LAB – 2: Introduction to AWS EC2 (Elastic Compute Cloud)

In this lab, you will learn :

- Launching an Amazon EC2 instance
- Monitoring an Amazon EC2 instance.
- Updating security group of instance and accessing web server
- Resizing an instance
- Stopping & terminating an instance

Introduction to AWS Educate:

AWS Educate is an online platform built by Amazon that enables users to learn AWS by providing access to online training resources and labs to learn, practice, and evaluate cloud skills without having to create an Amazon or AWS account. In this course, we will be working with AWS Educate, which familiarizes you with AWS.

Setting up an AWS Educate account :

1. Click here to go to AWS Educate.
2. Click on “Register Now”
3. Provide your **SRN** as the First Name and your **full name** as the last name while filling the required details to register.
4. Verify the given email address to complete the registration.
5. Set a password for the AWS Educate account
6. Login into your account and choose the course **“Getting Started with Compute Lab”** (https://awseducate.instructure.com/courses/907)
7. Go to the modules tab and click on the button “Load Getting Started with Compute Lab in a new window”
8. Click on “Start Lab”. This will provision your resources for the lab; might take about 5 minutes.
9. Follow the instructions given in the lab. The deliverables you need to submit are mentioned below.
10. Explore the course!

What is AWS EC2?

The AWS EC2 (Elastic Compute) service is one of the most essential services. This offers the actual computation for your cloud apps and is as scalable as any cloud service should be.

Amazon provides various types of instances with different configurations of CPU, memory, storage, and networking resources to suit user needs. Each type is available in various sizes to address specific workload requirements.

Some essential features of EC2 are:

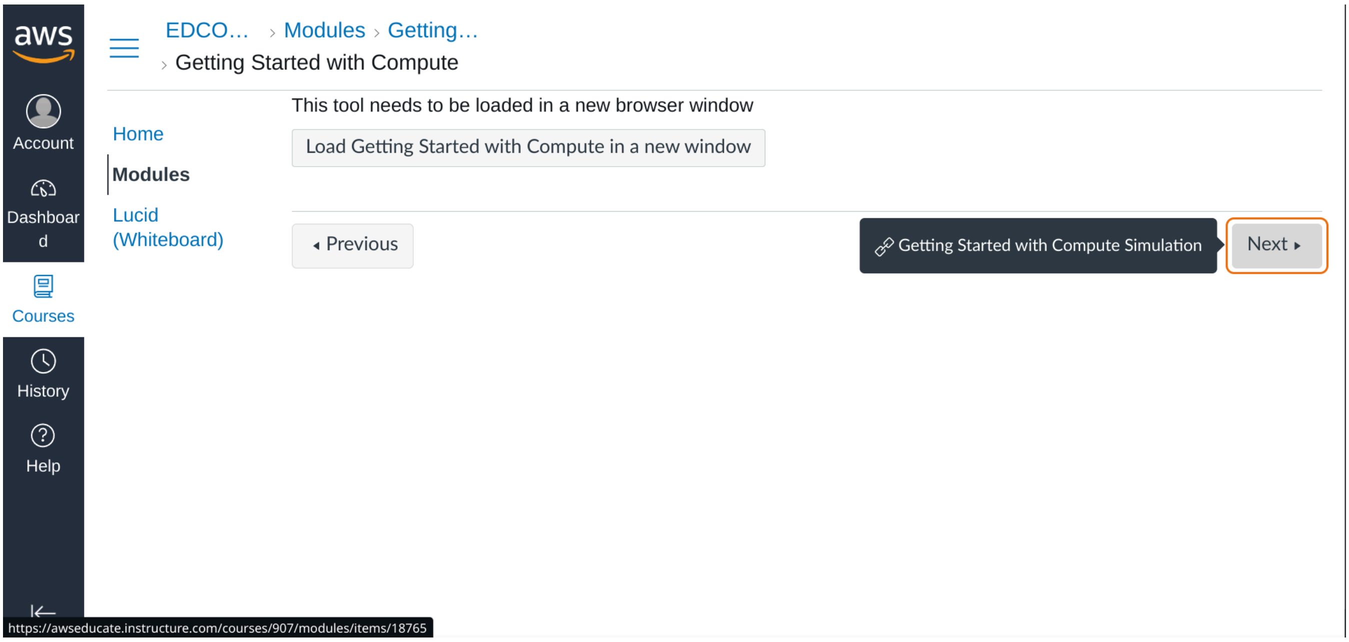
- EC2 instances are on-demand that are reliable and scalable infrastructures with the ability to increase and decrease the capacity within minutes.
- Configurable CPUs, memory storage, and networking capacity are called the instance types, to run your applications or software on the instance.
- Instance store volumes for temporary data that are deleted when you stop or terminate your instance.
- Amazon EC2 infrastructure is programmable and you can use scripts to automate the deployment process, install and configure.
- Public and private key pairs for secure login into instances.

Each EC2 instance has the following mandatory configuration requirements:

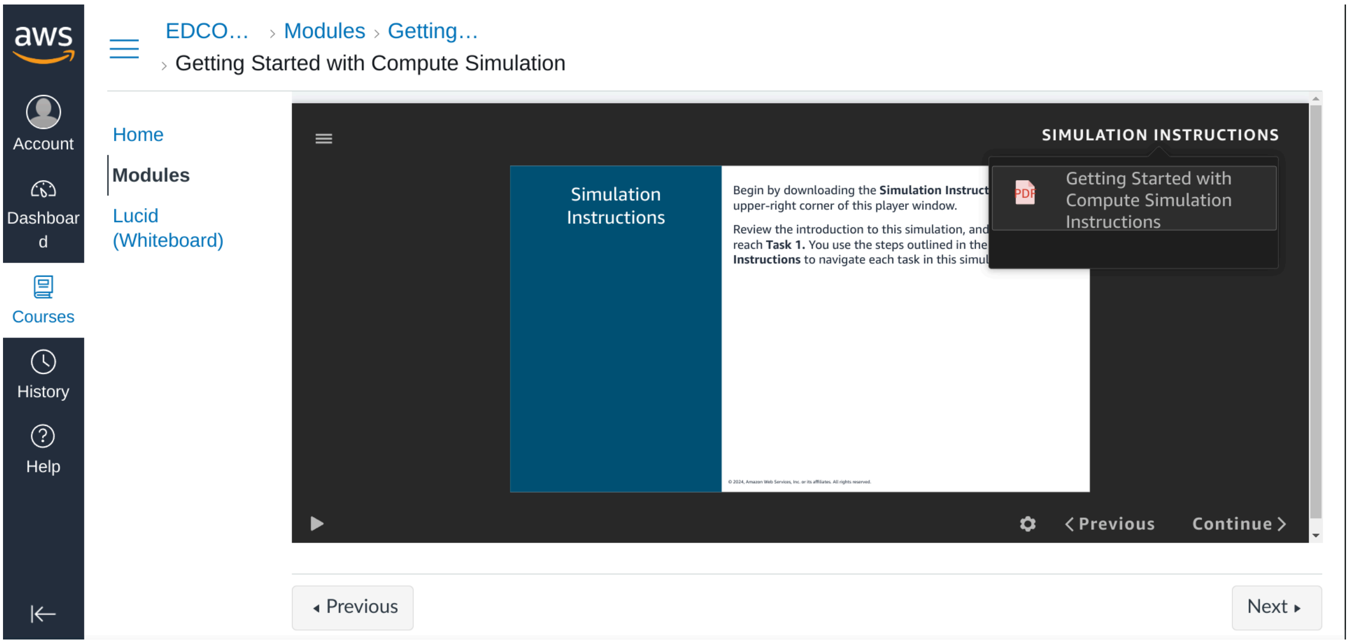
- Amazon Machine Image(AMI)
- Instance Type (The Instance Type usually depends on the use of the VM)
- Specific Instance Details such as network, subnets, start up scripts etc.
- Security Groups - These are essentially the firewalls to your instance, they control the access to your instance.

STEPS

1) Once you take the survey(optional) you will land on this page where you can click on **Next** to start the simulation. You could also click on **Load Getting Started with Compute in a new window** button to watch the videos explaining EC2.


aws
Account
Dashboard
Courses
History
Help
EDCO… › Modules › Getting…
› Getting Started with Compute
Home
Modules
Lucid (Whiteboard)
This tool needs to be loaded in a new browser window
Load Getting Started with Compute in a new window
◂ Previous
Getting Started with Compute Simulation
Next ▸
https://awseducate.instructure.com/courses/907/modules/items/18765

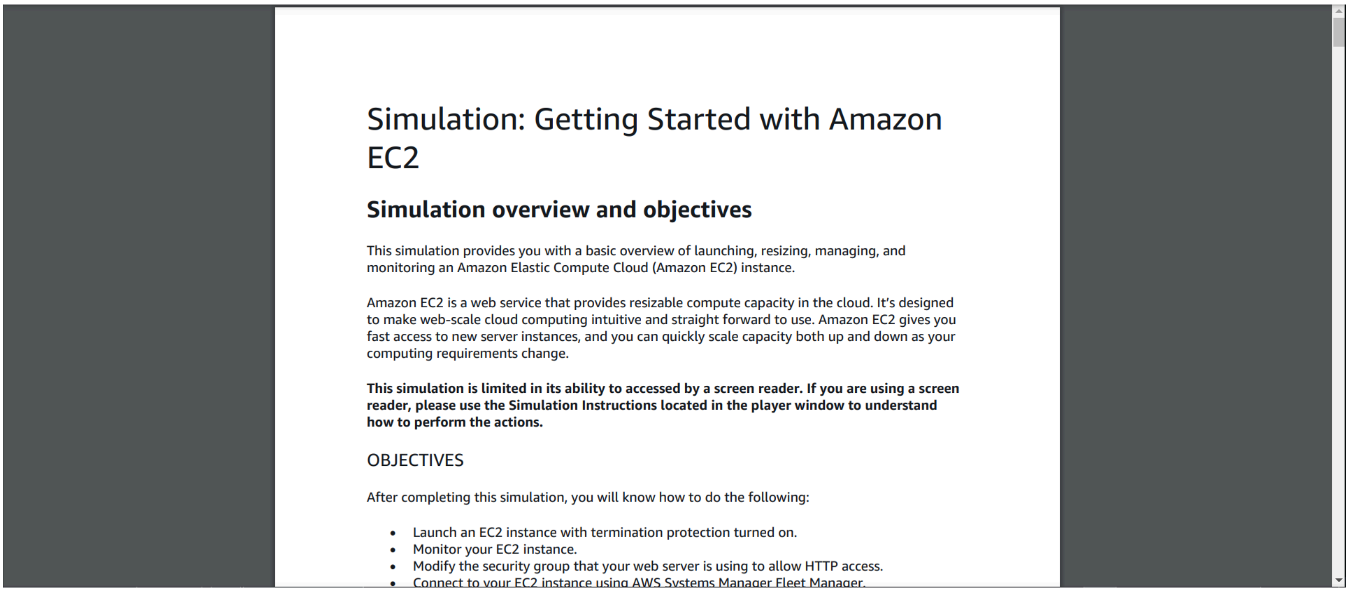
2) Download the **Simulation Instruction pdf**.



Begin by downloading the **Simulation Instruct** upper-right corner of this player window.

Review the introduction to this simulation, and reach **Task 1.** You use the steps outlined in the **Instructions** to navigate each task in this simul

3) Read the entire pdf to understand how to work with EC2 instances. The final results that appear after each task is shown in the window where the simulation runs.



Simulation: Getting Started with Amazon EC2

Simulation overview and objectives

This simulation provides you with a basic overview of launching, resizing, managing, and monitoring an Amazon Elastic Compute Cloud (Amazon EC2) instance.

Amazon EC2 is a web service that provides resizable compute capacity in the cloud. It's designed to make web-scale cloud computing intuitive and straight forward to use. Amazon EC2 gives you fast access to new server instances, and you can quickly scale capacity both up and down as your computing requirements change.

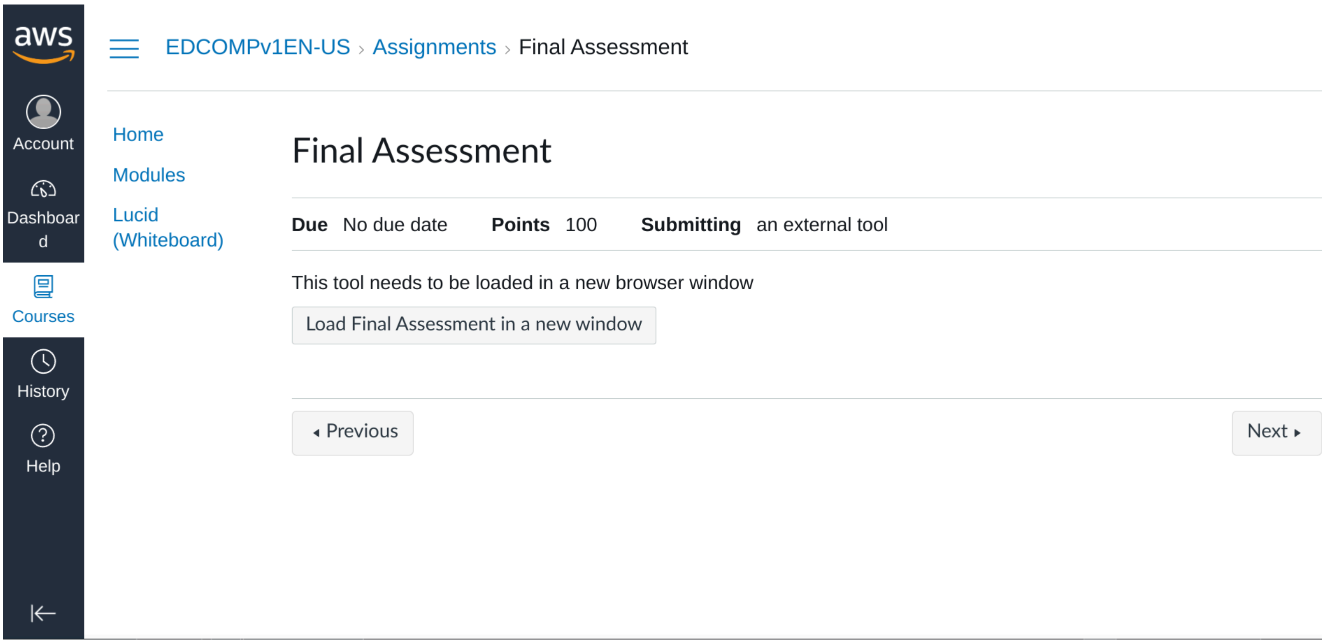
This simulation is limited in its ability to accessed by a screen reader. If you are using a screen reader, please use the Simulation Instructions located in the player window to understand how to perform the actions.

OBJECTIVES

After completing this simulation, you will know how to do the following:

- Launch an EC2 instance with termination protection turned on.
- Monitor your EC2 instance.
- Modify the security group that your web server is using to allow HTTP access.
- Connect to your EC2 instance using AWS Systems Manager Fleet Manager.

4) Once the simulation is complete you can take the Final Assessment by clicking on **Next** in the simulation window.



EDCOMPv1EN-US › Assignments › Final Assessment

Final Assessment

Due No due date **Points** 100 **Submitting** an external tool

This tool needs to be loaded in a new browser window

<u>NOTE</u>: This AWS lab is not hands-on as AWS Educate no longer supports live labs.

Points to note:

1. AWS Educate will create a temporary AWS account with all the required permissions and access to complete the lab. **Do not** use your personal AWS account. To prevent conflicts with any AWS account that you have already signed into on your browser, you can use incognito mode.
2. **DO NOT** change the default region/ VPC or any other settings that are automatically created by AWS Educate.
3. The AWS Educate lab session is timed. When the time limit is reached/the timer expires, the AWS account is deleted, and you must restart the lab from the beginning.
4. All code and configuration for the AWS Educate lab have already been given. You are not required to code anything from scratch or deviate from this for the lab experiments. However, in some cases, you may be required to name the resources you use differently, as instructed.
5. The assignments may require you to deviate from the AWS Educate instructions and use your own code. Instructions will be given.
6. **DO NOT** try to access or avail any other resources and services that have not been described in the lab session or your account will be blocked.

Docker Setup

Since the above lab was easy enough, in this lab we will setup docker for future labs. Docker is a os virtualization tool that helps you to package the application and make it run anywhere. Because of this you will not be provide reason as “this runs on my machine”

Step 1: Installing Docker

Linux

1. Update your system:
   ```
   sudo apt-get update
   sudo apt-get upgrade
   ```

2. Install dependencies:
   ```
    sudo apt-get install \
      apt-transport-https \
      ca-certificates \
      curl \
      software-properties-common
   ```

3. Add Docker’s GPG key:
   ```
    curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
   /usr/share/keyrings/docker-archive-keyring.gpg
   ```
4. Setup Docker’s repository:
   ```
   echo "deb [arch=$(dpkg --print-architecture)
   signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
   https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" |
   sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
   ```

5. Install Docker:
   ```
   sudo apt-get update

   sudo apt-get install docker-ce docker-ce-cli containerd.io
   ```

6. Verify the installation:
   ```
    docker --version
   ```

Mac

1. Download Docker Desktop from the Docker website.
2. Open the `.dmg` file and move Docker to Applications.

3. Launch Docker Desktop and sign in with your Docker account.
4. Confirm the installation in the terminal:

   ```
   docker --version
   ```

Windows

Docker Desktop on Windows relies on **WSL 2** (Windows Subsystem for Linux). If you haven't set it up, no worries—we'll guide you through it!

1. Enable the WSL Feature

Open PowerShell as Administrator and run:

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

2. Install WSL

Download and install the latest WSL version with a single command:

```
wsl --install
```

💡 *Pro Tip:* By default, this installs Ubuntu. You can choose a specific distribution (like Debian or Kali) with:

```
wsl --install -d <DistributionName>
```

Once it's installed, restart your computer.

3. Set WSL 2 as Default

Ensure WSL 2 is the default version:

```
wsl --set-default-version 2
```

4. Install Docker Desktop

1. Download Docker Desktop from the official website.
2. Run the installer. During installation, ensure **WSL 2 integration** is enabled.
3. Restart your system if prompted.

5. Verify Everything Works

Open PowerShell or CMD and check Docker's version:

docker --version

Step 2: Setting Up MongoDB with Docker

1. Pull the MongoDB image:
   docker pull mongo
2. Start a MongoDB container:

   Replace **my-mongo with your SRN**
   docker run --name my-mongo -d -p 27017:27017 mongo

   Explanation for the above :

   `--name my-mongo`: Name of the container.

   `-d`: Run it in the background.

   `-p 27017:27017`: Maps MongoDB's default port to your system.

3. Verify the container is running:
   docker ps

Step 3: Interacting with MongoDB Using Python

Install Python Packages

Make sure you have Python installed. Then, install the MongoDB driver:

pip install pymongo

Python Script to Add and View Data

Save this script as `student_mongo.py`:

```
from pymongo import MongoClient

# Connect to MongoDB
client = MongoClient("mongodb://localhost:27017/")

# Access the database and collection
db = client["student_database"]
collection = db["students"]
```

```
# Add your student ID
student_id = input("Enter your student ID: ")
collection.insert_one({"student_id": student_id})
print(f"Student ID {student_id} added successfully!")

# View all student IDs
print("\nHere are all the student IDs:")
for student in collection.find():
    print(student["student_id"])
```

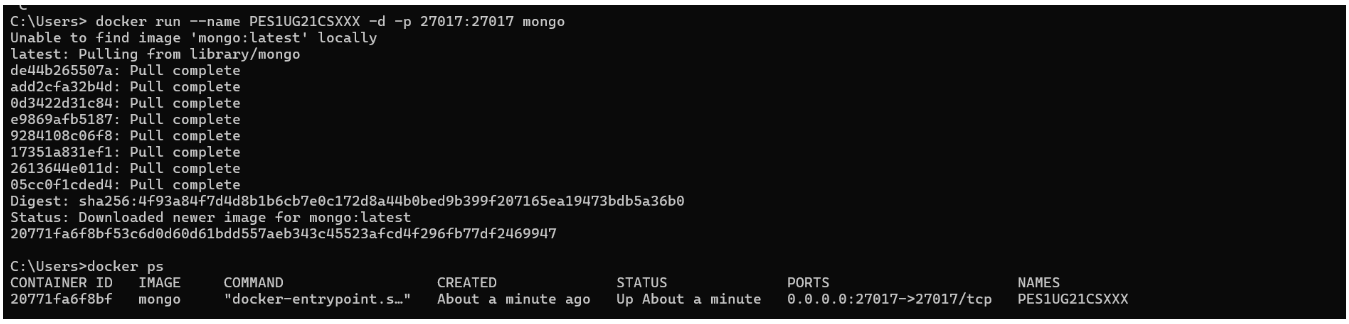
Run the Script

1. Execute the script:
   python student_mongo.py
2. Enter your student ID and confirm it's added to the database.
3. Run the script again to view all saved IDs.

Deliverables:

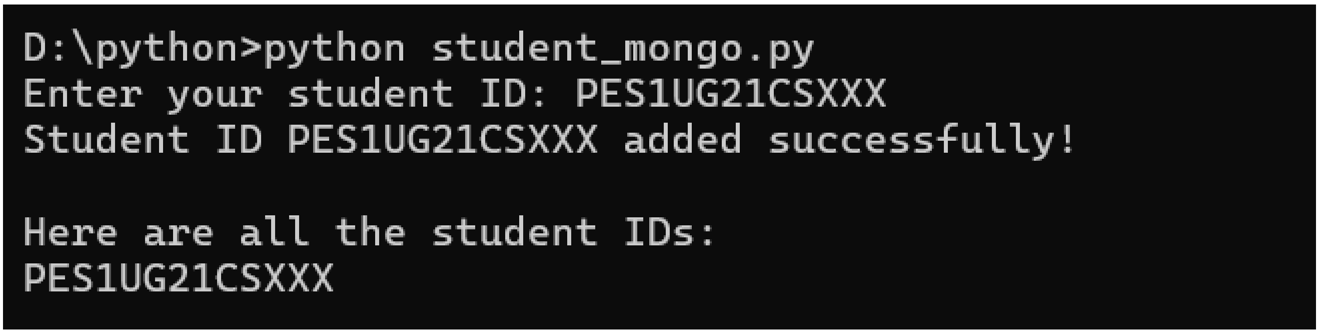
Note: Submit only the screenshots mentioned below The following screenshots are to be submitted:

a)



```
C:\Users> docker run --name PES1UG21CSXXX -d -p 27017:27017 mongo
Unable to find image 'mongo:latest' locally
latest: Pulling from library/mongo
de44b265507a: Pull complete
add2cfa32b4d: Pull complete
0d3422d31c84: Pull complete
e9869afb5187: Pull complete
9284108c06f8: Pull complete
17351a831ef1: Pull complete
2613644e011d: Pull complete
05cc0f1cded4: Pull complete
Digest: sha256:4f93a84f7d4d8b1b6cb7e0c172d8a44b0bed9b399f207165ea19473bdb5a36b0
Status: Downloaded newer image for mongo:latest
20771fa6f8bf53c6d0d60d61bdd557aeb343c45523afcd4f296fb77df2469947

C:\Users>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS                      NAMES
20771fa6f8bf   mongo     "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:27017->27017/tcp   PES1UG21CSXXX
```

b)



```
D:\python>python student_mongo.py
Enter your student ID: PES1UG21CSXXX
Student ID PES1UG21CSXXX added successfully!

Here are all the student IDs:
PES1UG21CSXXX
```

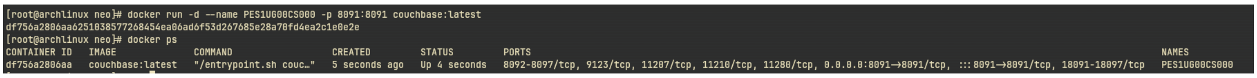
NOTE : The screenshots must be pasted into a document and sent in PDF format. The file should be named in this manner-___E1.pdf (Eg. A_PES1UG21CSXXX_Name_E1.pdf) stop the container with **docker stop PES1UG21CSXXX** at the end

Example with couchbase (foreshadowing)

Step 2: Setting Up Couchbase with Docker

4. Pull the MongoDB image:
   docker pull couchbase:latest
5. Start a MongoDB container:

   Replace **my-couchbase-node with your SRN**
   docker run -d --name my-couchbase-node -p 8091:8091 couchbase:latest



```
[root@archlinux neo]# docker run -d --name PES1UG00CS000 -p 8091:8091 couchbase:latest
df756a2806aa6251038577268454ea06ad6f53d267685e28a70fd4ea2c1e0e2e
[root@archlinux neo]# docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED         STATUS         PORTS                                                                                                         NAMES
df756a2806aa   couchbase:latest   "/entrypoint.sh couc…"   5 seconds ago   Up 4 seconds   8092-8097/tcp, 9123/tcp, 11207/tcp, 11210/tcp, 11280/tcp, 0.0.0.0:8091→8091/tcp, :::8091→8091/tcp, 18091-18097/tcp   PES1UG00CS000
```

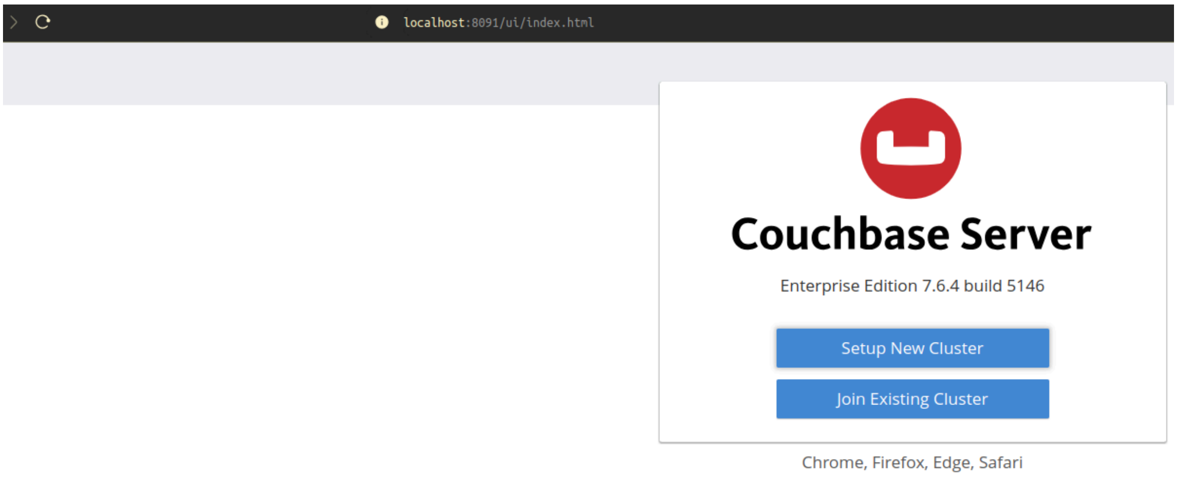
   Explanation for the above :

   `--name my-couchbase-node`: Name of the container -> PES1UGXXXXXXX.

   `-d`: Run it in the background.

   `-p 8091:8019`: Maps COUCHBASE's default port to your system.

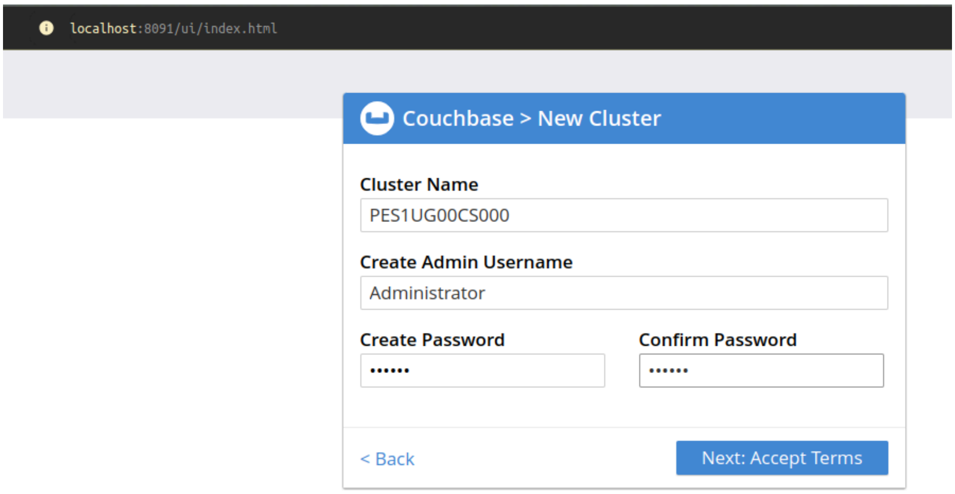
6. Verify the container is running:
   docker ps
7. now go to localhost:8019 to access the couchbase local server ui :



localhost:8091/ui/index.html

Couchbase Server
Enterprise Edition 7.6.4 build 5146
Setup New Cluster
Join Existing Cluster
Chrome, Firefox, Edge, Safari

8. select Setup new cluster and name it as your srn and fill in the password



localhost:8091/ui/index.html

Couchbase > New Cluster

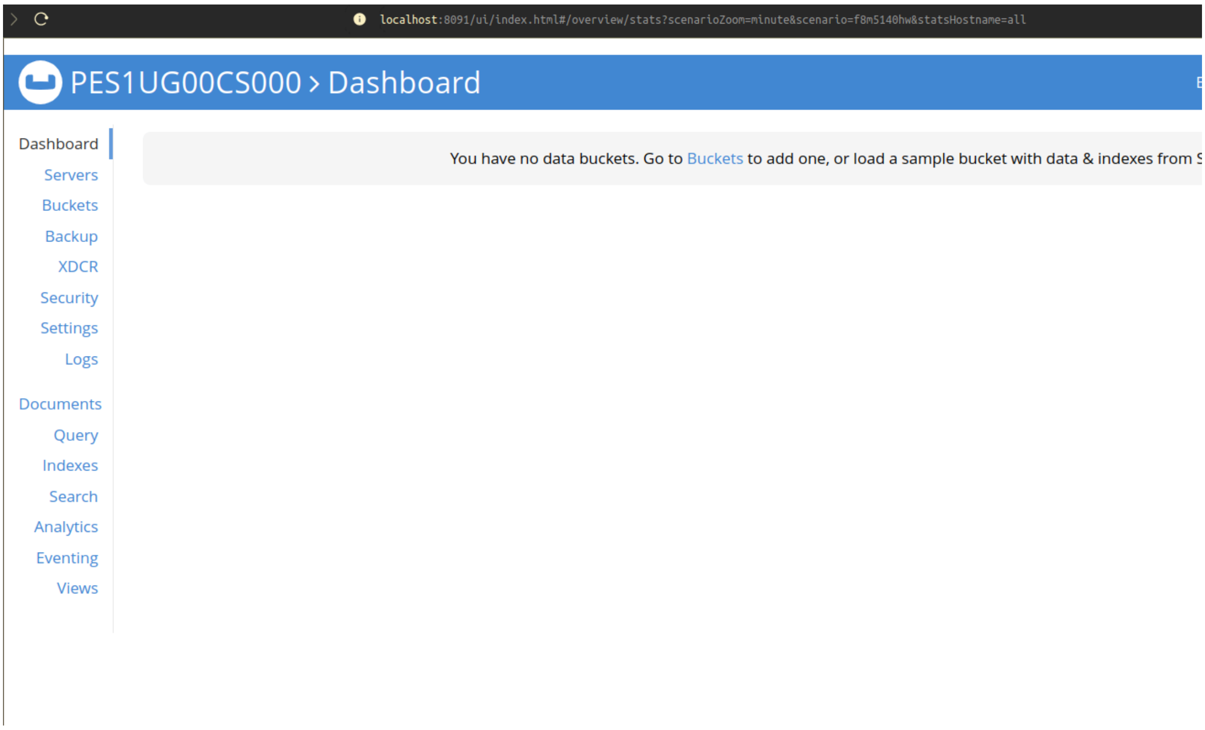
Cluster Name
PES1UG00CS000

Create Admin Username
Administrator

Create Password
••••••

Confirm Password
••••••

9. after that accept the terms and **finish with defaults** , you’ll be in the dashboard page :

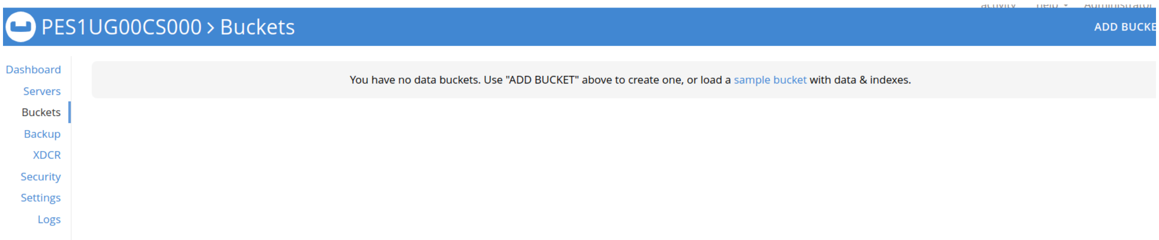


localhost:8091/ui/index.html#/overview/stats?scenarioZoom=minute&scenario=f8m5140hw&statsHostname=all

PES1UG00CS000 > Dashboard

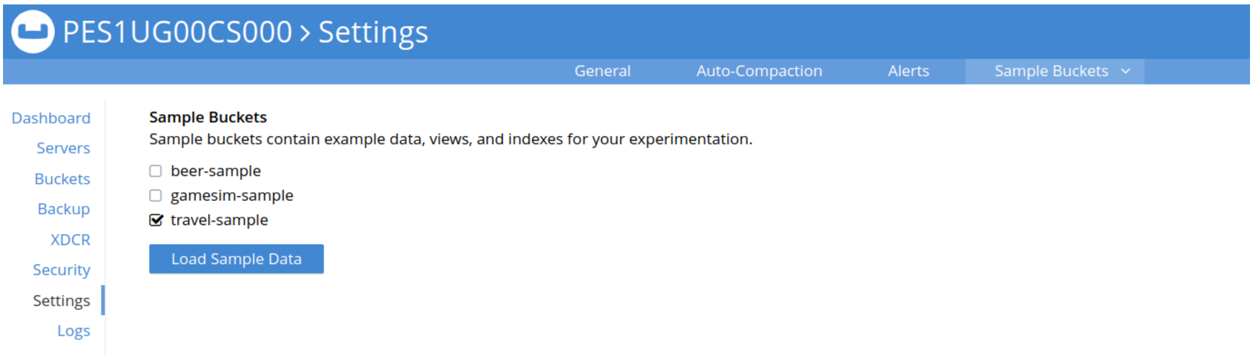
You have no data buckets. Go to Buckets to add one, or load a sample bucket with data & indexes from S

10. Now navigate to Buckets in the sidebar and then click on sample bucket to create a sample dataset bucket , once you do , select travel-sample and finish making the bucket:



PES1UG00CS000 > Buckets

You have no data buckets. Use "ADD BUCKET" above to create one, or load a sample bucket with data & indexes.

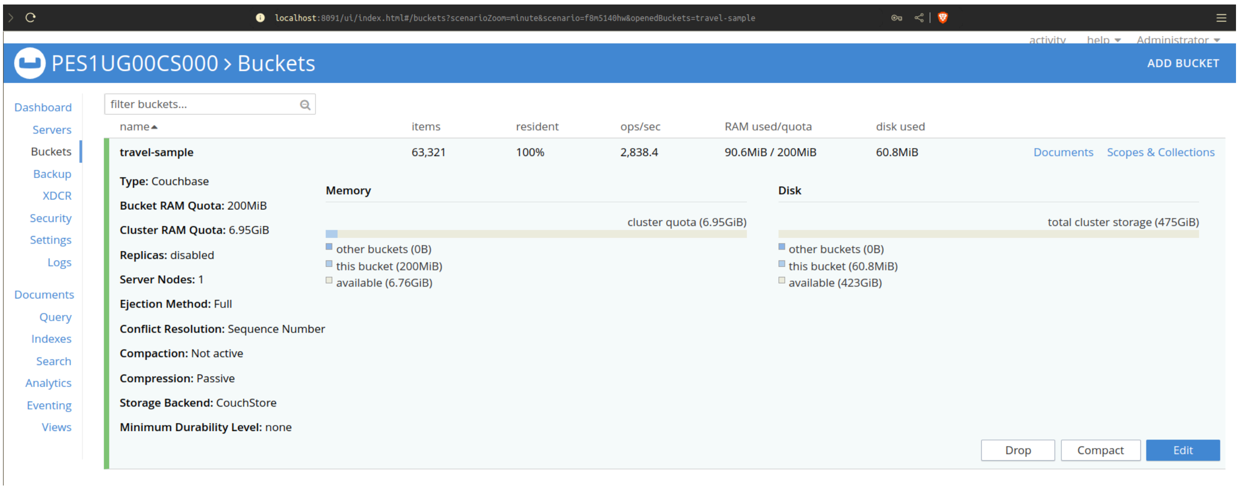


Sample Buckets

Sample buckets contain example data, views, and indexes for your experimentation.

- □ beer-sample
- □ gamesim-sample
- ☑ travel-sample

check the Buckets , you will have your sample bucket of travel-sample loaded in :



name	items	resident	ops/sec	RAM used/quota	disk used
travel-sample	63,321	100%	2,838.4	90.6MiB / 200MiB	60.8MiB

Type: Couchbase
Bucket RAM Quota: 200MiB
Cluster RAM Quota: 6.95GiB
Replicas: disabled
Server Nodes: 1
Ejection Method: Full
Conflict Resolution: Sequence Number
Compaction: Not active
Compression: Passive
Storage Backend: CouchStore
Minimum Durability Level: none

Memory

cluster quota (6.95GiB)

- other buckets (0B)
- this bucket (200MiB)
- available (6.76GiB)

Disk

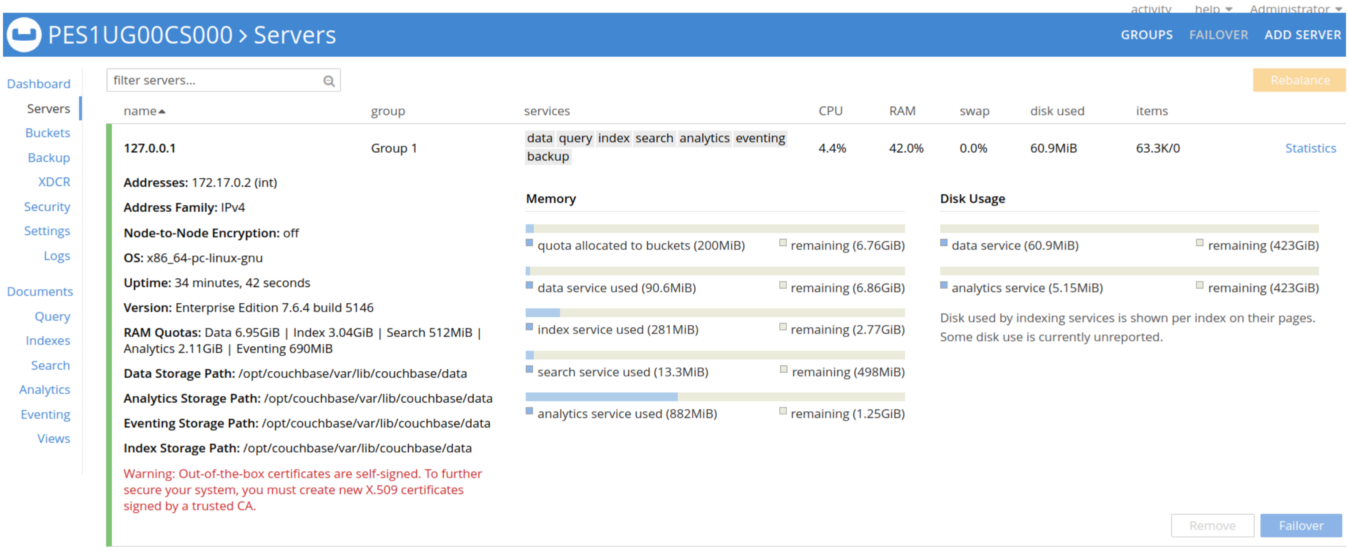
total cluster storage (475GiB)

- other buckets (0B)
- this bucket (60.8MiB)
- available (423GiB)

Step 3: Interacting with Couchbase Using Python

for additional references : docs

here's the main server node, in that the first field is your addresses which contain your ip address which you will be updating in the code below :



name	group	services	CPU	RAM	swap	disk used	items
127.0.0.1	Group 1	data query index search analytics eventing backup	4.4%	42.0%	0.0%	60.9MiB	63.3K/0

Addresses: 172.17.0.2 (int)
Address Family: IPv4
Node-to-Node Encryption: off
OS: x86_64-pc-linux-gnu
Uptime: 34 minutes, 42 seconds
Version: Enterprise Edition 7.6.4 build 5146
RAM Quotas: Data 6.95GiB | Index 3.04GiB | Search 512MiB | Analytics 2.11GiB | Eventing 690MiB
Data Storage Path: /opt/couchbase/var/lib/couchbase/data
Analytics Storage Path: /opt/couchbase/var/lib/couchbase/data
Eventing Storage Path: /opt/couchbase/var/lib/couchbase/data
Index Storage Path: /opt/couchbase/var/lib/couchbase/data

Warning: Out-of-the-box certificates are self-signed. To further secure your system, you must create new X.509 certificates signed by a trusted CA.

Memory

- quota allocated to buckets (200MiB) □ remaining (6.76GiB)
- data service used (90.6MiB) □ remaining (6.86GiB)
- index service used (281MiB) □ remaining (2.77GiB)
- search service used (13.3MiB) □ remaining (498MiB)
- analytics service used (882MiB) □ remaining (1.25GiB)

Disk Usage

- data service (60.9MiB) □ remaining (423GiB)
- analytics service (5.15MiB) □ remaining (423GiB)

Disk used by indexing services is shown per index on their pages. Some disk use is currently unreported.

Install Python Packages

Make sure you have Python installed. Then, install the couchbase driver:

```
pip install couchbase
```

Python Script to Add and View Data

Save this script as `couchbase.py`:

```
from datetime import timedelta

# needed for any cluster connection
from couchbase.auth import PasswordAuthenticator
from couchbase.cluster import Cluster
# needed for options -- cluster, timeout, SQL++ (N1QL) query, etc.
from couchbase.options import (ClusterOptions, ClusterTimeoutOptions,
                               QueryOptions)

# Update this to your cluster
username = "Administrator"
password = "password" # password you had started the cluster with
bucket_name = "travel-sample"
# User Input ends here.

# Connect options - authentication
auth = PasswordAuthenticator(
    username,
    password,
)

# Get a reference to our cluster
# NOTE: For TLS/SSL connection use 'couchbases://<your-ip-address>' instead
# if the below address does not work then look up your couchbase server
  address , on servers -> 127.0.0.1 -> addresses
cluster = Cluster('couchbase://172.17.0.2', ClusterOptions(auth)) #default

# Wait until the cluster is ready for use.
cluster.wait_until_ready(timedelta(seconds=5))

# get a reference to our bucket
cb = cluster.bucket(bucket_name)

cb_coll = cb.scope("inventory").collection("airline")

# Get a reference to the default collection, required for older Couchbase server
versions
cb_coll_default = cb.default_collection()
```

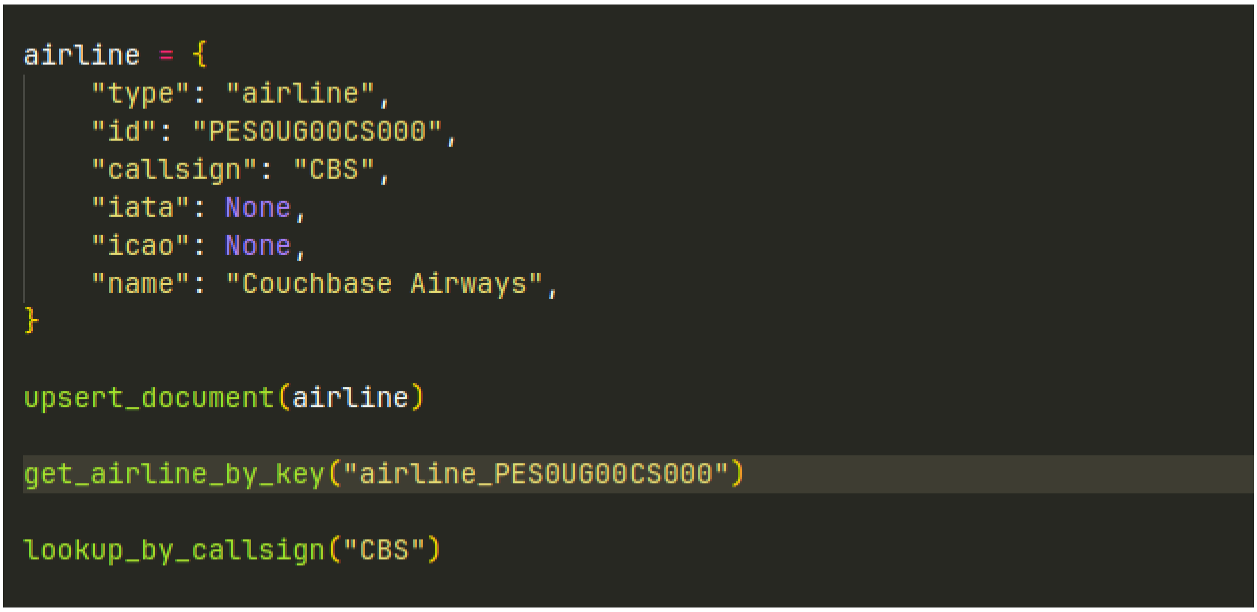
```
# upsert document function


def upsert_document(doc):
    print("\nUpsert CAS: ")
    try:
        # key will equal: "airline_8091"
        key = doc["type"] + "_" + str(doc["id"])
        result = cb_coll.upsert(key, doc)
        print(result.cas)
    except Exception as e:
        print(e)

# get document function


def get_airline_by_key(key):
    print("\nGet Result: ")
    try:
        result = cb_coll.get(key)
        print(result.content_as[str])
    except Exception as e:
        print(e)

# query for new document by callsign


def lookup_by_callsign(cs):
    print("\nLookup Result: ")
    try:
        inventory_scope = cb.scope('inventory')
        sql_query = 'SELECT VALUE name FROM airline WHERE callsign = $1'
        row_iter = inventory_scope.query(
            sql_query,
            QueryOptions(positional_parameters=[cs]))
        for row in row_iter:
            print(row)
    except Exception as e:
        print(e)

# modify the below according to your srn :
airline = {
    "type": "airline",
    "id": “your-srn-here”,
    "callsign": "CBS",
    "iata": None,
    "icao": None,
    "name": "Couchbase Airways",
```

}

upsert_document(airline)

get_airline_by_key("airline_<your-srn-here>") # update your srn here

lookup_by_callsign("CBS")
example :



```
airline = {
    "type": "airline",
    "id": "PES0UG00CS000",
    "callsign": "CBS",
    "iata": None,
    "icao": None,
    "name": "Couchbase Airways",
}

upsert_document(airline)

get_airline_by_key("airline_PES0UG00CS000")

lookup_by_callsign("CBS")
```

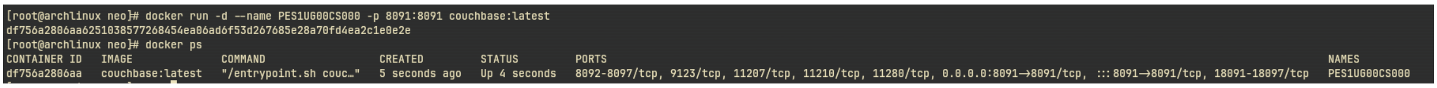
Run the Script

4. Execute the script:
   python couchbase_example.py and save it for the deliverables.

Deliverables:

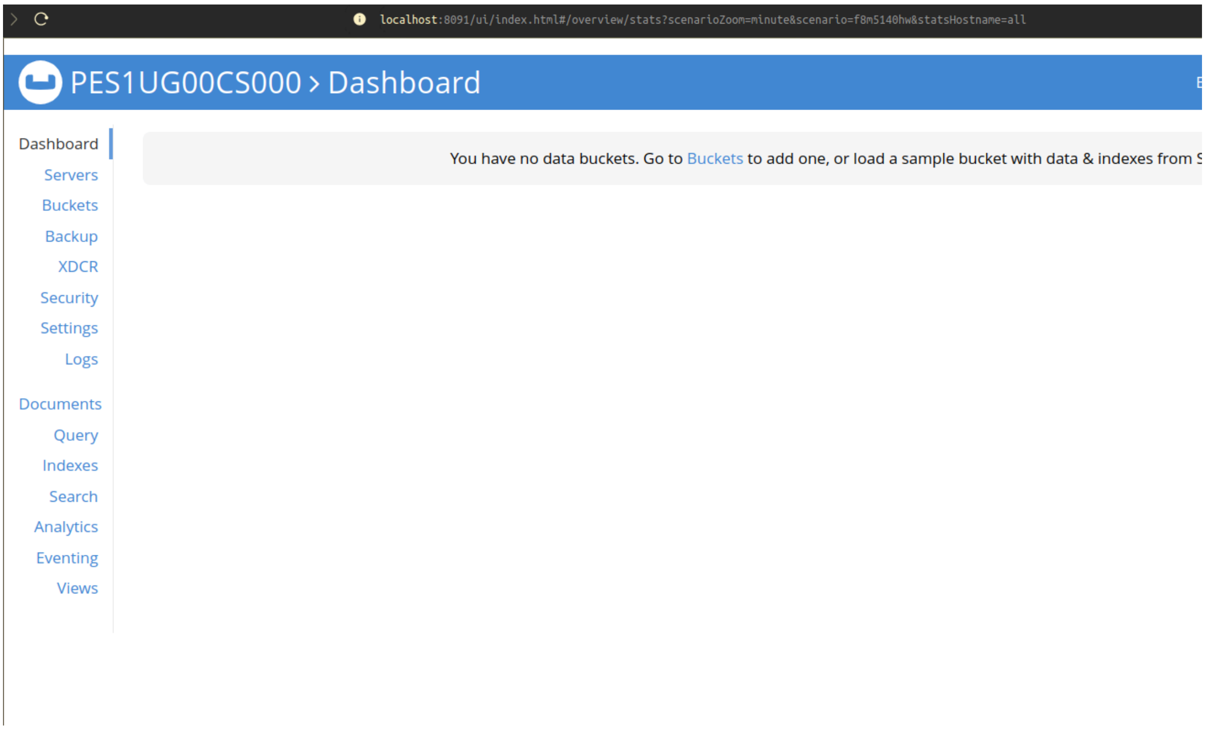
Note: Submit only the screenshots mentioned below The following screenshots are to be submitted:

a) the couchbase cluster running with your SRN



```
[root@archlinux neo]# docker run -d --name PES1UG00CS000 -p 8091:8091 couchbase:latest
df756a2806aa6251038577268454ea06ad6f53d267685e28a70fd4ea2c1e0e2e
[root@archlinux neo]# docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED         STATUS         PORTS                                                                                                    NAMES
df756a2806aa   couchbase:latest   "/entrypoint.sh couc…"   5 seconds ago   Up 4 seconds   8092-8097/tcp, 9123/tcp, 11207/tcp, 11210/tcp, 11280/tcp, 0.0.0.0:8091→8091/tcp, :::8091→8091/tcp, 18091-18097/tcp   PES1UG00CS000
```

b) the ui cluster with your srn :

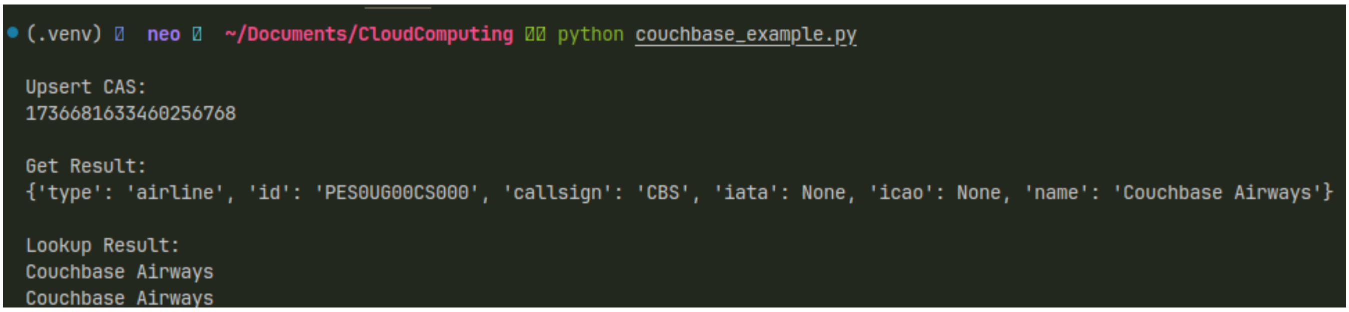


localhost:8091/ui/index.html#/overview/stats?scenarioZoom=minute&scenario=f8m5140hw&statsHostname=all

PES1UG00CS000 > Dashboard

You have no data buckets. Go to Buckets to add one, or load a sample bucket with data & indexes from S

b)the output from the python script



```
● (.venv) ▯  neo ▯  ~/Documents/CloudComputing ▯▯ python couchbase_example.py

  Upsert CAS:
  1736681633460256768

  Get Result:
  {'type': 'airline', 'id': 'PES0UG00CS000', 'callsign': 'CBS', 'iata': None, 'icao': None, 'name': 'Couchbase Airways'}

  Lookup Result:
  Couchbase Airways
  Couchbase Airways
```

NOTE : The screenshots must be pasted into a document and sent in PDF format. The file should be named in this manner-___E1.pdf (Eg. A_PES1UG21CSXXX_Name_E1.pdf)

stop the container with **docker stop PES1UG21CSXXX** at the end