Intro:

Background/experience

Tech stack

Cloud Computing familiarity

Movies/series

Req -

Storage

Processing

Network

Power

Heat dissipation

Case Studies:

https://youtu.be/5K4p4YHK6LU Big Data with AWS for JPL

Apr 25, 2023

Creating a VM & hosting apache web distribution manually

1. Go to Compute Engine > VM instances > Create
2. Specs:
    a. Name - machine name preferred
    b. Region - us-central1 zone - us-central1-a
    c. Machine series: N1 Type n1-std-1
    d. Boot disk:
        i. OS: Ubuntu

        ii. Version: 18.04 (x86 based)

    e. Create VM

3. Wait for the VM to have running status. Locate ssh button besides VM name.
4. Click on ssh button to connect with vm.
5. Execute following commands:
    a. sudo apt update
    b. sudo apt install apache2 -y
6. Go back to GCP console. Locate “External IP” in the VM entry. Copy the external IP and paste in any browser tab to access apache webpage. You should get a page not found error.

   {If you’re not able to locate external IP, click on three slider icon towards the right top corner of VM entry. This option is **column-display-options**. Select external IP & internal IP from the options.}
7. Go back to GCP console > locate “Internal IP”. Go to the ssh command window and execute command:
    a. curl http://[internal-ip]
8. You should see an html code. Take a screenshot of this as last step.
9. **Delete** the VM instance resource.

Creating a VPC

1. Go to VPC Networks > Create VPC
    a. Name: john-web
    b. Mode of Creation: custom
    c. Subnet name: john-sub
    d. Range: select a relevant cidr range
    e. Region: us-central1 (feel free to change)
    f. Create subnet
    g. Firewall rules: select all entries
    h. Create VPC
2. Once VPC is created, go to firewall rules

a. Create firewall rule

b. Name: vpc-allow-http

c. Network: select your vpc from dropdown

d. Target: input a target tag (http-john, spider-man)

e. Source Filter: 0.0.0.0/0

f. Protocol: TCP - 80

3. Visit compute engine > create instance

   a. Ensure all settings as earlier exec

   b. Click on Advanced > Networking

   c. Select your vpc name from network drop-down > Done

   d. Click on Automation > Startup script

   e. Input startup script as follows:

      i. apt update

      ii. apt -y install apache2

      iii. cat <<EOF > /var/www/html/index.html

      iv. <html><body><p>Linux startup script from a local file.</p></body></html>

      v. EOF

   f. Create Instance

   CLI way -

   - For creating instance: gcloud compute instances create vm-name --machine-type=n1-standard-1 --zone=us-central1-a --network=vpcname --subnet=subnetname
   - For accessing the vm - gcloud compute ssh vm-name --zone=us-central1-a
   - After accessing the vm, execute following commands:
     - sudo apt update
     - sudo apt install apache2 -y

4. Visit External IP of instance to check if apache2 hosted webpage is available.

5. For consistent failed page loads, go to firewall rule > click on the firewall rule you have recently created > edit > target: change to “all instances in the network” > save
6. Retry accessing the external IP
7. Once the webpage is accessible, take screenshots and delete instance, firewall-rule & VPC network.

Creating App Engine

1. Clone the code repo: git clone https://github.com/GoogleCloudPlatform/python-docs-samples.git
2. Shifting code to directory: cd python-docs-samples/appengine/standard_python3/hello_world
3. Edit code file
    a. nano main.py
    b. Replace *Hello World!* with a custom message with your name.
    c. Save file [Ctrl+o, enter, Ctrl+x]
4. Deploy application using - gcloud app deploy
5. Fetch endpoint using - gcloud app browse
6. Visit the end point for app output
7. Edit code file for more changes using nano editor
8. Redeploy app using gcloud app deploy
9. Visit the endpoint for observing output change. It must be updated to new code snippet that you edited
10. Go to App Engine > Versions > Split traffic
11. Split traffic using percentage values and random mode. Save split
12. Try visiting the endpoint over few iterations and notice the output change

Apr 27, 2023

K8s

1. Create gke cluster
    a. gcloud container clusters create clustername --zone=us-central1-a --machine-type=n1-standard-1 --num-nodes=2
2. Create a container deployment on cluster for the app [**using nginx server image**]
    a. kubectl create deployment deploymentname1 --image=nginx:1.10.0
3. Create deployment [**using manual code**]
    a. Create a file server.js with following code

    ```
    var http = require('http');
    var handleRequest = function(request, response) {
      response.writeHead(200);
      response.end("Hello World!");
    }
    var www = http.createServer(handleRequest);
    www.listen(8080);
    ```

    b. Test this file locally:
        i. Run following command in cloudshell
            1. node server.js
        ii. Go to web preview option on cloudshell and select preview on port 8080

    c. Create a new file with following code:

    ```
    FROM node:6.9.2
    EXPOSE 8080
    COPY server.js .
    CMD node server.js
    ```

    d. Save as **Dockerfile**

    e. Create a CI using this code
        i.

    ```
    gcloud builds submit
    --tag=gcr.io/[PROJECT_ID]/hello-node:v1 .
    ```

      ii. Visit Google Container Registry to verify a new CI image is available over there.

   f. Use the following command

      i. kubectl create deployment deploymentname2 --image=gcr.io/[PROJECT_ID]/hello-node:v1

4. Exposing the deployment (try both methods for differential understanding)

   a. **UI Way** - Go to Workloads > select deployment name > actions > expose

      i. Ensure the port is 80 & type is set to Load Balancer > expose

   b. **CLI** - kubectl expose deployment [deploymentname] --type="LoadBalancer" --port=80

   c. Once exposed, visit services > servicename > locate an external IP. Enter the external IP in browser to gert UI view

5. Scaling the deployment

   a. UI Way - Go to worklaods > deployment > actions > scale > edit replicas > input 3 > scale

   b. CLI - kubectl scale deployment [deploymentname] --replicas=4

6. Input following commands randomly during execution to fetch cluster status

   a. kubectl get pods

   b. kubectl get deployment

Alternative app files

For Python :

app.py

```
import os

from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    target = os.environ.get('TARGET', 'World')
    return 'Hello {}!\n'.format(target)

if __name__ == "__main__":
    app.run(debug=True,host='0.0.0.0',port=int(os.environ.get('PORT', 8080)))
```

Dockerfile

```
FROM python:3.7-slim

ENV APP_HOME /app
WORKDIR $APP_HOME
COPY . ./

RUN pip install Flask gunicorn
CMD exec gunicorn --bind :$PORT --workers 1 --threads 8 app:app
```

Apr 28, 2023

Automating CI/CD pipeline over google cloud platform

Code files >>

main.py

```
#!/usr/bin/env python

import webapp2

class MainHandler(webapp2.RequestHandler):
    def get(self):
        self.response.write('Hello, Peter!')

app = webapp2.WSGIApplication([
    ('/', MainHandler)
], debug=True)
```

Create a second file app.yaml

```
runtime: python27
api_version: 1
threadsafe: yes

handlers:
- url: .*
  script: main.app

libraries:
- name: webapp2
  version: "2.5.2"
```

Create a Google Cloud Sourse Repository with

- `gcloud source repos create repo-name`

Verify the repository creation with

- `gcloud source repos list`

Clone the repo in your cloud shell using

```
1. gcloud init
     a. Reinitialise [first choice]
     b. Existing user email [first choice]
     c. Project id [enter your project ID]
     d. Region & zone choice [enter numeric choice for your
        preferred region/zone]
2. gcloud source repos clone cloud-rep
   --project=project2-1676864400036
          [Change repo & cloud name to your configs]
```

Navigate to repo using cd repo-name

Create code files over here. Use basic git commands to upload file to GCSR

a. Git add .
b. Git status
c. Git commit -m “comment”
d. Git push origin master
e. If needed set git credentials with (Git config - -global user.email “youremail@gmail.com” & git config - -global user.name “github username”)

Go to the GCSR url and verify file upload.

Visit Cloudbuild > trigger > manage repositories > options > add trigger

Verify all the settings:

- Provide any trigger name
- Region is global
- Event set to push to a branch
- Source should be your repository name
- Branch should be default to master
- Configuration type is Cloudbuild
- Save the trigger
- After creation, notice your trigger and click on Run
- Observe the trigger run in Cloud build > history

Create a new configuration file

Cloudbuild.yaml >>

```
steps:
- name: "gcr.io/cloud-builders/gcloud"
  args: ["app", "deploy"]
timeout: "1600s"
```

- Push file to the GCSR using git commands
- Notice that in Cloud Build > history, you should have an automated new entry. This is because you have uploaded the cloudbuild file.
- Now, notice the output of build entry. You will observe an app engine appspot url. Visit the url to observe the output of your code.

Permission Error:

- 3432368@cloudbuild.gserviceaccount.com does not have permissions to access apps.
- **Solution 1**
    - Go to Cloudbuild > settings > **enable app engine**, **service account**, **cloud build**
- **Solution 2**
    - Go to home dashboard > Search App Engine Admin API > Enable

Verification:

- Make changes to the main.py file code in your cloudshell
    - Change Hello, Peter to some other quote
- Push the file again with git commands
    - Git add .
    - Git status
    - Git commit
    - Git push origin master
- Observe the update in GCSR for file change

- Observe new entry in the Cloud build > history

For filtering the invocations of your trigger

- Go to Cloudbuild triggers > Click on your trigger name
- Notice “view triggered builds” option
- Click on this to view all the invocations of your trigger in filtered way

Copying the files

- cd
- cp main.py app.yaml ~/repo-name
- Git add ….

Clean up -

- Go to app engine > remove all versions of your app deployment
- Go to Cloud build > delete the trigger
- Visit GCSR & delete repo
    - CLI gcloud source repos

May 2, 2023

Dataproc

Go to Dataproc > Create Cluster > Create with Compute Engine {Enable API if disabled}

- You are in Setup Cluster menu by default
    - Cluster name: teksys-john
    - Cluster Type: Standard
    - Check Enable Component gateway
    - Select Zookeeper from optional components
- Switch to Configure nodes from left hand menu
    - Manager Node Config
        - Machine Series: N1
        - Type: n1-standard-2
        - Primary-disk-size: 200 GB
    - Select same config for Worker node (number of worker nodes is 2)
    - Leave secondary worker node at zero
- Switch to Customize Cluster
    - Select the available subnetwork
- Click Create Cluster

Once the cluster has been created, go to jobs from dataproc navigation menu.
Submit job as follows:

- Select your Cluster name from the cluster list
- Set Job type to **Spark**
- Set Main class or jar to **org.apache.spark.examples.SparkPi**
- Set Arguments to the single argument **1000**
- Add **file:///usr/lib/spark/examples/jars/spark-examples.jar** to Jar files:
    - file:/// denotes a Hadoop LocalFileSystem scheme. Dataproc installed /usr/lib/spark/examples/jars/spark-examples.jar on the cluster's master node when it created the cluster.

- Submit the job.
- Click on job details to view the output/job information aling with it’s logs.

SQL

Fetch teh source file from - https://storage.googleapis.com/teksys-john-sql/create_table.sql

Creating instance with CLI:

- gcloud sql instances create mydb --tier=db-n1-standard-1 --activation-policy=ALWAYS
- gcloud sql users set-password root --host % --instance mydb --password Passw0rd
- export ADDRESS=$(wget -qO - http://ipecho.net/plain)/32
- gcloud sql instances patch mydb --authorized-networks $ADDRESS
- MYSQLIP=$(gcloud sql instances describe mydb --format="value(ipAddresses.ipAddress)")
- mysql --host=$MYSQLIP --user=root --password --verbose

Creating instance with UI:

Go to Cloud SQL > Create instance > Choose MySQL

- Instance ID: teksys-john
- Password: jack123 [setup something simple]
- DB version: latest
- Configuration to start with: Development
- Region & zone: can be left default [change in case of quota errors]
- Show advanced configurations > Storage > 20 GB
- Uncheck “Enable automatic storage increases”
- Create Instance

Wait for the instance to be in runnings status

Go to databases > create database > bts > create

To get data files in Google cloud storage bucket

- git clone https://github.com/GoogleCloudPlatform/data-science-on-gcp/
- cd data-science-on-gcp/03_sqlstudio
- export PROJECT_ID=$(gcloud info --format='value(config.project)')
- gsutil mb gs://teksys-john-sql
- gsutil cp create_table.sql gs://$BUCKET/create_table.sql
- You can import this file by visiting > instance > import > browse for cloud storage bucket and select create_table.sql file > import

To connect with SQL instance

- gcloud sql connect myinstance --user=root

To create db from command line

```
CREATE DATABASE guestbook;

USE guestbook;
CREATE TABLE entries (guestName VARCHAR(255), content VARCHAR(255),
   entryID INT NOT NULL AUTO_INCREMENT, PRIMARY KEY(entryID));
   INSERT INTO entries (guestName, content) values ("first guest", "I got here!");
INSERT INTO entries (guestName, content) values ("second guest", "Me too!");

SELECT * FROM entries;
```

May 3, 2023

Bigquery

Download the source data file from here:
https://storage.googleapis.com/teksys-john-sql/products.csv
Upload this data file to a cloud storage bucket.

1. Go to BigQuery > project > options > create dataset
    a. Dataset ID: prod
    b. Select region/multi-region depending upon your bucket location & region.
    c. Create dataset
2. Go to dataset > options > create table
    a. Create table from "Google Cloud Storage"
    b. Select the source gcs bucket & file from browse option
    c. File format will be autoselected
    d. Verify your project and dataset ID
    e. Input table name as products
    f. Check *Auto-detect* for schema
    g. Create Table
3. Once table is created, data from source file will be imported. Verify from table info, try out some queries for checking SQL compliant nature of bigQuery. You can try commands on data of your own also.

For fetching public dataset,

1. Use following command:

```
#standardSQL
SELECT
 weight_pounds, state, year, gestation_weeks
```

```
    FROM
     `bigquery-public-data.samples.natality`;
```

2. After entering the command, observe the predicted size of data that query will process upon execution. This can also be estimated from cloudshell by the following command :

```
    bq query \
    --use_legacy_sql=false \
    --dry_run \
    'SELECT
      COUNTRY,
      AIRPORT,
      IATA
     FROM
      `project_id`.dataset.airports
     LIMIT
      1000'
```

3. Change the projectid, dataset name and table name in the above command as needed.

Looker

1. Go to lookerstudio.google.com
2. Click on create report > choose source as bigquery > public datasets > select your project > select any dataset you're interested to work on.
3. Create a visualization using add chart option in the above task bar.
4. Alternatively, you can use your own dataset from bigquery for visualizatio building too.

babynames
spls/gsp072/baby-names/yob2014.txt

Tab: names_2014
Schema
name:string,gender:string,count:integer

```
#standardSQL
SELECT
 name, count
FROM
 `babynames.names_2014`
WHERE
 gender = 'M'
ORDER BY count DESC LIMIT 5;
```

Ip-team-1
May 2nd, 2023 Project work Summary:

- Overview the case study document
- Understood cloud architecture
- Researched about cloud service components

User case stories
Status quo
Challenges
Cloud solution

Bigtable

1. Enable BigTable and Dataflow API.
2. Create a GCS bucket.
3. Upload files to your GCS bucket via cloudshell
   a.
   ```
   gsutil cp
   gs://dataflow-templates/latest/GCS_SequenceFile_to_Cloud_Bigtable gs://your_bucket_name/
   ```
4. Setup environment variables -
   a.
   ```
   INSTANCE_ID="del-instance"
   CLUSTER_ID="del-clust"
   TABLE_ID="bus-data"
   CLUSTER_NUM_NODES=3
   CLUSTER_ZONE="us-central1-c"
   ```
5. Create cloud bigtable instance with CLI
   a.
   ```
   gcloud bigtable instances create $INSTANCE_ID \
       --cluster=$CLUSTER_ID \
       --cluster-zone=$CLUSTER_ZONE \
       --cluster-num-nodes=$CLUSTER_NUM_NODES \
       --display-name=$INSTANCE_ID
   ```
6. Write environment variables to cbt file
   a.
   ```
   echo project = $GOOGLE_CLOUD_PROJECT > ~/.cbtrc
   echo instance = $INSTANCE_ID >> ~/.cbtrc
   ```
   b.
   ```
   cbt createtable $TABLE_ID
   cbt createfamily $TABLE_ID cf
   ```
7. Setup a variable for max workers & create dataflow job
   a.
   ```
   NUM_WORKERS=$(expr 2 \* $CLUSTER_NUM_NODES)
   ```
   b.
   ```
   gcloud beta dataflow jobs run import-bus-data-$(date
   +%s) \
   --gcs-location
   gs://replace_your_bucket_name_here/GCS_SequenceFile_to_Cloud_Bigtable \
   --num-workers=$NUM_WORKERS --max-workers=$NUM_WORKERS
   \
   --parameters
   bigtableProject=$GOOGLE_CLOUD_PROJECT,bigtableInstance
   ```

```
Id=$INSTANCE_ID,bigtableTableId=$TABLE_ID,sourcePatter
n=gs://cloud-bigtable-public-datasets/bus-data/*
```

8. Go to Dataflow. Observe the newly submitted job. Monitor number of workers via autoscaling in job info panel. Note time of completion required for job.

Observe the job output. Repeat steps from 7 by replacing following command:

a. `NUM_WORKERS=$(expr 1 \* $CLUSTER_NUM_NODES)`

We are reducing the number of workers here, hence the time should also change.

May 8, 2023

1) Create a bigquery dataset - ‘*ecommerce*’
2) Execute following query to fetch and save the data:
   ```
   #standardSQL
    CREATE OR REPLACE TABLE ecommerce.all_sessions_raw_dataprep
    OPTIONS(
      description="Raw data from analyst team to ingest into Cloud Dataprep"
    ) AS
    SELECT * FROM `data-to-insights.ecommerce.all_sessions_raw`
    WHERE date = '20170801'; # limiting to one day of data 56k rows for this lab
   ```
3) Go to Dataprep > New flow > For source (click on connect data) > Bigquery > select ecommerce > all_sessions… as source
4) Once the data is loaded observe any/all fields with mismatch or corrupt data values. Take necessary steps to transform this data.
5) Once done with transformations, click on schedule > select a weekly mode and time for execution of this flow.
6) Now, this automated flow pipeline for data fetch and transformation will work along with the final data dump. By default, your final data_dump destination will be a cloud storage location.
7) You can confirm location by clicking on third/final block on the transformation flow. This automated process will be run with the help of dataflow. You can also check the settings for dataflow and what transformation its going to make by clicking on manual settings option in info panel.
8)

May 9, 2023

AWS console sign-in url:
https://500452834600.signin.aws.amazon.com/console

Creation of security groups:

1) Go to VPC > copy VPC ID
2) Go to Security groups > paste the ID and search
3) You will get one entry. Copy the default security group ID
4) Click on create security groups
    a) Give any name & description
    b) Select VPC from the dropdown
    c) Add inbound rule : entry 1
        i) Source : paste the default security group ID
        ii) Type: All traffic
    d) Add inbound rule : entry 2
        i) Source: anywhere ipv4
        ii) Type: SSH
    e) Create Security group

For connecting from a server to other server:

```
ssh -i keypair username@hostname
```

Mount filesystem:

- sudo file -s /dev/xvdf
- sudo mkfs -t ext4 /dev/xvdf
- sudo mkdir /data
- df -h
- Lsblk > to verify 12G entry

May 11, 2023

Install CLI -
https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

RDS db

1) Create an EC2 instance
    a) regular specifications
        i) Ubuntu 20.04
        ii) Default VPC & subnet, 2 security groups for SSH access
        iii) Keypair setting etc
    b) Note down name & ID
2) Create database instance
    a) DB creation method: Standard Create
    b) Engine options: Aurora (MySQL compatible)
    c) Template: Dev/Test
    d) Settings > DB cluster identifier: rds-john-db
    e) Master username: admin
    f) Password: admin123
    g) Instance config: db.r5.large (2vCPU, 16 Gib)
    h) Availability & durability: Don't create an Aurora Replica
    i) Connectivity > Compute Resource > Connect to an EC2 instance
    j) Select instance from drop down (choose the instance created earlier)
    k) Network type: ipv4
    l) DB subnet group: choose existing > default
    m) Public Access: No
    n) VPC security group : Create new > name: rds-sg-john
    o) DB auth options: Password authentication
    p) Monitoring: *uncheck* Turn on Performance insights

q) Create db

3) ssh into EC2 instance & execute

a) For mysql version check: mysql --version

b) You won't have a default version. Install a new one using: sudo apt install mysql-client-core-8.0

c) This will not install the client as you don't have updated sys libraries

d) For sys update: sudo apt update

e) Reexecute mysql installation command: sudo apt install mysql-client-core-8.0

f) Check version: mysql --version

4) Connect with database using: mysql -h hostname -P 3306 -u username -p

To find hostname:

- Go to RDS > click on instance name
- Under connectivity security tab, check Endpoints
- Copy Endpoint name for writer instance (type of instance is specified in information towards right column of same entry)

The command should finally look something like this:

```
mysql -h
userdb.cluster-cf9mlmkyeqmy.eu-central-1.rds.amazonaws.com -P
3306 -u admin -p
```

PS. the command should be written continuously and follows case sensitive behaviour.

- Upon executing this, you'd be prompted for password. Enter the password that you set earlier. This should connect you to RDS.
- Fire up some Mysql commands here to create DB & verify db working

Once done, delete EC2 instance as well as RDS. Also, go to snapshots in RDS and delete the snapshot

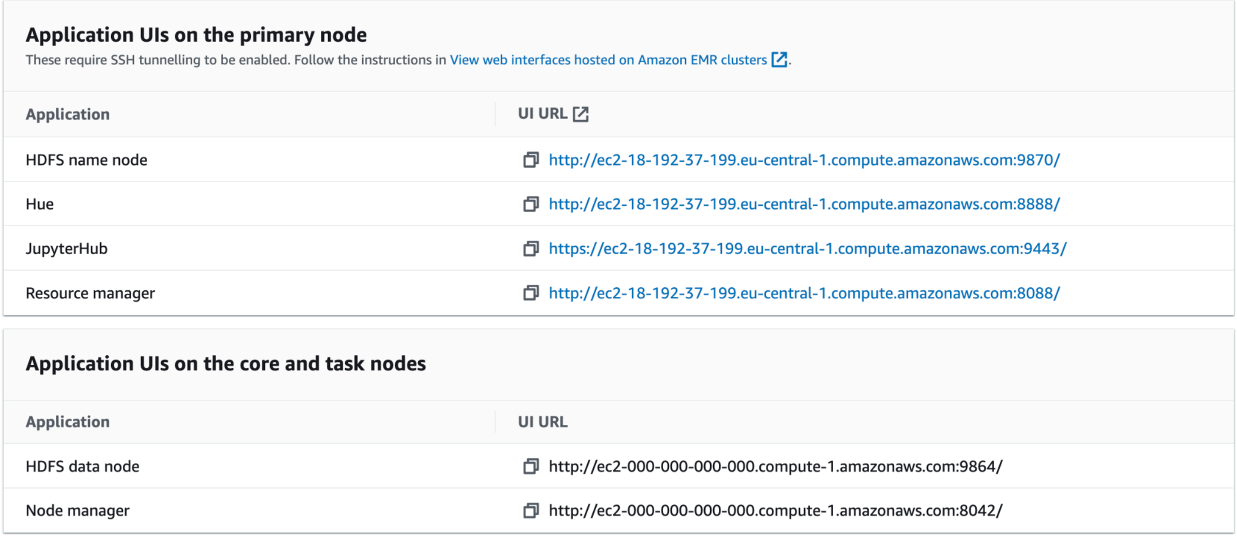
May 16, 2023

EMR

Go to EMR > Create cluster

1. Name: teksys-clust
2. EMR release: emr-6.10.0
3. Application bundle: custom
    a. Select - Hue, Zookeeper, Hadoop, JupyterHub [deselect all others]
4. OS: Amazon Linux release
5. Primary > Instance type: m5.xlarge
6. Core > Instance type: m5.xlarge
7. Task 1 > Remove task instance group
8. Cluster scaling: set cluster size manually
9. Cluster termination: Deselect "Use termination protection"
10. IAM roles: **Create a service role**
11. EC2 instance profile for amazon EMR: **Create an instance profile**
12. Leave everything as default > create cluster

Once the cluster is up and running, go to cluster details > Applications
You will find url to connect to jupyterhub here. Visit the url and gain access to jupyter notebook. In case of connection issue, check security groups and allow the relevant traffic ports.



Application UIs on the primary node

These require SSH tunnelling to be enabled. Follow the instructions in View web interfaces hosted on Amazon EMR clusters.

Application	UI URL
HDFS name node	http://ec2-18-192-37-199.eu-central-1.compute.amazonaws.com:9870/
Hue	http://ec2-18-192-37-199.eu-central-1.compute.amazonaws.com:8888/
JupyterHub	https://ec2-18-192-37-199.eu-central-1.compute.amazonaws.com:9443/
Resource manager	http://ec2-18-192-37-199.eu-central-1.compute.amazonaws.com:8088/

Application UIs on the core and task nodes

Application	UI URL
HDFS data node	http://ec2-000-000-000-000.compute-1.amazonaws.com:9864/
Node manager	http://ec2-000-000-000-000.compute-1.amazonaws.com:8042/

===================

===================

Redshift

Click on get redshift serverless

Customize settings

1. Namespace: teksys
2. Admin user credentials: "Customize admin user credentials"
3. Admin user name: john
4. Admin password: john12345

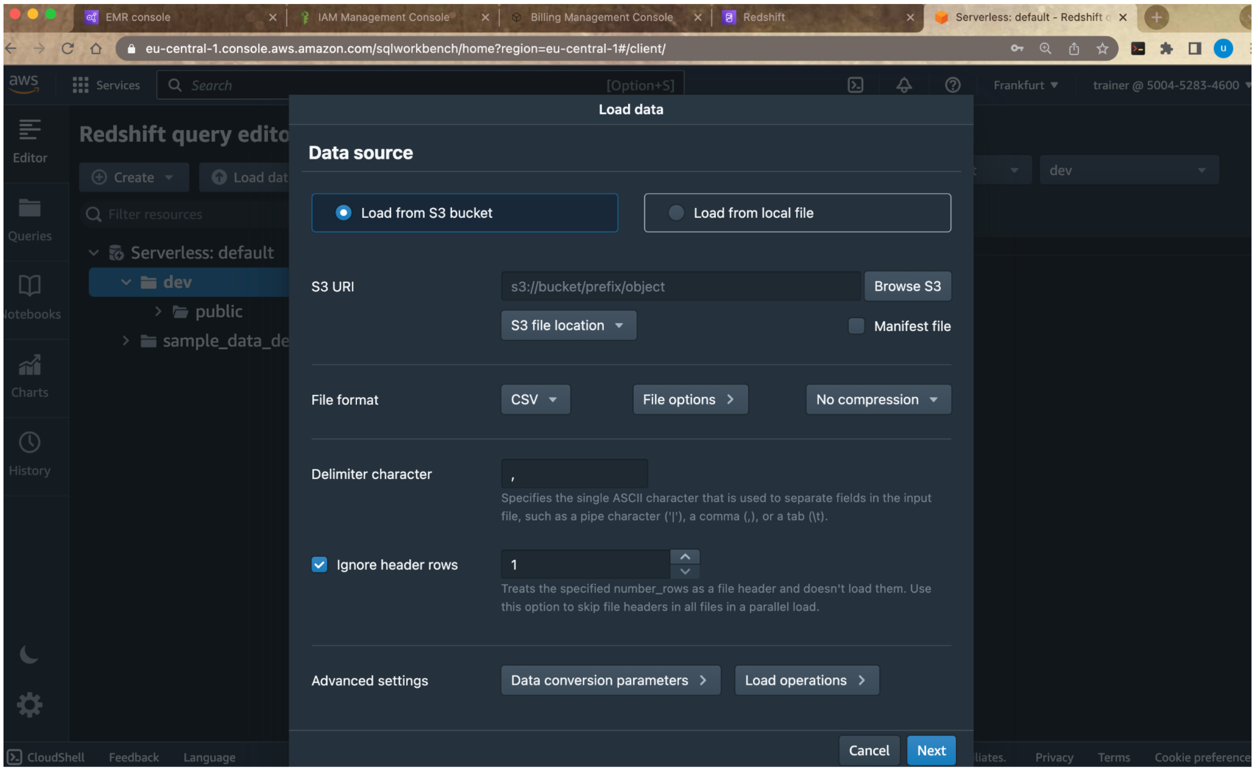
Leave everything else default and click **Save Configuration**

Once the configurations have been saved > you will be presented with a dashboard.
Click on Query data > this will redirect console to redshift UI

Click on options towards the right of serverless configuration: default > Create connection > federated user > create.

Upon successful connection, you will see dev as database name and a query editor will be available towards right.

Click on load data at top to explore data loading through S3.



Load data

Data source

● Load from S3 bucket ○ Load from local file

S3 URI: s3://bucket/prefix/object

S3 file location ☐ Manifest file

File format: CSV | File options | No compression

Delimiter character: ,

Specifies the single ASCII character that is used to separate fields in the input file, such as a pipe character ('|'), a comma (,), or a tab (\t).

☑ Ignore header rows: 1

Treats the specified number_rows as a file header and doesn't load them. Use this option to skip file headers in all files in a parallel load.

Advanced settings: Data conversion parameters | Load operations