web framework.

G. GROK

Grok was built on the existing Zope 3 libraries. It aims to provide an easier learning curve, and a more agile development experience by emphasizing on convention over configuration and DRY (Don't Repeat Yourself).

H. PYLONS



Pylons

Python Web Framework — Python Pylons

Python Web Framework — Python Pylons

Pylons is a lightweight Web framework aiming at flexibility and rapid development. With the best ideas from Ruby, Python, and Perl, it makes for a structured, but extremely flexible Python Web framework. With Pylons, Web development is fast, flexible, and easy. Pylons is built on top of Paste. But after being merged with Pyramid to form the Pylons project, it is in maintenance-only status.

I. REAHL

You can use Reahl to develop web applications in pure Python. However, you may use, customize, or compose widgets in usual Python code. These widgets portray certain server-side and client-side behaviors.

J. WHEEZY .WEB

Wheezy is a lightweight, high performance, and high concurrency WSGI web framework. Its key features include routing, model update/validation, authentication/authorization, content caching with dependency, middleware, and more. With these, we can build modern, efficient web.

K. ZOPE2



Python Web Framework — Python Zope

Python Web Framework — Python Zope

Zope2 is rightly the granddaddy of Python web frameworks, it has been a family of networks. It is a web framework and a general-purpose application server. Today, it is primarily used for CMS. We also have Zope3, which is a standalone framework and a collection of related libraries.



Python Web Framework — Tornado

Python Web Framework — Tornado

While Tornado isn't that famous, it is great with non-blocking I/O. You can scale it to handle tens of thousands of open connections. It makes for a perfect framework for long polling, WebSockets, and other usages needing a continuous connection. Officially, Tornado only supports Linux and BSD OS (Windows and Mac OS X- only for development). Tornado finds its origin in the FriendFeed project, which now belongs to Facebook.

NON-FULL-STACK FRAMEWORKS IN PYTHON

A Python non full-stack framework will provide the base application server. This either runs as its own independent process, upon Apache, or in other environments. Let's look at the most popular ones.

A. PYTHON BOTTLE

Bottle is a simple and fast microframework that you can use to create small Web applications. It provides request-dispatching routes with URL-parameter support, templates, key/value databases, and a built-in HTTP server. It also offers adapters for third-party WSGI/HTTP-server and template engines. This is all in a single file; there are no dependencies except the Python Standard Library.

B. CHERRY PY



Python Web Framework — Python CherryPy

It is a pythonic, object-oriented HTTP framework. A web application powered by CherryPy is a standalone Python application that embeds its own multi-threaded web server.

In a way, CherryPy is a way between the programmer and the problem. It also supports various web servers like Apache, IIS, and so. CherryPy will let you launch multiple HTTP servers at once.

C. PYTHON FLASK



Python Web Framework — Python Flask

Like we've said before, Flask is a microframework for Python. It includes a built-in development server, and unit-testing support. It is also fully Unicode-enabled with RESTful request-dispatching and WSGI compliance.

Learn: [Python Regular Expressions](#)

Flask will be useful when you want to develop small, simple applications. With it, you can operate your database however you like- using SQLAlchemy or whatever. A goof Flask example is it is used by LinkedIn and Pinterest.

D. HUG



Python Web Framework — Python Hug

Python Web Framework — Python Hug

Hug is among the fastest web frameworks for Python. With it, you can build APIs. It supports several API versions, automatic API documentation, and annotation-powered validation. It is built on top of another JSON framework, Falcon.

E. PYRAMID



Pyramid

Python Web Framework — pyramid-positive

Python Web Framework — pyramid-positive

Unlike a few that we discussed so far, Pyramid is a framework for large applications. It is flexible; a Pyramid web application starts from a single-file module, and evolves into an ambitious project. You can say that it makes real-world Web application development and deployment more fun, predictable, and productive. Actually, Pyramid is a Pylons project.

F. ALBATROSS

It is a small, flexible Python toolkit that lets you develop highly stateful Web applications. Albatross deploys to CGI, FastCGI, and ModPython servers.

G. CIRCUITS

Circuits are much like CherryPy, but is a highly efficient web framework to develop standalone multiprocess applications. It supports concurrency, asynchronous I/O components, and is event-driven.

H. FALCON



Falcon

Python Web Framework — Python Falcon

Python Web Framework — Python Falcon

A microframework for small applications, app backends, and higher-level frameworks, Falcon encourages to follow the concept of REST. It is among the fastest web frameworks for Python and is used by EMC, Hurricane Electric, OpenStack, Opera Software, Wargaming, and others.

I. GROWLER



Python Web Framework — Python Growler

Growler is built on top of asyncio, and is inspired by Connect and Express frameworks for Node.js. If you want ORM or templating, you must install it manually. It handles requests by passing through a middleware chain.

J. MOREPATH

MorePath is a flexible, model-driven web framework. It supports REST and focusses on reusability and extensibility.

K. PYCNIC

Pycnic is among the fastest web frameworks for Python for developing JSON APIs. The framework is object-oriented and optimized for JSON APIs. It only includes tools for creating Web APIs that leave a lighter footprint.

L. SANIC



Python Web Framework — Python Sanic

Sanic is a flask-like framework, but it is fast. It supports asynchronous request handlers, and makes code non-blocking and speedy.

So, this was all about Python Web Framework Tutorial. Hope you like our explanation.

Designing a RESTful Web API:

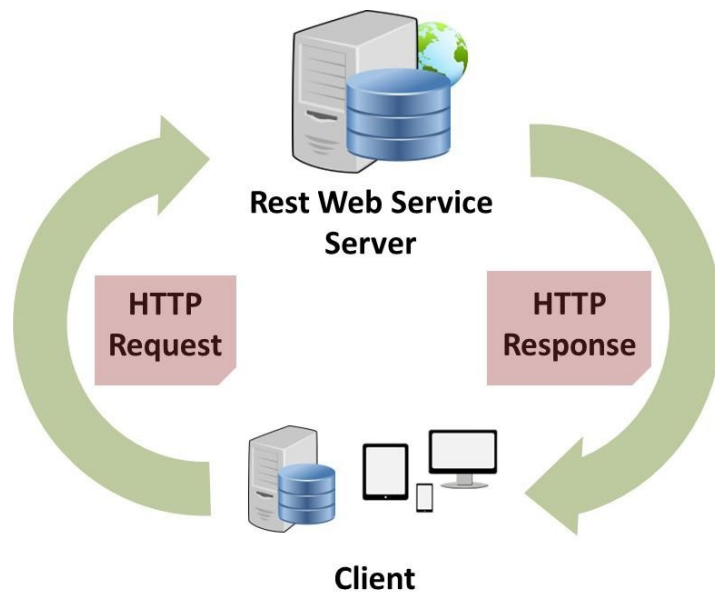
- Accept and respond with JSON
- Use nouns instead of verbs in endpoint paths
- Name collections with plural nouns
- Nest resources for hierarchical objects
- Handle errors gracefully and return standard error codes
- Allow filtering, sorting, and pagination
- Maintain good security practices

REST APIs are the most common APIs used across the web today. They provide simple, uniform interfaces that can be used to make data, content, algorithms, media, and other digital resources available through web URLs.

REST APIs should be stateless, which means that all client-server operations should be stateless, and any state management that is required should take place on the client, not the server.

REST APIs should also allow caching unless explicitly indicated that caching is not possible.

To secure REST APIs containing sensitive information, you can implement authentication mechanisms that control their exposure, mainly through user credentials and encrypted access codes.



Cloud Application Development in Python:

There are three main approaches to cloud application development in Python:

Platform as a Service (PaaS)

: PaaS provides a platform for developing, testing, and deploying cloud applications. It includes everything you need, including a programming language, development environment, and database.

Infrastructure as a Service (IaaS)

: IaaS provides you with the basic building blocks for cloud applications, such as compute, storage, and networking. You are responsible for setting up and managing your own environment.

Software as a Service (SaaS)

: SaaS provides you with access to software applications that are running in the cloud. You do not need to install or manage the software yourself.

The best approach for you will depend on your specific needs and requirements. If you are new to cloud development, PaaS is a good option because it provides you with everything you need to get started. If you have more experience, you may want to use IaaS or SaaS to give you more control over your environment.

Here are some additional tips for developing cloud applications in Python:

- Use a cloud-friendly programming language like Python.
- Take advantage of cloud-based tools and services.
- Design your application for scalability and reliability.
- Test your application thoroughly before deploying it to production.
- Monitor your application after deployment to ensure that it is performing as expected.

REST APIs are one of the most common kinds of web services available today. They allow various clients including browser apps to communicate with a server via the REST API. Therefore, it's very important to design REST APIs properly so that we won't run into problems down the road. We have to take into account [security](#), performance, and ease of use for API consumers.

Otherwise, we create problems for clients that use our APIs, which isn't pleasant and detracts people from using our API. If we don't follow commonly accepted conventions, then we confuse the maintainers of the API and the clients that use them since it's different from what everyone expects.

In this article, we'll look at how to design REST APIs to be easy to understand for anyone consuming them, future-proof, and secure and fast since they serve data to clients that may be confidential.

- [Accept and respond with JSON](#)
- [Use nouns instead of verbs in endpoint paths](#)
- [Name collections with plural nouns](#)
- [Nesting resources for hierarchical objects](#)
- [Handle errors gracefully and return standard error codes](#)
- [Allow filtering, sorting, and pagination](#)
- [Maintain Good Security Practices](#)

- [Cache data to improve performance](#)
- [Versioning our APIs](#)

WHAT IS A REST API?

A REST API is an application programming interface that conforms to specific architectural constraints, like stateless communication and cacheable data. It is not a protocol or standard. While REST APIs can be accessed through a number of communication protocols, most commonly, they are called over HTTPS, so the guidelines below apply to REST API endpoints that will be called over the internet.

Note: For REST APIs called over the internet, you'll like want to follow the [best practices for REST API authentication](#).

ACCEPT AND RESPOND WITH JSON

REST APIs should accept JSON for request payload and also send responses to JSON. JSON is the standard for transferring data. Almost every networked technology can use it: JavaScript has built-in methods to encode and decode JSON either through the Fetch API or another HTTP client. Server-side technologies have libraries that can decode JSON without doing much work.

There are other ways to transfer data. XML isn't widely supported by frameworks without transforming the data ourselves to something that can be used, and that's usually JSON. We can't manipulate this data as easily on the client-side, especially in browsers. It ends up being a lot of extra work just to do normal data transfer.

Form data is good for sending data, especially if we want to send files. But for text and numbers, we don't need form data to transfer those since—with most frameworks—we can transfer JSON by just getting the data from it directly on the client side. It's by far the most straightforward to do so.

To make sure that when our REST API app responds with JSON that clients interpret it as such, we should set Content-Type in the response header to application/json after the request is made. Many server-side app frameworks set the response header automatically. Some HTTP clients look at the Content-Type response header and parse the data according to that format.

The only exception is if we're trying to send and receive files between client and server. Then we need to handle file responses and send form data from client to server. But that is a topic for another time.

We should also make sure that our endpoints return JSON as a response. Many server-side frameworks have this as a built-in feature.

USE NOUNS INSTEAD OF VERBS IN ENDPOINT PATHS

We shouldn't use verbs in our endpoint paths. Instead, we should use the nouns which represent the entity that the endpoint that we're retrieving or manipulating as the pathname.

This is because our HTTP request method already has the verb. Having verbs in our API endpoint paths isn't useful and it makes it unnecessarily long since it doesn't convey any new information. The chosen verbs could vary by the developer's whim. For instance, some like 'get' and some like 'retrieve', so it's just better to let the HTTP GET verb tell us what and endpoint does.

The action should be indicated by the HTTP request method that we're making. The most common methods include GET, POST, PUT, and DELETE.

- GET retrieves resources.
- POST submits new data to the server.
- PUT updates existing data.
- DELETE removes data.

RESTful APIs' ROLE IN CLOUD SERVICES

RESTful APIs can be looked on as the glue that connects the cloud service providers and cloud service consumers. For example, application developers requiring to display a weather forecast can consume the Google Weather API. In this section, we will look at the applicability of RESTful APIs for provisioning resources in the cloud.

For an illustration of RESTful APIs, we will be using the Oracle Cloud service platform. Users can set up a free trial account via <https://Cloud.oracle.com/home> and try out the examples discussed in the following sections.

The screenshot displays the Oracle Cloud 'Compute Classic' interface, specifically the 'Instances' tab. At the top, there's a navigation bar with 'Compute Classic' and tabs for 'Instances', 'Network', 'Storage', 'Orchestrations', 'Images', and a 'Visualization' button. On the left, a sidebar shows 'Instances' and 'Instance Snapshots'. The main area features a 'Summary' section with large numbers: 2 instances, 2 OCPUs, and 15GB memory. Below this is a description of instances and a 'Learn more' link. A search bar and filters (Category: All, Show: All) are present above a table of instances. The table has columns for Name, Status, OCPUs, Memory, Volumes, Public IP, and Private IP. Two instances are listed: 'test-vm-1' and 'test-vm-2', both with a status of 'Running'. A 'Create Instance' button is located at the bottom right of the table area.

Name	Status	OCPUs	Memory	Volumes	Public IP	Private IP
test-vm-1	Running	1	7.5 GB			10.16.113.254
test-vm-2	Running	1	7.5 GB			10.16.191.182

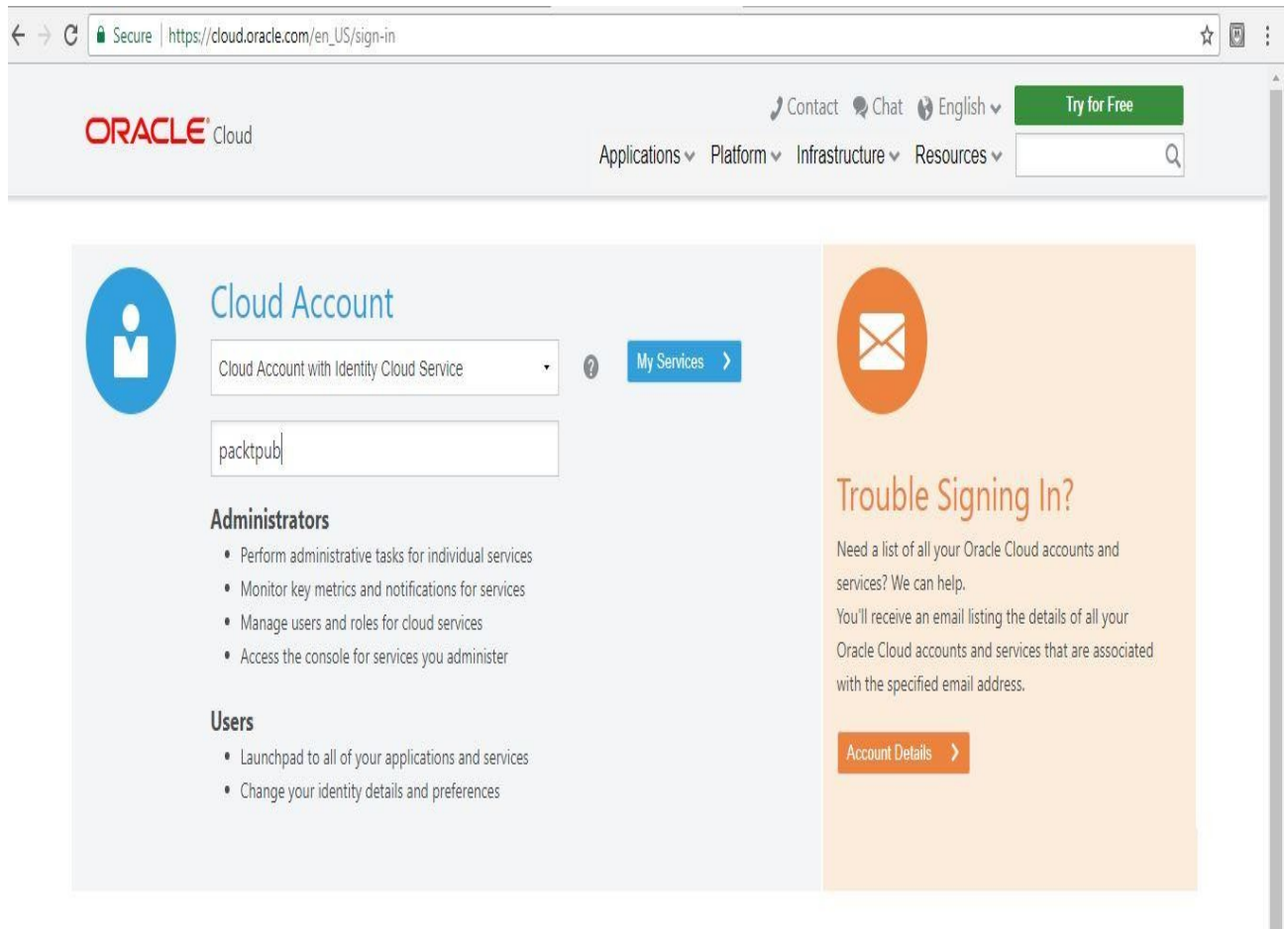
For example, we will try to set up a test virtual machine instance using the REST APIs. The high-level steps required to be performed are as follows:

1. Locate REST API endpoint
2. Generate authentication cookie
3. Provision virtual machine instance

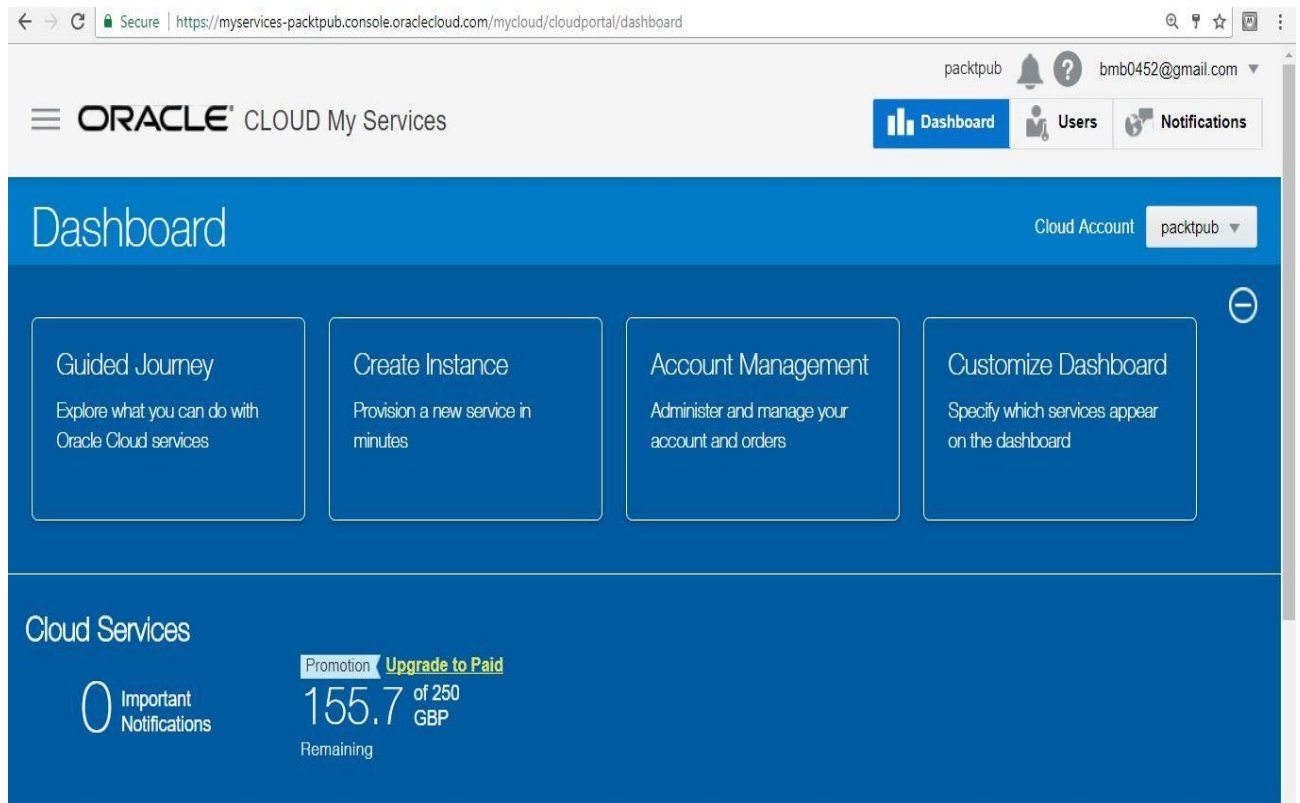
Locating the REST API endpoint

Once users have signed up for an Oracle Cloud account, they can locate the REST API endpoint to be used by navigating via the following steps:

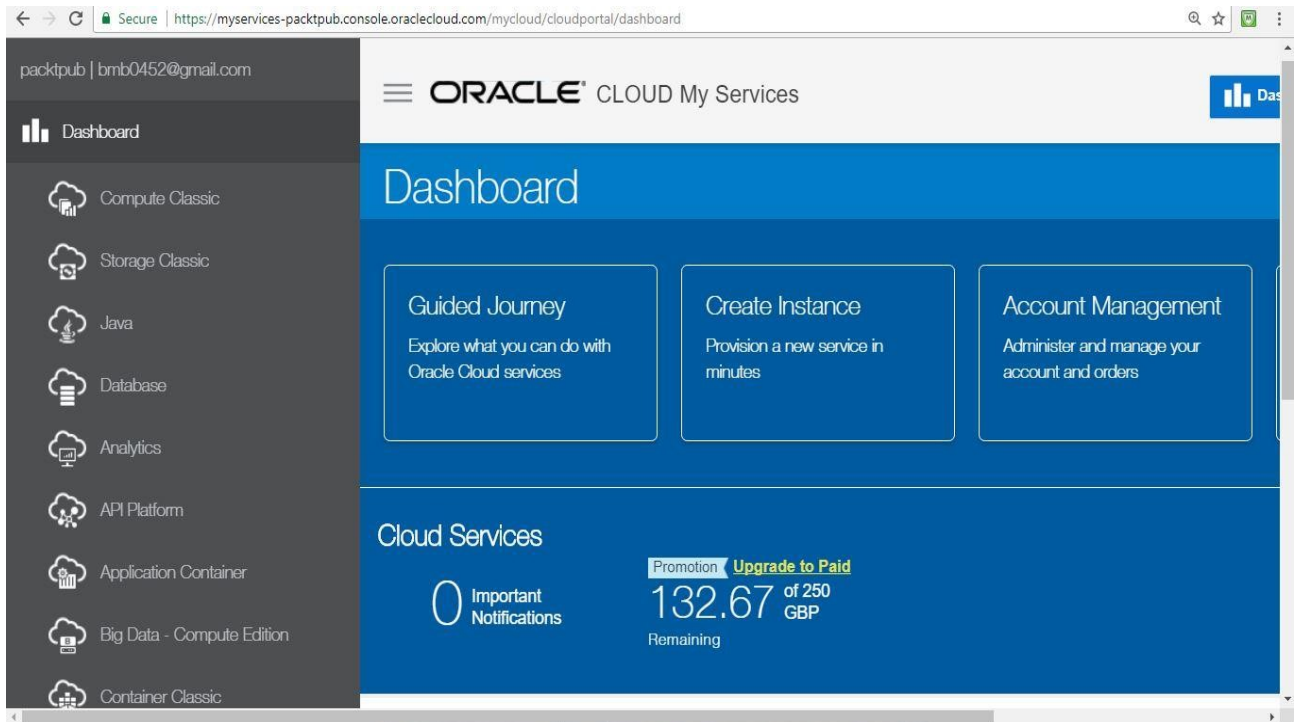
1. Login screen: Choose the relevant Cloud Account details and click the My Services button as shown in the screenshot ahead:



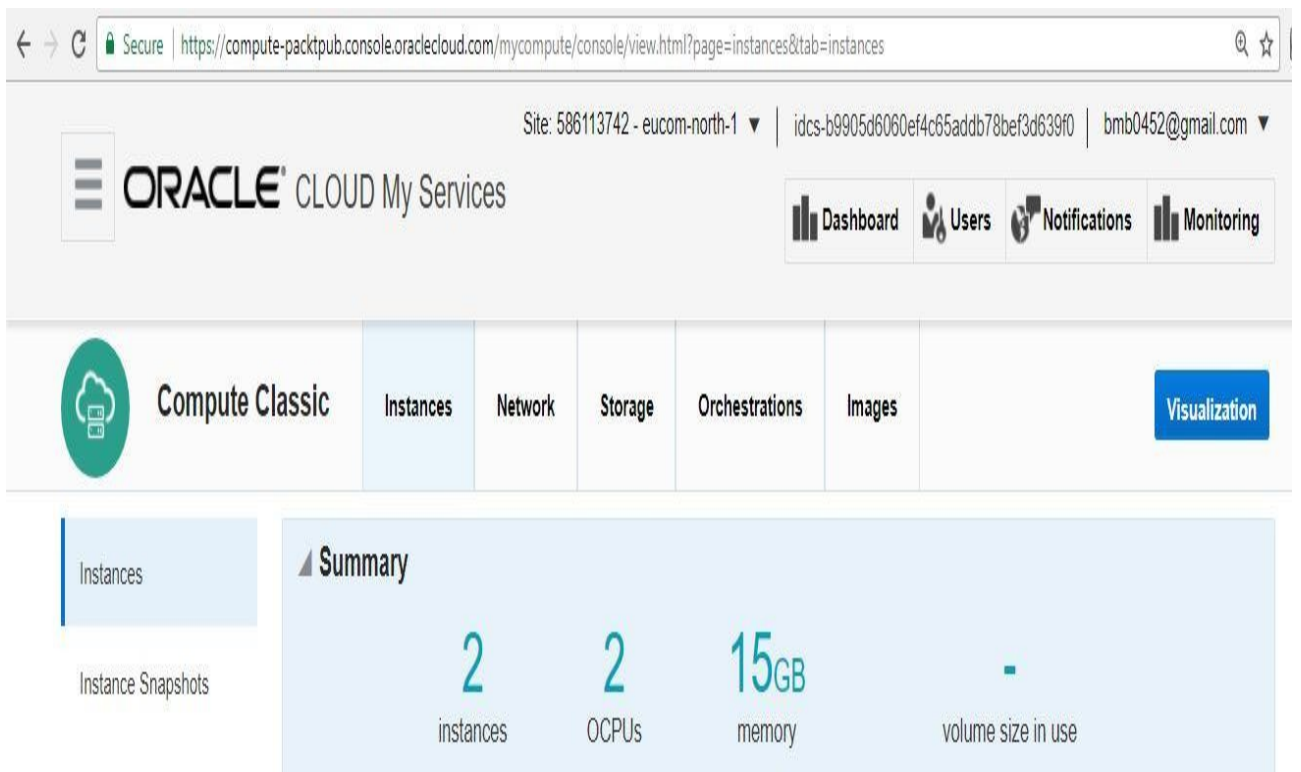
2. Home page: Displays the cloud services Dashboard for the user. Click the Dashboard icon as shown in the following screenshot:



3. Dashboard screen: Lists the various cloud offerings. Click the Compute Classic offering:



4. Compute Classic screen: Displays the details of infrastructure resources utilized by the user:



5. Site Selector screen: Displays the REST endpoint:

Site Selector

Service Instance ID: 586113742

? Site: eucom-north-1

Data Center: nldc1

REST Endpoint: https://compute.eucom-north-1.oraclecloud.com/

Site Usage

Resource	Usage
OCPU	13.4%
Memory	9.8%
IP Reservations	4.1%

Ok Cancel

Generating an authentication cookie

Authentication is required for provisioning the IT resources. For this purpose, we will be required to generate an authentication cookie using the Authenticate User REST API. The details of the API are as follows:

API details
Description API function Authenticate supplied user credential and generate authentication cookie for use in the subsequent API calls.
Endpoint <REST Endpoint captured in previous section>/authenticate/

Example: <https://compute.eucom-north-1.oracleCloud.com/authenticate/>

HTTP method

POST

Request header properties

Content-Type: application/oracle-compute-v3+json Accept: application/oracle-compute-v3+json Request body

- **user:** Two-part name of the user in the format /Computeidentity_domain/user
- **password:** Password for the specified userSample
- **request:** {
"password": "xxxxx",
"user": "/Compute-586113456/test@gmail.com"
}

Response header properties

set-cookie: Authentication cookie value

The following screenshot shows the authentication cookie generated by invoking the Authenticate User REST API via the Postman tool:

POST <https://compute.eucom-north-1.oraclecloud.com/launchplan/> Params Send Save

Authorization Headers (3) Body Pre-request Script Tests Cookies Code

form-data x-www-form-urlencoded raw binary Text

```

1 {
2   "instances": [
3     {
4       "shape": "oc3",
5       "image_list": "/oracle/public/oe1_6.4_2GB_v1",
6       "name": "/Compute-586113742/bmb0452@gmail.com/test-vm-2",
7       "label": "test-vm-2",
8       "sshkeys": []
9     }
10  ]
11 }

```

Body Cookies (1) Headers (13) Tests Status: 201 Created Time: 2022 ms Size: 3.74 KB

Pretty Raw Preview JSON

```

1 {
2   "relationships": [],
3   "instances": [
4     {
5       "domain": "compute-586113742.oraclecloud.internal.",
6       "placement_requirements": [
7         "/system/compute/pool/general",
8         "/system/compute/allow_instances"
9       ],
10      "ip": "0.0.0.0",
11      "fingerprint": "",
12      "image_metadata_bag": null,
13      "site": "",
14      "last_state_change_time": null

```

HTTP Response Status 201 confirms the request for provisioning was successful. Check the provisioned instance status via the cloud service instances page as shown here:

Compute Classic Instances Network Storage Orchestration Images Visualization

Instances

Instance Snapshots

Summary

2 instances 2 OCPUs 15GB memory - volume size in use

Instances

A Compute Classic instance is a virtual machine running a specific operating system, with the CPU and memory resources that you specify. [Learn more.](#)

Category: All Show: All Create Instance

Name	Status	OCPUs	Memory	Volumes	Public IP	Private IP
test-vm-1	Running	1	7.5 GB			10.16.113.254
test-vm-2	Running	1	7.5 GB			10.16.191.182

CLOUD APPLICATION DEVELOPMENT IN PYTHON

DEPLOYING A PYTHON APP

- [Before you begin](#)
- [Installing the gcloud CLI](#)
- [Using a proxy](#)
- [Deploying an app](#)
- [Deploying multiple service applications](#)
- [Viewing build logs](#)
- [Updating indexes](#)
- [Troubleshooting](#)

Deploy your app to upload and run it on App Engine. When you deploy your apps, you create versions of those apps and their corresponding [services](#) in App Engine. You can deploy entire apps, including all the source code and configuration files, or you can deploy and update individual versions or [configuration files](#).

To programmatically deploy your apps, [use the Admin API](#).

BEFORE YOU BEGIN

Before you can deploy your app:

- The [Owner](#) of the Cloud project must create the [App Engine application](#).
- Ensure that your user account includes the [required privileges](#).
- [Give Cloud Build permission to deploy apps](#) in your project. When you deploy your app, App Engine uses Cloud Build to build the app into a container and deploy the container to the runtime in the app's region. Cloud Build does not have permission to deploy Python 2 apps by default, so you need to give permission before you can deploy apps.

INSTALLING THE GLOUD CLI

To deploy your app with the gcloud CLI, you must download, install, and initialize the [gcloud CLI](#).
[Download the SDK](#)

If you already have the gcloud CLI installed and want to configure it to use a Cloud project ID other than the one that you initialized it to, see [Managing gcloud CLI Configurations](#).

USING A PROXY

If you are running the deployment command from a system which uses an HTTP or HTTPS proxy, you must configure the tool so that it can communicate via the proxy.

Run the following commands to configure the gcloud CLI:

```
gcloud config set proxy/type [PROXY_TYPE]
gcloud config set proxy/address [PROXY_ADDRESS]
gcloud config set proxy/port [PROXY_PORT]
```

You can also set a username and password for the proxy. For more information, see [gcloud config](#).

DEPLOYING AN APP

To deploy your app to App Engine, use the gcloud app deploy command from where your [configuration files](#) are located, for example app.yaml.

```
gcloud app deploy [CONFIGURATION_FILES]
```

By default, the command deploys the app.yaml configuration file from the current directory. If you're running the command from a directory that does not contain your app's app.yaml, or if you want to [deploy multiple apps](#), replace [CONFIGURATION_FILES] with the path to one or more configuration files. Use a single white space to separate pathnames.

Optional flags:

- --version: Specifies a [custom version ID](#). By default, App Engine generates a version ID.

- `--no-promote`: Deploys your app without automatically routing all traffic to that version. By default, each version that you deploy is automatically configured to receive 100% of traffic.
- `--project`: Specifies an alternate Cloud project ID to what you initialized as the default in the `gcloud` CLI.
- For more information, see the [gcloud app deploy reference](#) or run `gcloud help` from the command line.

Examples:

```
gcloud app deploy
```

```
gcloud app deploy app.yaml dos.yaml index.yaml
```

```
gcloud app deploy --version [YOUR_VERSION_ID] --no-promote --project [YOUR_PROJECT_ID]
```

If you deploy a version that specifies the same version ID as a version that already exists on App Engine, the files that you deploy will overwrite the existing version. This can be problematic if the version is serving traffic because traffic to your application might be disrupted. You can avoid disrupting traffic if you deploy your new version with a different version ID and then move traffic to that version.

DEPLOYING MULTIPLE SERVICE APPLICATIONS

When your application is factored into multiple [services](#), you can deploy and update individually targeted services or all the services simultaneously. Deploying updates to services can include updating individual [configuration files](#) or updating the source code in the corresponding versions.

For example, you can deploy and create two versions in App Engine, where each version runs in their own service. The first version serves as the frontend service and the other as the backend of your app. You can then deploy individual configuration files to update only the settings of a service. You can also choose to deploy a new version to a service in order to update the source code of the frontend, backend, or both simultaneously.

Requirements for multiple services

- You use the same deployment commands for deploying and updating the multiple services of your application with the following requirements:
- You must initially deploy a version of your app to the default service before you can create and deploy subsequent services.
- You must specify the ID of your service in the [app.yaml](#) configuration file of the corresponding version. To specify the service ID, you include the `service: [YOUR_SERVICE_ID]` element definition in each configuration file. By default, excluding this element definition from your configuration file deploys the version to the default service.
- You must specify all the corresponding `app.yaml` configuration files in your deployment command to simultaneously deploy multiple services.

To deploy multiple services

From the root directory of the application where the configuration files are located, you run the deployment command and specify the relative paths and file names for each service's `app.yaml` file.

```
gcloud app deploy [CONFIGURATION_FILES]
```

Where `[CONFIGURATION_FILES]` is one or more configuration file's path and name separated by a single whitespace.

Example

```
gcloud app deploy main/app.yaml service1/app.yaml service2/app.yaml
```

You will receive verification via the command line as each service is successfully deployed.

VIEWING BUILD LOGS

[Cloud Build](#) streams build and deploy logs that are viewable in the [Cloud Build history section of the Google Cloud console](#). To view builds in the app's region, use the **Region** drop-down menu at the top of the page to choose the region you would like to filter by.

UPDATING INDEXES

To create or update the indexes that your apps use, upload the `index.yaml` configuration file to Datastore. Indexes that don't exist yet are created after that configuration file is uploaded.

It can take a while for Datastore to create all the indexes and therefore, those indexes won't be immediately available to App Engine. If your app is already configured to receive traffic, then exceptions can occur for queries that require an index which is still in the process of being built.

To avoid exceptions, you must allow time for all the indexes to build, for example:

- Upload the `index.yaml` configuration file to Datastore before you deploy your version:

1. Upload the index.yaml file to Datastore:


```
<pre class="prettyprint lang-sh">gcloud datastore indexes create index.yaml</pre>
```

 For information, see the [gcloud datastore](/sdk/gcloud/reference/datastore/) reference.
 2. Use the Google Cloud console to monitor the status of all your indexes: [Go to the Datastore page](#)
 3. After all your indexes are built, [deploy the new version to App Engine](#).
 - Build your indexes before migrating or splitting traffic to your version:
 1. Deploy the new version without routing traffic to that version: You must specify both the app.yaml and index.yaml files and also include the --no-promote flag so that no traffic is routed to the version:


```
gcloud app deploy app.yaml index.yaml --no-promote
```
 2. Use the Google Cloud console to monitor the status of all your indexes: [Go to the Datastore page](#)
 3. After all your indexes are built, use the Google Cloud console to migrate or split traffic to your version: [Go to the Versions page](#)
- For more information about indexes, see [Configuring Datastore Indexes](#).
- ### TROUBLESHOOTING
- The following are common error messages that you might encounter:
- PERMISSION_DENIED: Operation not allowed**
 The "appengine.applications.create" permission is required.
 If the Cloud project does not include the [required App Engine application](#), the gcloud app deploy command can fail when it tries to run the gcloud app create command. Only accounts with Owner role have the necessary permissions to create App Engine applications.
- Command not found**
 If you did not create symlinks for the dev_appserver.sh tools when you installed the ([deprecated](#)) App Engine SDK, you might need to specify the full directory path to run the tool, for example: [PATH_TO_CLOUD_SDK]/bin/dev_appserver.py.
- Import Error**
 If you installed both the gcloud CLI as well as the original App Engine SDK, the entries to your PATH might conflict with one another and cause import errors. If you received errors when running gcloud CLI commands, try explicitly using the original App Engine SDK. You can move the entry for the original App Engine SDK to earlier in your PATH so that those commands have priority. Alternatively, you can run the command by specifying the full directory path: [PATH_TO_APP_ENGINE_SDK]/dev_appserver.py.
 On Linux or Mac, you can run **which dev_appserver.py** to determine which SDK is first in your PATH.
- [400] The first service (module) you upload to a new application must be the 'default' service (module)
 Before you can deploy and create the multiple services of your application, you must first deploy and create the default [service](#). For details about how to deploy a version to the default service, see [Deploying multiple service applications](#).
- Too Many Versions (403)**
 App Engine has a [limit on the number of deployed versions](#) of your application. These differ for free applications and deployed applications. You can use the [Google Cloud console](#) to delete an older version and then upload your latest code.
- [13] An internal error occurred while creating a Cloud Storage bucket.
 App Engine [creates a default Cloud Storage multi-regional bucket](#) on your behalf, on the same region where your application is created. This bucket is required to store the contents of your application. This error is returned when this bucket cannot be created, in the following scenarios:
- The default [App Engine service account](#) is not present in your project. If your account was removed before 30 days elapsed since its deletion, you can [restore it](#).
 - Your project is under an organization enforcing the [constraints/gcp.resourceLocations policy](#), and the organization is not allowing the creation of resources on the same [region where your App Engine](#) was created. You will need to [override the enforced constraints/gcp.resourceLocations policy for your project](#), and allow the [multi-region locations](#) on the same region where your App Engine app is created.
- Note:** For example, if your App Engine app is created on the **europe-west** region, even though the region maps to the [europe-west1 locations](#), you will have to modify the constraint to allow resources in the **in:eu-locations**, which includes all the EU regional. If your App Engine application is created on **US** and **ASIA** regions, you would have to allow **in:us-locations** or **in:asia-locations**, respectively.
- [13] An internal error occurred

This error can occur if the App Engine's `app.yaml` configuration file contains an invalid resource name under the `vpc_access_connector` key. Make sure that the [name field](#) contains the correct project and region where your Serverless VPC Access connector is created.

If the issue persists after ensuring your `app.yaml` configuration is valid, use the Google Cloud SDK to re-deploy your service, adding the `--verbosity=debug flag`, and contact [GCP Support](#) by providing the command's output.

Other deployment error

If your deployment fails, make sure the Cloud Build API is [enabled in your project](#). App Engine enables this API automatically the first time you deploy an app, but if someone has since disabled the API, deployments will fail

DESIGN APPROACHES

When we consider class design, we often have a built-in or library class which does some of the job we want. For example, we want to be able to accumulate a list of values and then determine the average: this is a very list-like behavior, extended with a new feature.

There are two overall approaches for extending a class: *wrapping* and *inheritance*.

- Wrap an existing class (for example, a tuple, list, set or map) in a new class which adds features. This allows you to redefine the interface to the existing class, which often involves removing features.
- Inherit from an existing class, adding features as part of the more specialized subclass. This may require you to read more of the original class documentation to see a little of how it works internally.

Both techniques work extremely well; there isn't a profound reason for making a particular choice. When wrapping a collection, you can provide a new, focused interface on the original collection; this allows you to narrow the choices for the user of the class. When subclassing, however, you often have a lot of capabilities in the original class you are extending.

"Duck" Typing. In [the section called "Polymorphism"](#), we mentioned "Duck" Typing. In Python, two classes are practically polymorphic if they have the same interface methods. They do not have to be subclasses of the same class or interface (which is the rule in Java.)

This principle means that the distinction between wrapping and inheritance is more subtle in Python than in other languages. If you provide all of the appropriate interface methods to a class, it behaves as if it was a proper subclass. It may be a class that is wrapped by another class that provides the same interface.

For example, say we have a class `Dice`, which models a set of individual `Die` objects.

```
class Dice( object ):
    def __init__( self ):
        self.theDice= [ Die(), Die() ]
    def roll( self ):
        for d in self.theDice:
            d.roll()
        return self.theDice
```

In essence, our class is a wrapper around a list of dice, named `theDice`. However, we don't provide any of the interface methods that are parts of the built-in list class.

Even though this class is a wrapper around a list object, we can add method names based on the built-in list class: `append`, `extend`, `count`, `insert`, etc. We'll look closely at this in [Chapter 24, Creating or Extending Data Types](#).

```
class Dice( object ):
    def __init__( self ):
        self.theDice= [ Die(), Die() ]
    def roll( self ):
        for d in self.theDice:
            d.roll()
        return self.theDice
    def append( self, aDie ):
        self.theDice.append( aDie )
    def __len__( self ):
        return len( self.theDice )
```

Once we've defined these list-like functions we have an ambiguous situation.

- We could have a subclass of list, which initializes itself to two `Die` objects and has a `roll` method.
- We could have a distinct `Dice` class, which provides a `roll` method and a number of other methods that make it look like a list.

For people who will read your Python, clarity is the most important feature of the program. In making design decisions, one of your first questions has to be "what is the real thing that I'm modeling?" Since many alternatives will work, your design should reflect something that clarifies the problem you're solving.

APPLICATIONS OF DIGITAL IMAGE PROCESSING

Almost in every field, digital image processing puts a live effect on things and is growing with time to time and with new technologies.

1) IMAGE SHARPENING AND RESTORATION

It refers to the process in which we can modify the look and feel of an image. It basically manipulates the images and achieves the desired output. It includes conversion, sharpening, blurring, detecting edges, retrieval, and recognition of images.

2) MEDICAL FIELD

There are several applications under medical field which depends on the functioning of digital image processing.

- Gamma-ray imaging
- PET scan
- X-Ray Imaging
- Medical CT scan
- UV imaging

3) ROBOT VISION

There are several robotic machines which work on the digital image processing. Through image processing technique robot finds their ways, for example, hurdle detection robot and line follower robot.

4) PATTERN RECOGNITION

It involves the study of image processing, it is also combined with artificial intelligence such that computer-aided diagnosis, handwriting recognition and images recognition can be easily implemented. Now a days, image processing is used for pattern recognition.

5) VIDEO PROCESSING

It is also one of the applications of digital image processing. A collection of frames or pictures are arranged in such a way that it makes the fast movement of pictures. It involves frame rate conversion, motion detection, reduction of noise and colour space conversion etc.

Document storage App:

Some cloud storage services, such as **Apple iCloud, Google Drive and Microsoft OneDrive**, are generalists, offering not only folder and file syncing, but also media-playing and device syncing. These products even double as collaboration software, offering real-time document co-editing.

OUR TOP TESTED PRODUCTS



Microsoft OneDrive
Best for Windows Users



IDrive
Best for Low-Cost Backup and Syncing



Google Drive
Best for Google Workspace Users



Dropbox
Best for Integration With Third-Party Services



SpiderOak One Backup
Best for Secure Backups



Box (Personal)
Best for Business Integrations



Apple iCloud Drive
Best for Mac and iPhone users

Application Of MapReduce

Entertainment: To discover the most popular movies, based on what you like and what you watched in this case Hadoop MapReduce help you out. It mainly focuses on their logs and clicks.

E-commerce: Numerous E-commerce suppliers, like Amazon, Walmart, and eBay, utilize the MapReduce programming model to distinguish most loved items dependent on clients' inclinations or purchasing behavior.

It incorporates making item proposal Mechanisms for E-commerce inventories, examining website records, buy history, user interaction logs, etc.

Data Warehouse: We can utilize MapReduce to analyze large data volumes in data warehouses while implementing specific business logic for data insights.

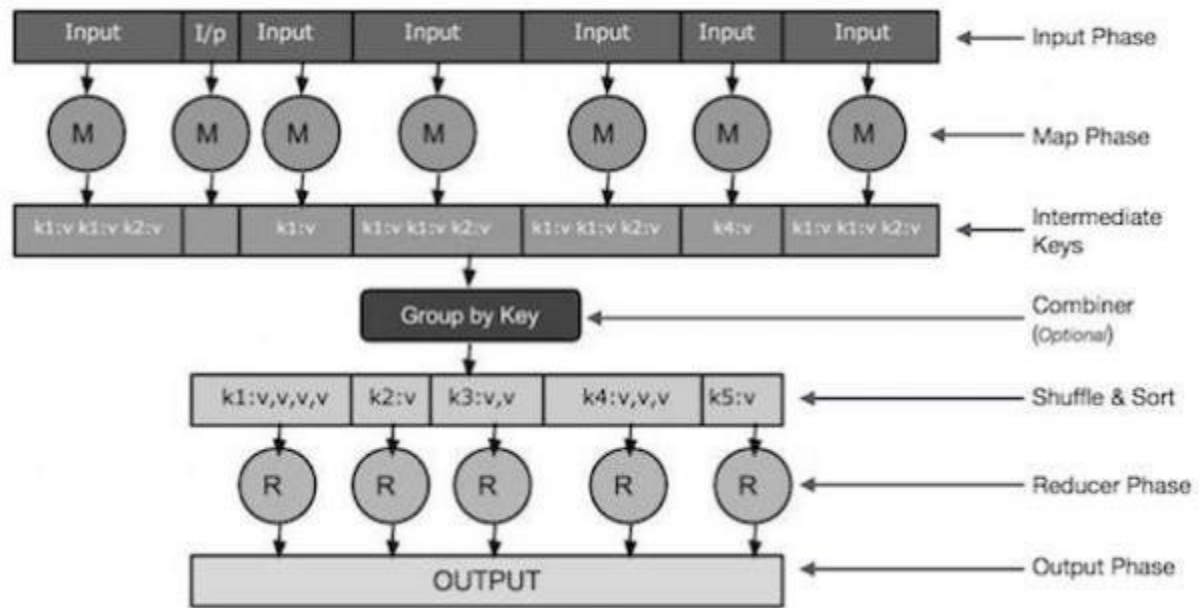
Fraud Detection: Hadoop and MapReduce are utilized in monetary enterprises, including organizations like banks, insurance providers, installment areas for misrepresentation recognition, pattern distinguishing proof, or business metrics through transaction analysis.

HOW DOES MAPREDUCE WORKS?

The MapReduce algorithm contains two important tasks, namely Map and Reduce.

- The Map task takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key-value pairs).
- The Reduce task takes the output from the Map as an input and combines those data tuples (key-value pairs) into a smaller set of tuples.

The reduced task is always performed after the map job.



Input Phase – Here we have a Record Reader that translates each record in an input file and sends the parsed data to the mapper in the form of key-value pairs.

Map – Map is a user-defined function, which takes a series of key-value pairs and processes each one of them to generate zero or more key-value pairs.

Intermediate Keys – The key-value pairs generated by the mapper are known as intermediate keys.

Combiner – A combiner is a type of local Reducer that groups similar data from the map phase into identifiable sets. It takes the intermediate keys from the mapper as input and applies a user-defined code to aggregate the values in a small scope of one mapper. It is not a part of the main MapReduce algorithm; it is optional.

Shuffle and Sort – The Reducer task starts with the Shuffle and Sort step. It downloads the grouped key-value pairs onto the local machine, where the Reducer is running. The individual key-value pairs are sorted by key into a larger data list. The data list groups the equivalent keys together so that their values can be iterated easily in the Reducer task.

Reducer – The Reducer takes the grouped key-value paired data as input and runs a Reducer function on each one of them. Here, the data can be aggregated, filtered, and combined in a number of ways, and it requires a wide range of processing. Once the execution is over, it gives zero or more key-value pairs to the final step.

Output Phase – In the output phase, we have an output formatter that translates the final key-value pairs from the Reducer function and writes them onto a file using a record writer.

ADVANTAGE OF MAPREDUCE

Fault tolerance: It can handle failures without downtime.

Speed: It splits, shuffles, and reduces the unstructured data in a short time.

Cost-effective: Hadoop MapReduce has a scale-out feature that enables users to process or store the data in a cost-effective manner.

Scalability: It provides a highly scalable framework. MapReduce allows users to run applications from many nodes.

Parallel Processing: Here multiple job-parts of the same dataset can be processed in a parallel manner. This can reduce the task that can be taken to complete a task.

LIMITATIONS OF MAPREDUCE

- MapReduce cannot cache the intermediate data in memory for a further requirement which diminishes the performance of Hadoop.
- It is only suitable for Batch Processing of a Huge amounts of Data.

10 SOCIAL MEDIA ANALYTICS TOOLS CAN HELP YOU TRACK YOUR SOCIAL PRESENCE

1. [Sprout Social](#)
2. [HubSpot](#)
3. [TapInfluence](#)
4. [BuzzSumo](#)
5. [Snaplytics](#)
6. [Curalate](#)
7. [Keyhole](#)
8. [Google Analytics](#)
9. [ShortStack](#)
10. [SHIELDApp](#)