Ex 3

Implement a simple map-reduce code for the wordcount problem using Java/Python. (Create the jar files and run the code using HDFS.)

```
su - hadoop

mkdir ~/WordCountProject
cd ~/WordCountProject

nano WordCountMapper.java
```

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    @Override
    public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
        String[] words = value.toString().split("\\s+");
        for (String str : words) {
            word.set(str);
            context.write(word, one);
        }
    }
}
```

```
nano WordCountReducer.java
```

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
```

```
public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

    @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        context.write(key, new IntWritable(sum));
    }
}
```

```
nano WordCount.java
```

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");

        job.setJarByClass(WordCount.class);
        job.setMapperClass(WordCountMapper.class);
        job.setCombinerClass(WordCountReducer.class);
        job.setReducerClass(WordCountReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

```
mkdir wordcount_classes

javac -classpath $(hadoop classpath) -d wordcount_classes WordCountMapper.java WordCountReducer.java WordCount.java

jar -cvf WordCount.jar -C wordcount_classes/ .

start-dfs.sh
start-yarn.sh

echo "Hello Hadoop Hello MapReduce" > input.txt
```

whoami (find username)

(hadoop - username)

```
hdfs dfs -mkdir /user/hadoop/input

hdfs dfs -put input.txt /user/hadoop/input

hadoop jar WordCount.jar WordCount /user/hadoop/input /user/hadoop/output

hdfs dfs -cat /user/hadoop/output/part-r-00000
```

Delete existing dic -error

```
hdfs dfs -rm -r /user/hadoop/output

hadoop jar WordCount.jar WordCount /user/hadoop/input /user/hadoop/output

hdfs dfs -cat /user/hadoop/output/part-r-00000
```

Python version

```
mkdir ~/WordCountPythonProject

nano mapper.py
```

```
# mapper.py
import sys

# Input comes from standard input (line by line)
for line in sys.stdin:
    line = line.strip()  # Remove leading and trailing whitespace
    words = line.split()  # Split line into words

    # Output each word with a count of 1
    for word in words:
        print(f"{word}\t1")
```

```
nano reducer.py
```

```
# reducer.py
import sys

current_word = None
current_count = 0
word = None

# Input comes from standard input
for line in sys.stdin:
    line = line.strip()
    word, count = line.split('\t', 1)

    try:
        count = int(count)
    except ValueError:
        continue

    # Sum counts for each word
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print(f"{current_word}\t{current_count}")
        current_word = word
        current_count = count

# Output the last word
if current_word == word:
    print(f"{current_word}\t{current_count}")
```

```
start-dfs.sh
start-yarn.sh

(Delete file output)
hdfs dfs -rm /user/hadoop/input/input.txt

echo "Hello Hadoop Hello MapReduce" > input1.txt
hdfs dfs -put input1.txt /user/hadoop/input

 hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-*.jar \
    -input /user/hadoop/input \
    -output /user/hadoop/output \
    -mapper "python3 mapper.py" \
    -reducer "python3 reducer.py"


hdfs dfs -cat /user/hadoop/output/part-00000
```

Ex 4

1. Implement map reduce for NCDC weather dataset using Hadoop -fine the max and min temperature.

```
start-dfs.sh
start-yarn.sh
```

2. Implement Apriori algorithm using map reduce paradigm.

```
start-dfs.sh
start-yarn.sh

# Create a directory for your input data in HDFS
hadoop fs -mkdir -p /user/hadoop/input

# Create the transactions file
```

```
echo -e "milk,bread,butter\nbread,butter,juice\nmilk,juice\nbread,milk,juice" > transactions.txt

# Upload the input file to HDFS
hadoop fs -put transactions.txt /user/hadoop/input

mkdir AprioriMR
cd AprioriMR
mkdir src
cd src
```

AprioriMR.java (create inside src)

```
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;
import java.util.StringTokenizer;

public class AprioriMR {

  public static class ItemsetMapper extends Mapper<Object, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text item = new Text();

    public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString(), ",");
      while (itr.hasMoreTokens()) {
        item.set(itr.nextToken().trim());
        context.write(item, one);
      }
    }
  }

  public static class ItemsetReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    private int minSupport;
```

```
        @Override
        protected void setup(Context context) {
            Configuration conf = context.getConfiguration();
            minSupport = conf.getInt("minSupport", 2); // Example threshold
        }

        public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            if (sum >= minSupport) {
                context.write(key, new IntWritable(sum));
            }
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        conf.setInt("minSupport", 2);  // Set minimum support here
        Job job = Job.getInstance(conf, "apriori");
        job.setJarByClass(AprioriMR.class);
        job.setMapperClass(ItemsetMapper.class);
        job.setCombinerClass(ItemsetReducer.class);
        job.setReducerClass(ItemsetReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

```
# Navigate to the source directory
cd src

# Compile the code
javac -classpath `hadoop classpath` -d ../ AprioriMR.java

# Go back to the project root
cd ..
```

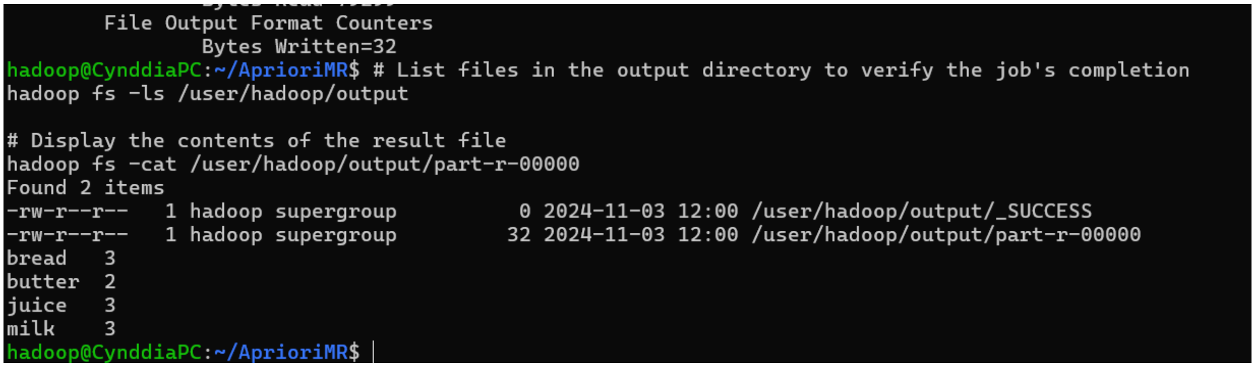
```
# Package the compiled code into a JAR file
jar -cvf AprioriMR.jar -C . .

# Run the Hadoop job
hadoop jar AprioriMR.jar AprioriMR /user/hadoop/input /user/hadoop/output

# List the output files in HDFS
hadoop fs -ls /user/hadoop/output

# View the results
hadoop fs -cat /user/hadoop/output/part-r-00000
```

<u>(Troubleshooting)</u>

```
# Remove input and output directories from HDFS
hadoop fs -rm -r /user/hadoop/input
hadoop fs -rm -r /user/hadoop/output

# List files in the src directory to check if AprioriMR.java is there
ls src

# Navigate to the src directory
cd src

# Compile the code using Hadoop's classpath
javac -classpath `hadoop classpath` -d ../ AprioriMR.java
```

<u>(if already output file is there)</u>

```
hadoop fs -rm -r /user/hadoop/output


hadoop jar AprioriMR.jar AprioriMR /user/hadoop/input /user/hadoop/output
```



```
        File Output Format Counters
                Bytes Written=32
hadoop@CynddiaPC:~/AprioriMR$ # List files in the output directory to verify the job's completion
hadoop fs -ls /user/hadoop/output

# Display the contents of the result file
hadoop fs -cat /user/hadoop/output/part-r-00000
Found 2 items
-rw-r--r--   1 hadoop supergroup          0 2024-11-03 12:00 /user/hadoop/output/_SUCCESS
-rw-r--r--   1 hadoop supergroup         32 2024-11-03 12:00 /user/hadoop/output/part-r-00000
bread   3
butter  2
juice   3
milk    3
hadoop@CynddiaPC:~/AprioriMR$ 
```

Ex 5

Installing pyspark + jupyter

How to Run PySpark on Jupyter Notebook | phoenixNAP KB

Pyspark with jupyter
https://chatgpt.com/share/6727240e-8e94-8013-a25c-b04e1d94de0d

1. Run the wordcount program that you did using hadoop usingpyspark.

```
import os
import findspark
from pyspark.sql import SparkSession

# Set environment variables
os.environ['SPARK_HOME'] = "/home/hadoop/.local/lib/python3.10/site-packages/pyspark"  # Adjust this if necessary
os.environ['HADOOP_HOME'] = "/path/to/hadoop"  # If you have Hadoop installed
os.environ['PYSPARK_PYTHON'] = "python3"  # Or "python" depending on your setup

# Initialize findspark to find the Spark installation
findspark.init()

# Create a Spark session
spark = SparkSession.builder \
    .appName("WordCount") \
    .getOrCreate()

# Print Spark version to confirm it’s working
print(spark.version)

# Example: Word Count
text_data = ["Hello world", "Hello Spark", "Hello Jupyter"]
rdd = spark.sparkContext.parallelize(text_data)
```

```
word_counts = rdd.flatMap(lambda line: line.split(" ")) \
              .map(lambda word: (word, 1)) \
              .reduceByKey(lambda a, b: a + b)

# Collect and print results
for word, count in word_counts.collect():
   print(f"{word}: {count}")
```

Movielens dataset - find out for each movie, how are the ratings distributed

(create txt movielens dataset in jupyter directory - not local file)

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count

# Create a Spark session
spark = SparkSession.builder \
   .appName("MovieLens Ratings Distribution") \
   .getOrCreate()



data_path = "Movielens.txt"
movies_df = spark.read.csv(data_path, sep='\t', inferSchema=True) \
.toDF("user_id", "movie_id", "rating", "timestamp")
# Calculate the distribution of ratings for each movie
rating_distribution = movies_df.groupBy("movie_id", "rating") \
.agg(count("rating").alias("rating_count")) \
.orderBy("movie_id", "rating")


rating_distribution.show()
```

Ex 6

1. Use the "friends_test" dataset. Col1 is ID, Col2 is name, Col 3 is Age, Col 4 is num of friends. Understand mapvalues function of RDD in spark and find the average number of friends for each unique age present in the dataset.

```
import pyspark
from pyspark.sql import SparkSession
spark =
SparkSession.builder.master('local').appName('friends_dataset').getOrCreat
e()
sc=spark.sparkContext

# Load dataset
data_path='friends_test.csv'
rdd=sc.textFile(data_path)

age_friends_rdd = rdd.map(lambda line: line.split(',')) \
                     .map(lambda cols: (int(cols[2]), (int(cols[3]), 1)))

# Sum up num_friends and count for each age
age_friends_totals = age_friends_rdd.reduceByKey(lambda a, b: (a[0] +
b[0], a[1] + b[1]))

# Calculate the average number of friends for each age
average_friends_by_age = age_friends_totals.mapValues(lambda total:
total[0] / total[1])

# Collect and display the results
results = average_friends_by_age.collect()
for age, avg_friends in results:
    print(f"Age: {age}, Average Friends: {avg_friends:.2f}")
```

2. Use the "temp.csv" dataset. Column headers are present in the dataset. Understand filter operations and filter out only the "TMIN" values from the "desc" column. With the resultant data (RDD) find the following: a. Minimum temperature (overall) b. Minimum temperature for every ItemID c. Minimum temperature for every StationID.

Use the same dataset, filter only "TMAX" column and find the maximum temperatures just like the ones mentioned above.

```
from pyspark.sql import SparkSession

# Initialize Spark session
spark = SparkSession.builder \
    .appName("Temp_dataset") \
    .getOrCreate()
# Load dataset
data_path = "temp.csv"  # Replace with actual path
rdd = spark.sparkContext.textFile(data_path)
# Extract the header
header = rdd.first()
data_rdd = rdd.filter(lambda row: row != header)  # Remove the header
# Split each row by comma and convert to (StationID, ItemID, desc, temp)
format
data_rdd = data_rdd.map(lambda line: line.split(",")) \
                   .map(lambda cols: (cols[0], cols[1], cols[2],
float(cols[3])))  # assuming temp is in column 4
# 1. Filter for "TMIN" and find minimum temperatures
tmin_rdd = data_rdd.filter(lambda x: x[2] == "TMIN")

# a. Overall minimum temperature
overall_min_tmin = tmin_rdd.map(lambda x: x[3]).min()

# b. Minimum temperature for each ItemID
min_temp_by_item = tmin_rdd.map(lambda x: (x[1], x[3])) \
                           .reduceByKey(lambda a, b: min(a, b))
```

```python
# c. Minimum temperature for each StationID
min_temp_by_station = tmin_rdd.map(lambda x: (x[0], x[3])) \
                              .reduceByKey(lambda a, b: min(a, b))
# Display results for TMIN
print(f"Overall minimum temperature (TMIN): {overall_min_tmin}")
print("Minimum temperature for each ItemID (TMIN):")
for item, min_temp in min_temp_by_item.collect():
    print(f"ItemID: {item}, Min Temp: {min_temp}")

print("Minimum temperature for each StationID (TMIN):")
for station, min_temp in min_temp_by_station.collect():
    print(f"StationID: {station}, Min Temp: {min_temp}")

# 2. Filter for "TMAX" and find maximum temperatures
tmax_rdd = data_rdd.filter(lambda x: x[2] == "TMAX")

# a. Overall maximum temperature
overall_max_tmax = tmax_rdd.map(lambda x: x[3]).max()

# b. Maximum temperature for each ItemID
max_temp_by_item = tmax_rdd.map(lambda x: (x[1], x[3])) \
                           .reduceByKey(lambda a, b: max(a, b))

# c. Maximum temperature for each StationID
max_temp_by_station = tmax_rdd.map(lambda x: (x[0], x[3])) \
                              .reduceByKey(lambda a, b: max(a, b))

# Display results for TMAX
print(f"Overall maximum temperature (TMAX): {overall_max_tmax}")
print("Maximum temperature for each ItemID (TMAX):")
for item, max_temp in max_temp_by_item.collect():
    print(f"ItemID: {item}, Max Temp: {max_temp}")

print("Maximum temperature for each StationID (TMAX):")
for station, max_temp in max_temp_by_station.collect():
    print(f"StationID: {station}, Max Temp: {max_temp}")
```

Ex 7

Set up a simple Hadoop environment using Docker containers, including at least one NameNode and one DataNode. Ensure the containers are properly configured to interact with each other.After the setup, verify that the Hadoop cluster is operational by running a simple HDFS file operation (e.g., uploading a fileto HDFS).

Check and install docker

```
docker --version
```

```
sudo apt install docker.io
sudo docker run hello-world
```

```
docker network create hadoop-net
```

```
docker pull bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8
docker pull bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8
```

```
docker run -d \
  --name namenode \
  --network hadoop-net \
  -e CLUSTER_NAME="my-hadoop-cluster" \
  -e CORE_CONF_fs_defaultFS=hdfs://namenode:9000 \
  bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8
```

```
docker run -d \
  --name datanode \
  --network hadoop-net \
  -e CORE_CONF_fs_defaultFS=hdfs://namenode:9000 \
  bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8
```

```
docker ps
```

```
docker logs namenode
docker logs datanode
```

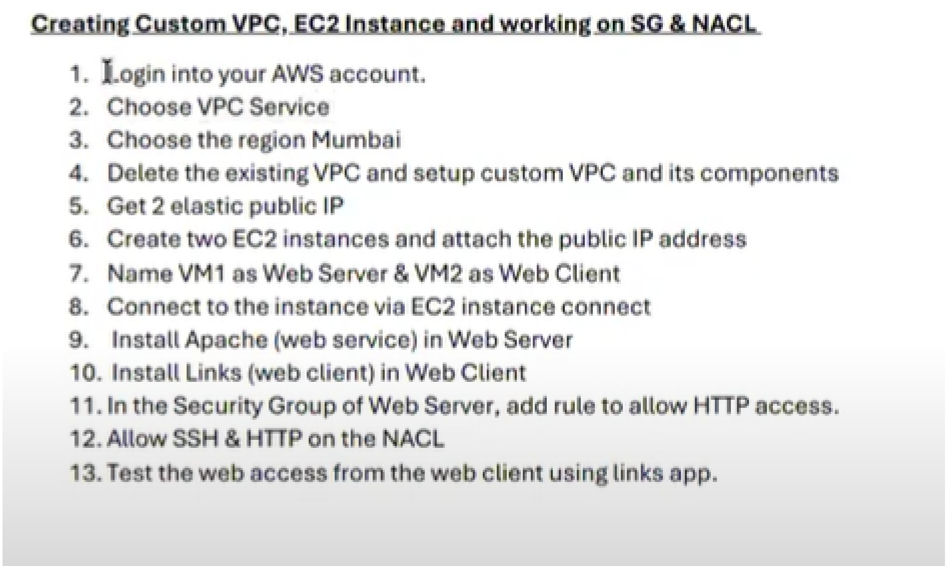
(Run simple hdfs file system)

```
echo "Hello Hadoop!" > hello.txt

docker cp hello.txt namenode:/hello.txt

docker exec -it namenode hdfs dfs -mkdir -p /user/root
docker exec -it namenode hdfs dfs -put /hello.txt /user/root/

docker exec -it namenode hdfs dfs -ls /user/root/

docker exec -it namenode hdfs dfs -cat /user/root/hello.txt
```

(clean up)

```
docker rm -f namenode datanode
docker network rm hadoop-net
```

AWS



Creating Custom VPC, EC2 Instance and working on SG & NACL

1. Login into your AWS account.
2. Choose VPC Service
3. Choose the region Mumbai
4. Delete the existing VPC and setup custom VPC and its components
5. Get 2 elastic public IP
6. Create two EC2 instances and attach the public IP address
7. Name VM1 as Web Server & VM2 as Web Client
8. Connect to the instance via EC2 instance connect
9. Install Apache (web service) in Web Server
10. Install Links (web client) in Web Client
11. In the Security Group of Web Server, add rule to allow HTTP access.
12. Allow SSH & HTTP on the NACL
13. Test the web access from the web client using links app.

VPC and EC2

VPC : snu-dc-vpc
Ipv4 : 192.168.0.0/16

=> Connect subnet to vpc created

Subnet name: public-subnet
Availability zone: Mumbai
IPv4 subnet block (2nd one) : 192.168.1.0/24

=> Internet gateway

Name tag: snuc-dc-igw

=> Connect ig to vpc created
=> Route tables

Click on route table id -> edit routes -> add routes
0.0.0.0/0 - Internet gateway -> igw (auto-completion)

(chk point: completed setting up of vpc and it’s components)

=> Elastic IP

Allocate ip address (create web server and web client (rename after creation)

=> EC2

Go to instances -> launch instances
Number of instances: 2
Name: VMs
Quick start -> change to ubuntu
Key pair -> proceed without key pair (not)
Now launch instance

-> go to instance
Rename to web server and web client

-> go to elastic ip

Click on web server ip add -> associate elastic ip add with it
Instance: choose web server

Do same for web client

(chk point: create two ec2 instances and attach public ip address)

-> go to instances
Click web server and connect -> connect instance

(if error ipv4 not public check again and refresh then connect)

=>New tab for terminal opens up(ec2-instance-connect)

```
ping 8.8.8.8
sudo apt update
sudo apt install apache2
service apache2 status (ctrl c to escape)
```

=> connect web client to instance
(tab should pop up)

```
ping 8.8.8.8
sudo apt update
sudo apt install links
clear
```

(chck point: Install apache and links)

Click on web server ip add -> go to security
Click security group id
Click on edit inbound rules

-> add rule -> http -> source -> anywhere ipv4

(chck point: HTTP access)

Go to instances -. Click link for web server go to security

=> nacl is in vpc

Vpc ->network security -> nacl

Click on nacl id
Click on edit inbound rules

Remove the existing rule
Add new rule

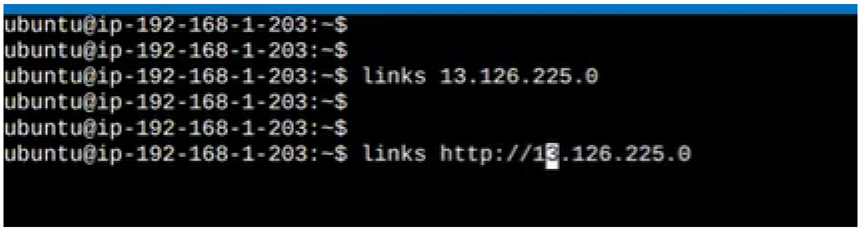
Type : SSH (20)

New rule -> 101 -> HTTP(80) -> save changes

(chck point: test the web access)

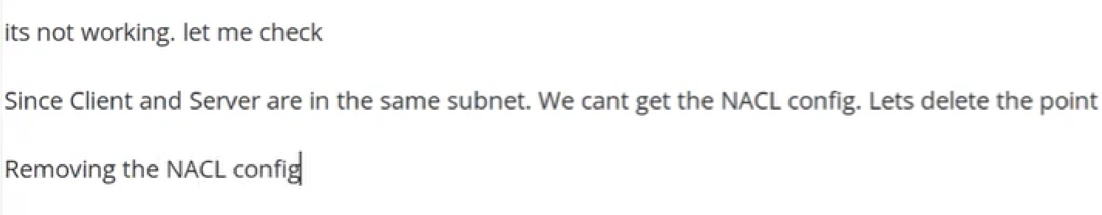
Go to ec2 interface for web client (the cleared one)

links (the public ip of web server -> in interface terminal)



```
ubuntu@ip-192-168-1-203:~$
ubuntu@ip-192-168-1-203:~$
ubuntu@ip-192-168-1-203:~$ links 13.126.225.0
ubuntu@ip-192-168-1-203:~$
ubuntu@ip-192-168-1-203:~$
ubuntu@ip-192-168-1-203:~$ links http://13.126.225.0
```

Now removing the nacl from



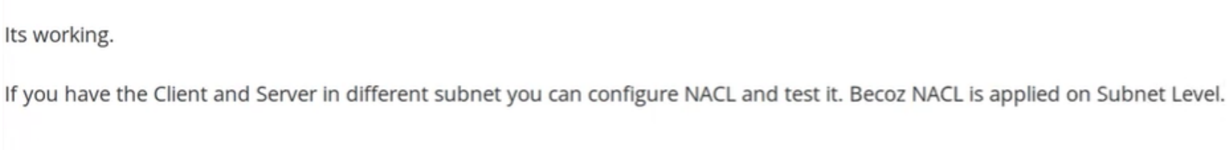
its not working. let me check

Since Client and Server are in the same subnet. We cant get the NACL config. Lets delete the point

Removing the NACL config

Remove the created inbound rules (100,101)
Add new rule -> 100-> all traffic -> save changes

Now it will work



Its working.

If you have the Client and Server in different subnet you can configure NACL and test it. Becoz NACL is applied on Subnet Level.

(1. VPC & EC2 Lab : https://youtu.be/AsSQb--MNXA (no audio))

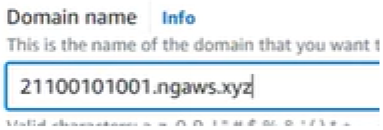
Route 53 Labs

Open route 53

Go to dashboard -> create hosted zone

=> inside hosted zone

Domain name:



Domain name Info

This is the name of the domain that you want t

21100101001.ngaws.xyz

Then create it

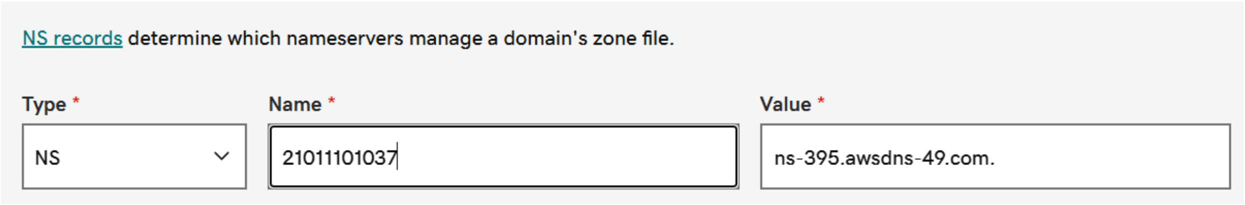
Go to go daddy domain

Log in

Now copy (.com) value/route traffic from route 53

-> add new record in godaddy

-> type: NS -> value (copied .com)

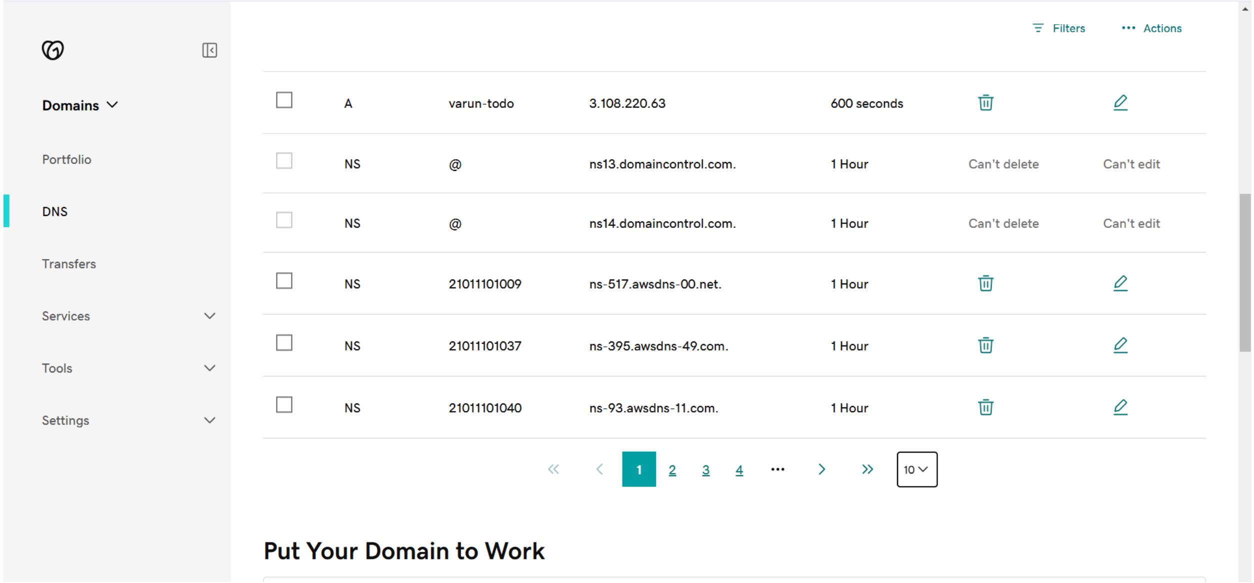


NS records determine which nameservers manage a domain's zone file.

Type * NS

Name * 21011101037

Value * ns-395.awsdns-49.com.



□	A	varun-todo	3.108.220.63	600 seconds		
□	NS	@	ns13.domaincontrol.com.	1 Hour	Can't delete	Can't edit
□	NS	@	ns14.domaincontrol.com.	1 Hour	Can't delete	Can't edit
□	NS	21011101009	ns-517.awsdns-00.net.	1 Hour		
□	NS	21011101037	ns-395.awsdns-49.com.	1 Hour		
□	NS	21011101040	ns-93.awsdns-11.com.	1 Hour		

Put Your Domain to Work

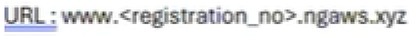
Now go back

Go to hosted zones -> create records

Record name: www
Ip address: use ip of web server (public ipv4) for instances (created prev)

Create record

Go to web client terminal -> links url

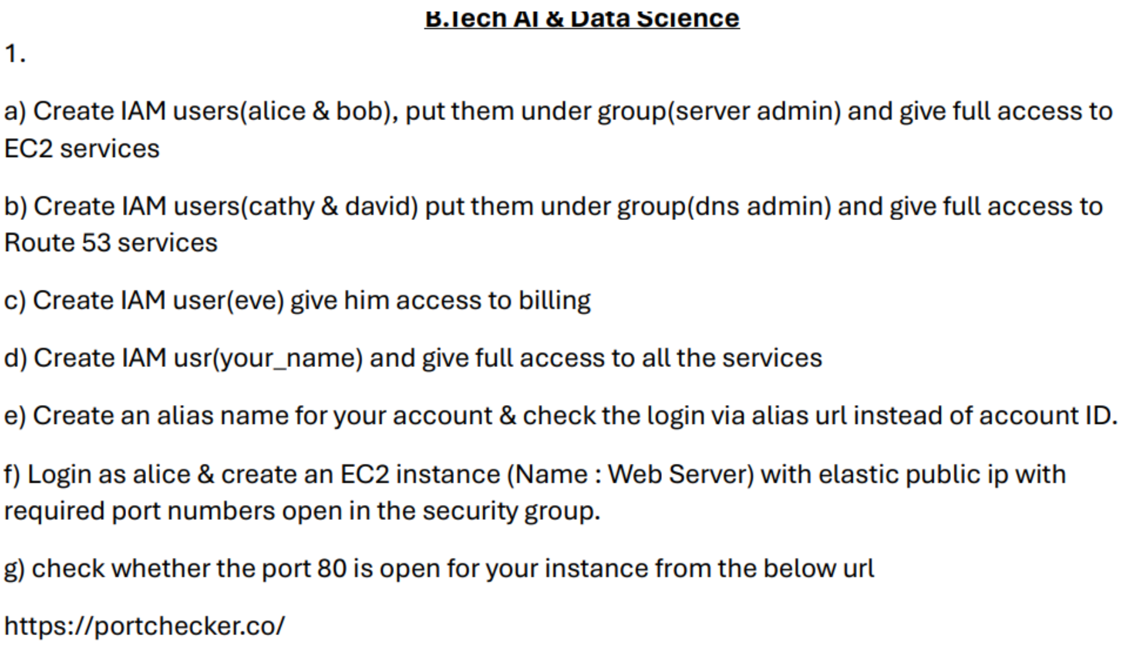


URL : www.<registration_no>.ngaws.xyz

nslookup url

(https://www.youtube.com/watch?v=-ndsfa-6GMI)

IAM :



B.Tech AI & Data Science

1.

a) Create IAM users(alice & bob), put them under group(server admin) and give full access to EC2 services

b) Create IAM users(cathy & david) put them under group(dns admin) and give full access to Route 53 services

c) Create IAM user(eve) give him access to billing

d) Create IAM usr(your_name) and give full access to all the services

e) Create an alias name for your account & check the login via alias url instead of account ID.

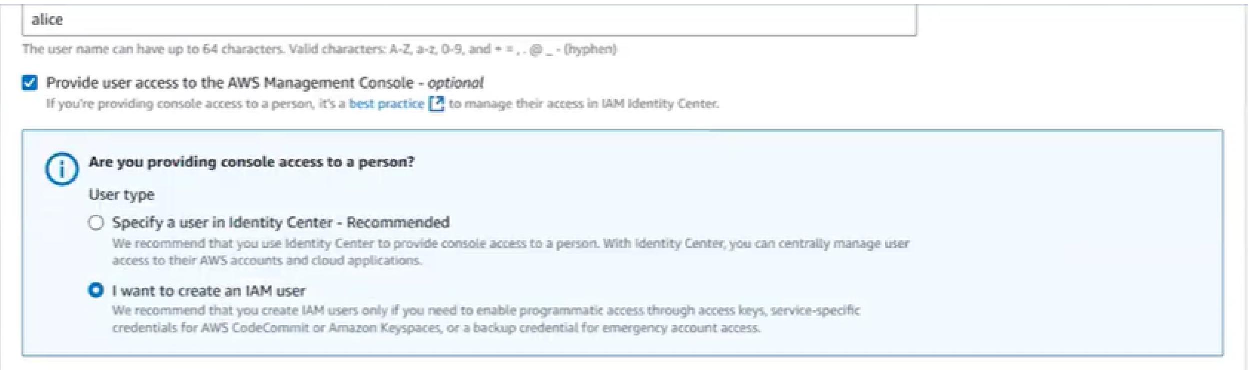
f) Login as alice & create an EC2 instance (Name : Web Server) with elastic public ip with required port numbers open in the security group.

g) check whether the port 80 is open for your instance from the below url

https://portchecker.co/

=> search for IAM
-> go to users

-> go to create user
alice -> provide user access click -> Specify user details



alice

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

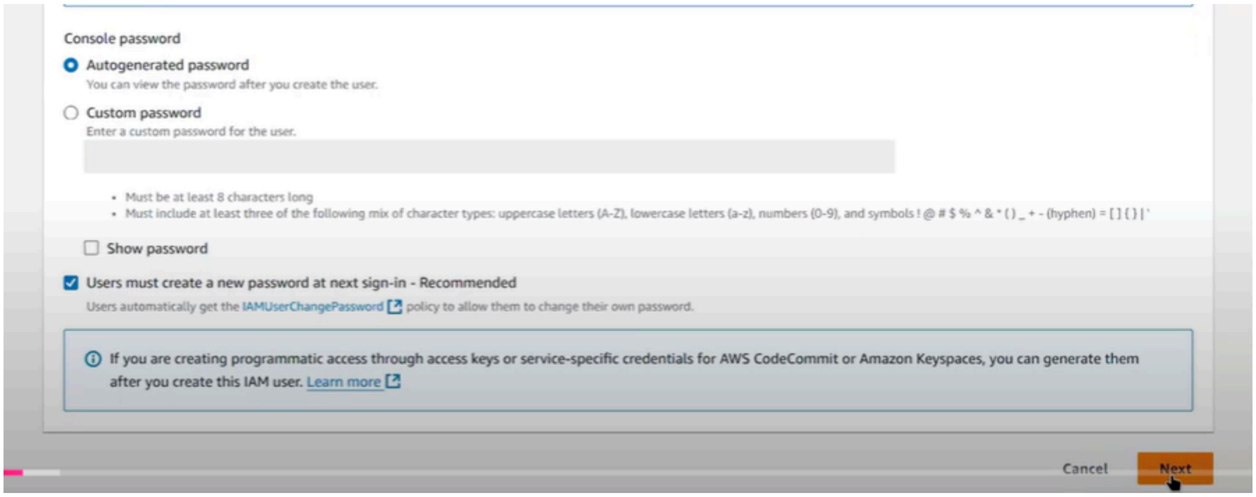
☑ Provide user access to the AWS Management Console - *optional*
If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.

Are you providing console access to a person?

User type

○ Specify a user in Identity Center - Recommended
We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.

● I want to create an IAM user
We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keyspaces, or a backup credential for emergency account access.



Console password

● Autogenerated password
You can view the password after you create the user.

○ Custom password
Enter a custom password for the user.

- Must be at least 8 characters long
- Must include at least three of the following mix of character types: uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9), and symbols ! @ # $ % ^ & * ( ) _ + - (hyphen) = [ ] { } | '

☐ Show password

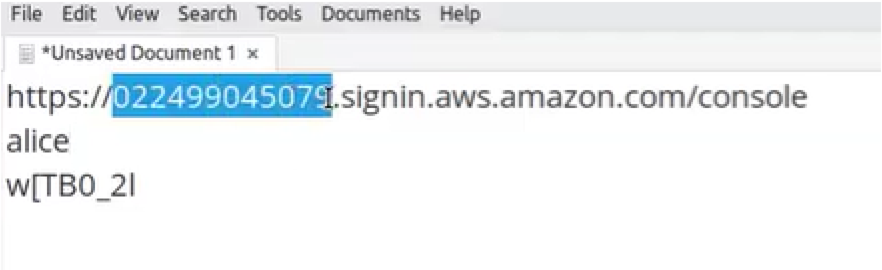
☑ Users must create a new password at next sign-in - Recommended
Users automatically get the IAMUserChangePassword policy to allow them to change their own password.

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. Learn more

Set permissions -> give next
Click on creator user after getting user details

Copy the password and Console sign-in URL

Copy your account id from top right (???)



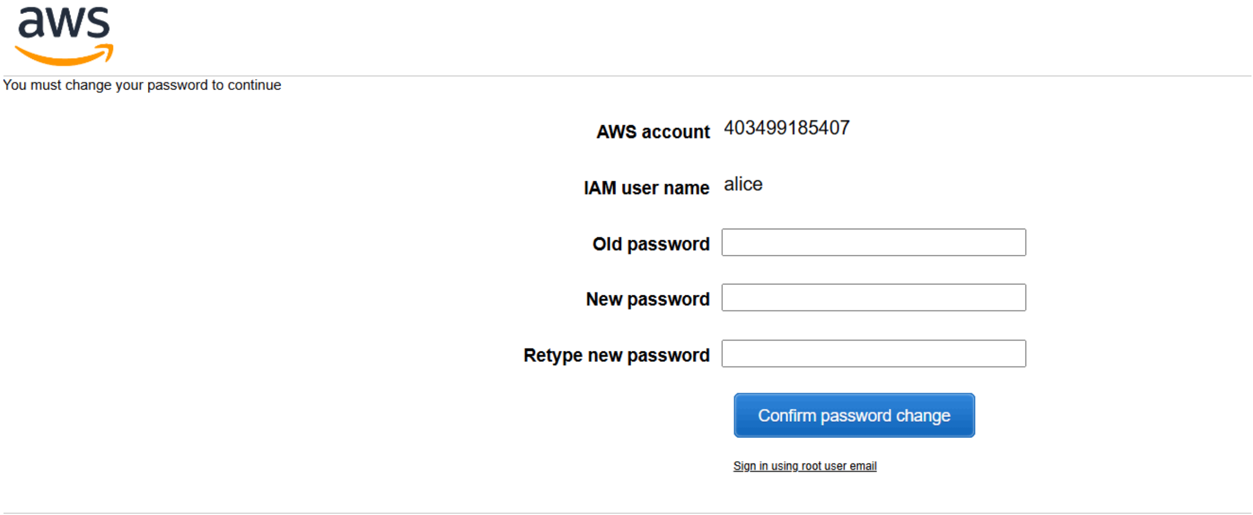
```
https://022499045079.signin.aws.amazon.com/console
alice
w[TB0_2l
```

Return to users list

Go to user groups

Now go to chrome and paste alice's url

Now fill alice name and password (copied)



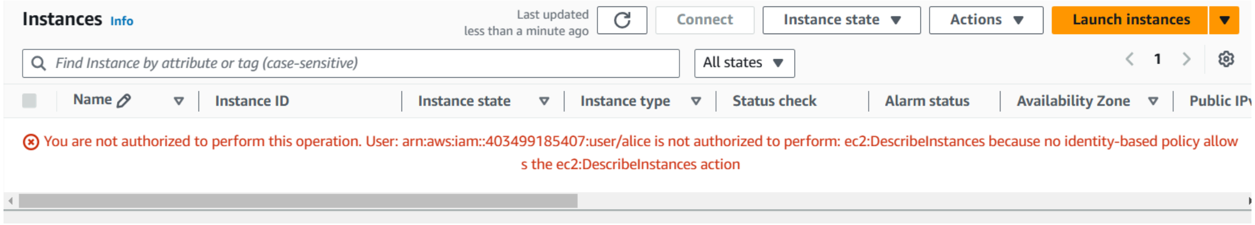
You must change your password to continue

AWS account	403499185407
IAM user name	alice
Old password	
New password	
Retype new password	

New password : old password+1

(put alice in a diff tab as you will be signed out of your actual account!)

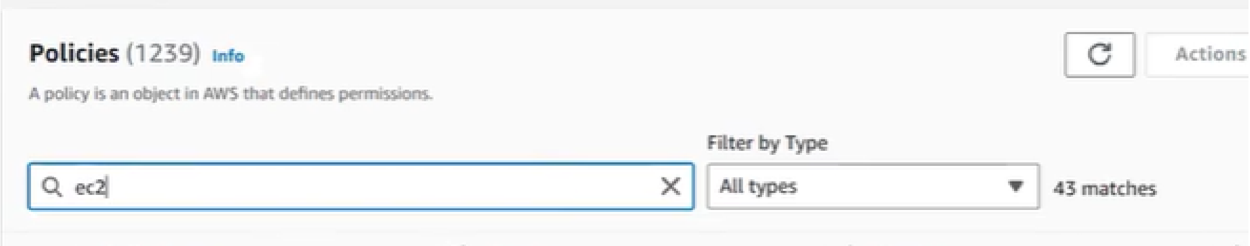
=> search for ec2 on alice and go to instances



You are not authorized to perform this operation. User: arn:aws:iam::403499185407:user/alice is not authorized to perform: ec2:DescribeInstances because no identity-based policy allows the ec2:DescribeInstances action

Now go to Cynddia => IAM

Go to policies



Policies (1239)

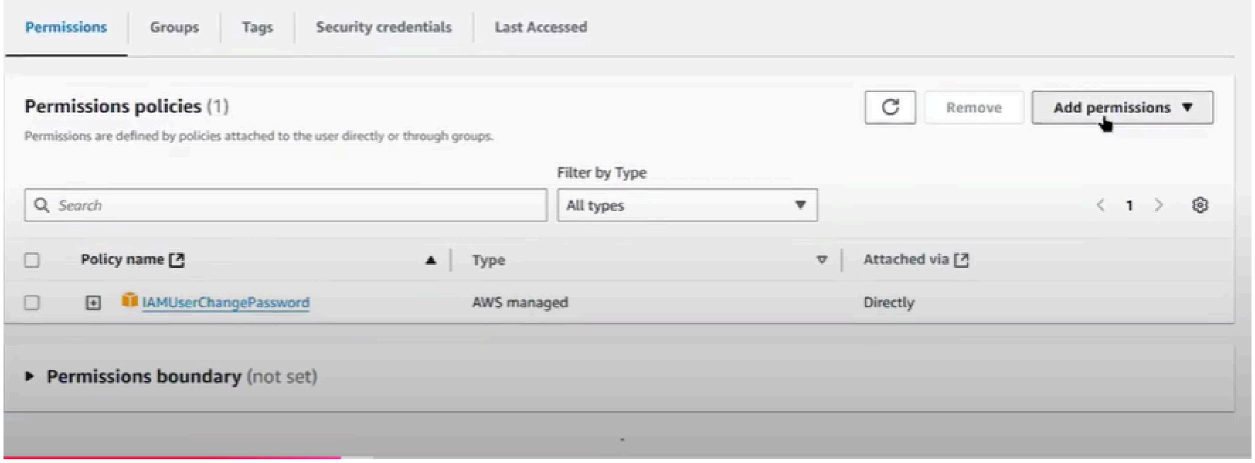
A policy is an object in AWS that defines permissions.

Search: ec2

Filter by Type: All types

43 matches

Go to users -> click on alice
Go to permissions -> add permissions x 2



Permissions policies (1)

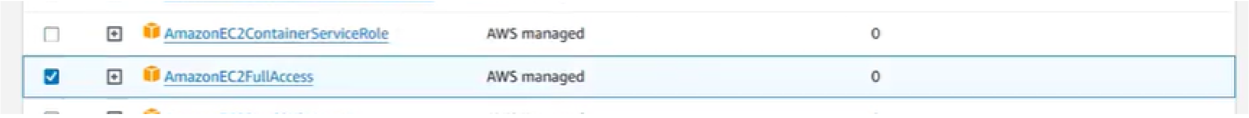
Permissions are defined by policies attached to the user directly or through groups.

Filter by Type: All types

□	Policy name	Type	Attached via
□	IAMUserChangePassword	AWS managed	Directly

▸ **Permissions boundary** (not set)

Attach policies directly

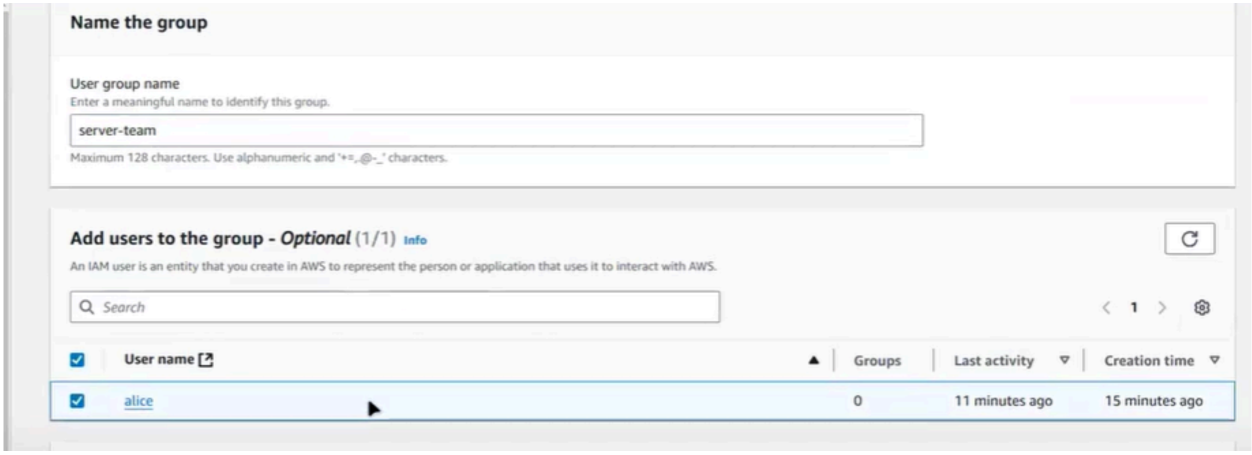


□	AmazonEC2ContainerServiceRole	AWS managed	0
☑	AmazonEC2FullAccess	AWS managed	0

Next -> add permission

Now alice will have access to it

Go to create user group (in cynddia)



Name the group

User group name

Enter a meaningful name to identify this group.

server-team

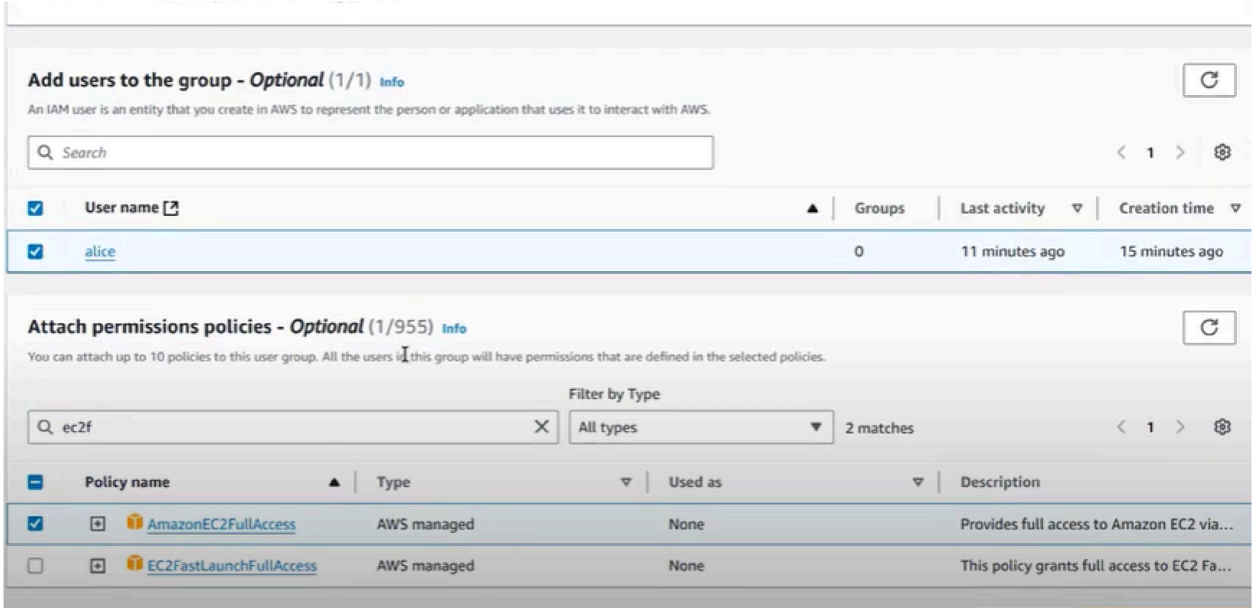
Maximum 128 characters. Use alphanumeric and '+=,.@-_' characters.

Add users to the group - *Optional* (1/1) Info

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

☑	User name	Groups	Last activity	Creation time
☑	alice	0	11 minutes ago	15 minutes ago

Click alice and go for attached permission policies



Add users to the group - *Optional* (1/1) Info

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

☑	User name	Groups	Last activity	Creation time
☑	alice	0	11 minutes ago	15 minutes ago

Attach permissions policies - *Optional* (1/955) Info

You can attach up to 10 policies to this user group. All the users in this group will have permissions that are defined in the selected policies.

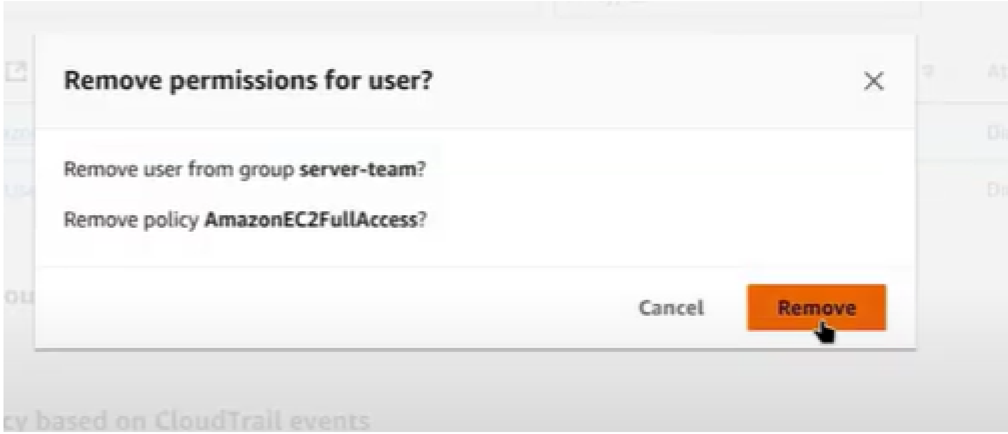
Search: ec2f

Filter by Type: All types

2 matches

☐	Policy name	Type	Used as	Description
☑	AmazonEC2FullAccess	AWS managed	None	Provides full access to Amazon EC2 via...
☐	EC2FastLaunchFullAccess	AWS managed	None	This policy grants full access to EC2 Fa...

Remove the policy

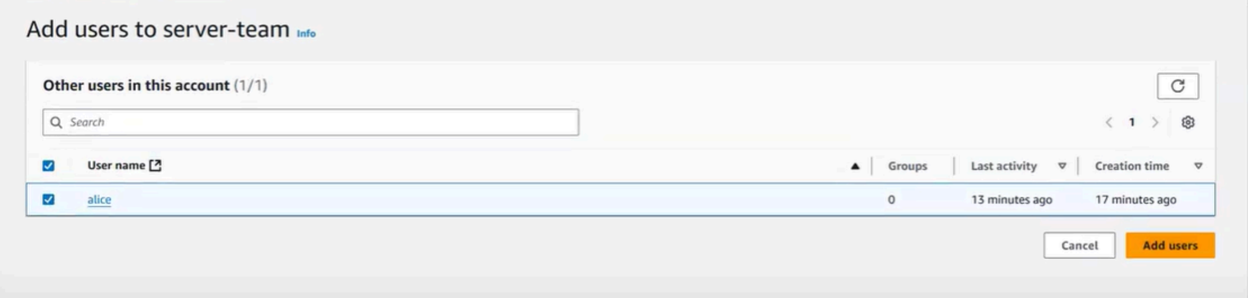


Remove permissions for user?

Remove user from group **server-team**?

Remove policy **AmazonEC2FullAccess**?

Now click server teams in user groups

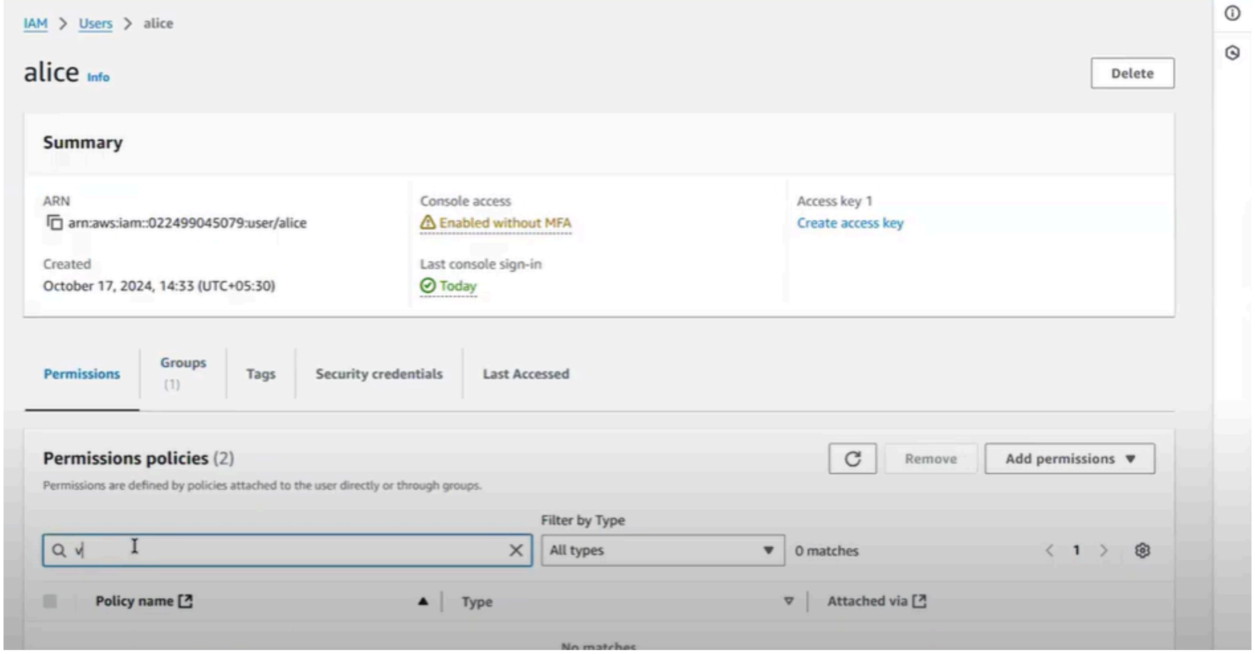


Add users to server-team Info

Other users in this account (1/1)

☑	User name	Groups	Last activity	Creation time
☑	alice	0	13 minutes ago	17 minutes ago

Add permission to alice in Cynddia



IAM > Users > alice

alice Info

Summary

ARN
arn:aws:iam::022499045079:user/alice

Created
October 17, 2024, 14:33 (UTC+05:30)

Console access
Enabled without MFA

Last console sign-in
Today

Access key 1
Create access key

Permissions policies (2)

Permissions are defined by policies attached to the user directly or through groups.

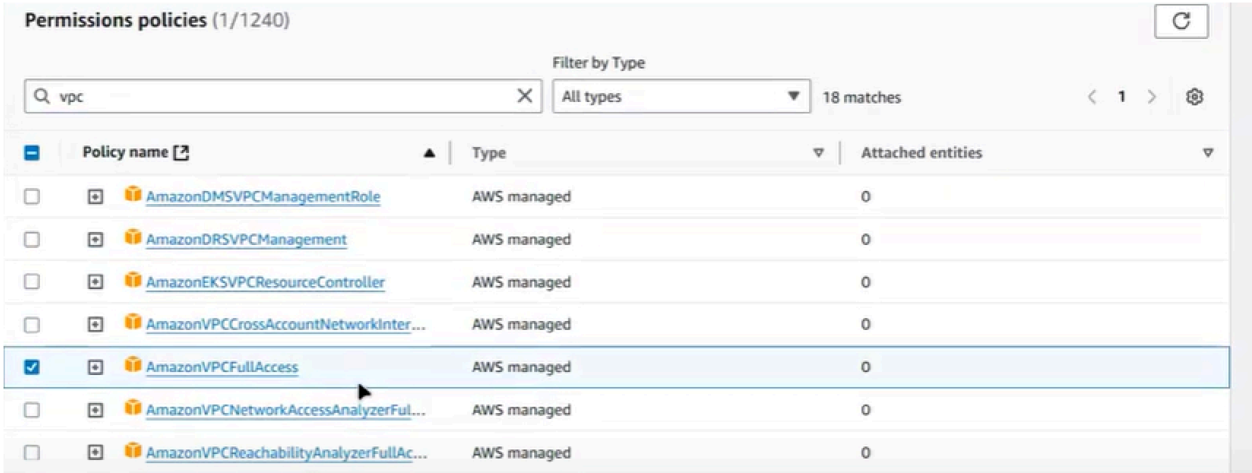
Filter by Type: All types

v

0 matches

☐	Policy name	Type	Attached via

No matches



Permissions policies (1/1240)

Filter by Type: All types

vpc

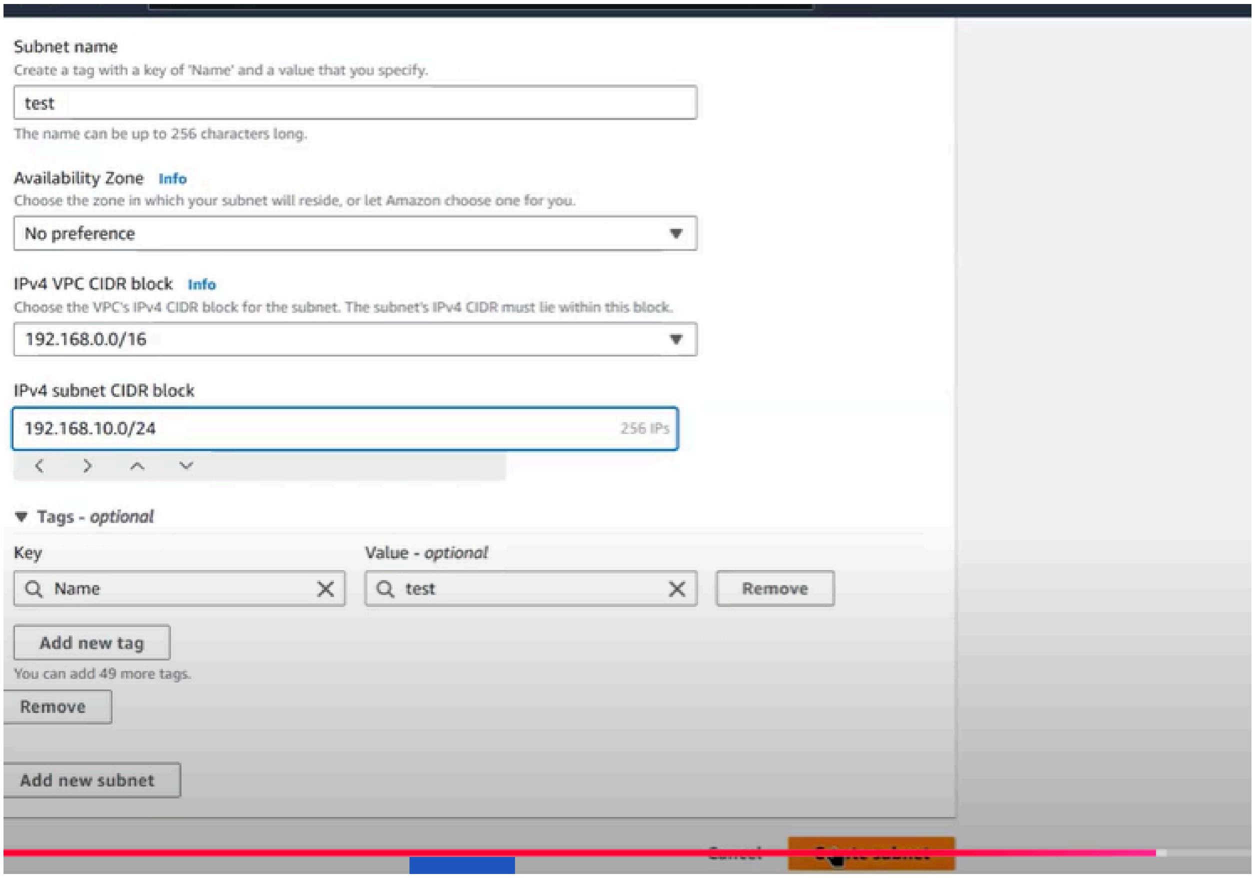
18 matches

		Policy name	Type	Attached entities
☐	⊞	AmazonDMSVPCManagementRole	AWS managed	0
☐	⊞	AmazonDRSVPCManagement	AWS managed	0
☐	⊞	AmazonEKSVPCResourceController	AWS managed	0
☐	⊞	AmazonVPCCrossAccountNetworkInter...	AWS managed	0
☑	⊞	AmazonVPCFullAccess	AWS managed	0
☐	⊞	AmazonVPCNetworkAccessAnalyzerFul...	AWS managed	0
☐	⊞	AmazonVPCReachabilityAnalyzerFullAc...	AWS managed	0

Create bob similarly

Search for vpc in bob

Create subnet



Subnet name
Create a tag with a key of 'Name' and a value that you specify.

test

The name can be up to 256 characters long.

Availability Zone Info
Choose the zone in which your subnet will reside, or let Amazon choose one for you.

No preference

IPv4 VPC CIDR block Info
Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.

192.168.0.0/16

IPv4 subnet CIDR block

192.168.10.0/24 256 IPs

Tags - *optional*

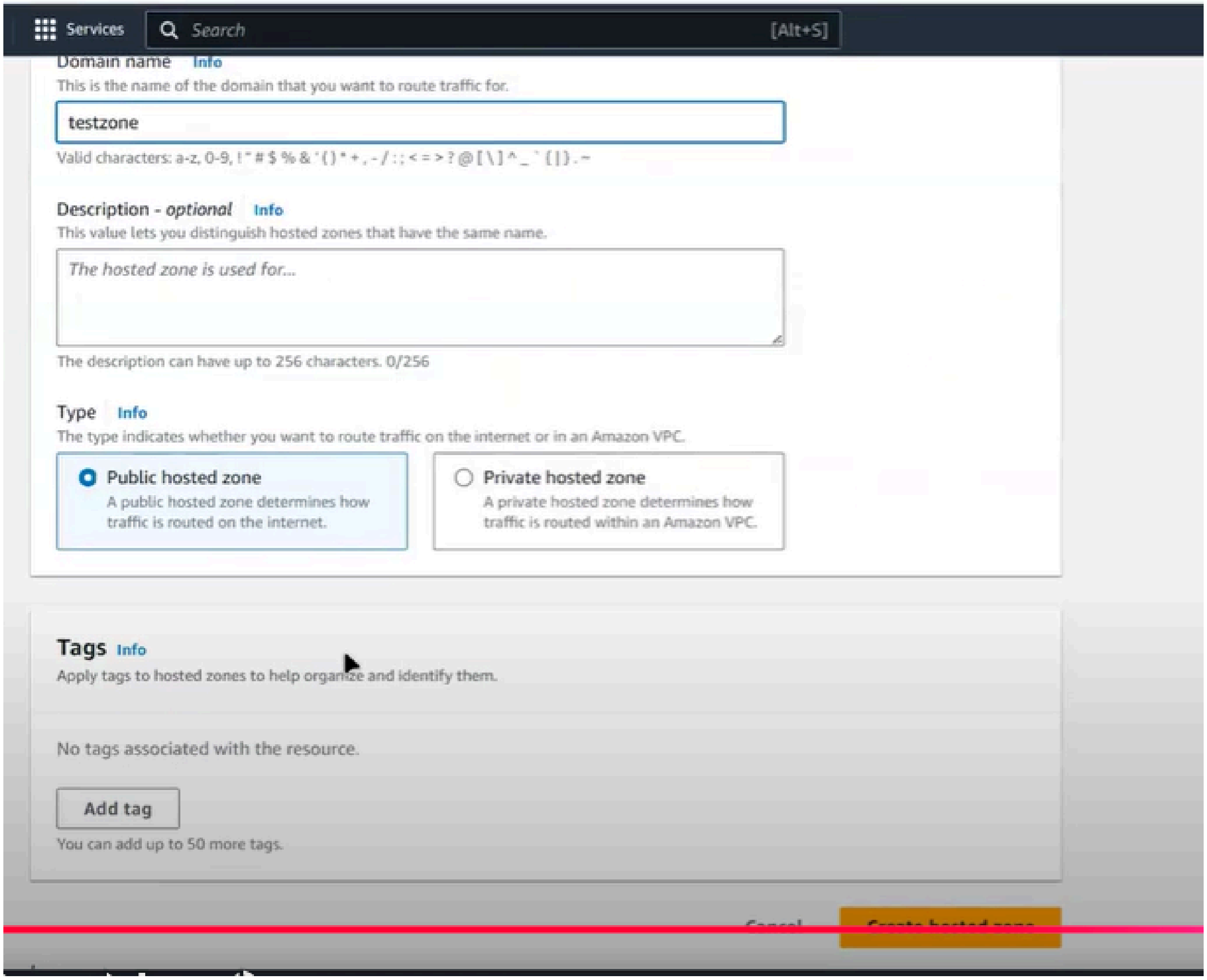
Key	Value - *optional*
Name	test

You can add 49 more tags.

Now to go your user
In user groups go for permission

And now go to route 53 in bob

Create host zone -> getting started



Domain name

This is the name of the domain that you want to route traffic for.

testzone

Valid characters: a-z, 0-9, ! " # $ % & ' ( ) * + , - / : ; < = > ? @ [\ ] ^ _ ` { | } . ~

Description - *optional*

This value lets you distinguish hosted zones that have the same name.

The hosted zone is used for...

The description can have up to 256 characters. 0/256

Type

The type indicates whether you want to route traffic on the internet or in an Amazon VPC.

● Public hosted zone
A public hosted zone determines how traffic is routed on the internet.

○ Private hosted zone
A private hosted zone determines how traffic is routed within an Amazon VPC.

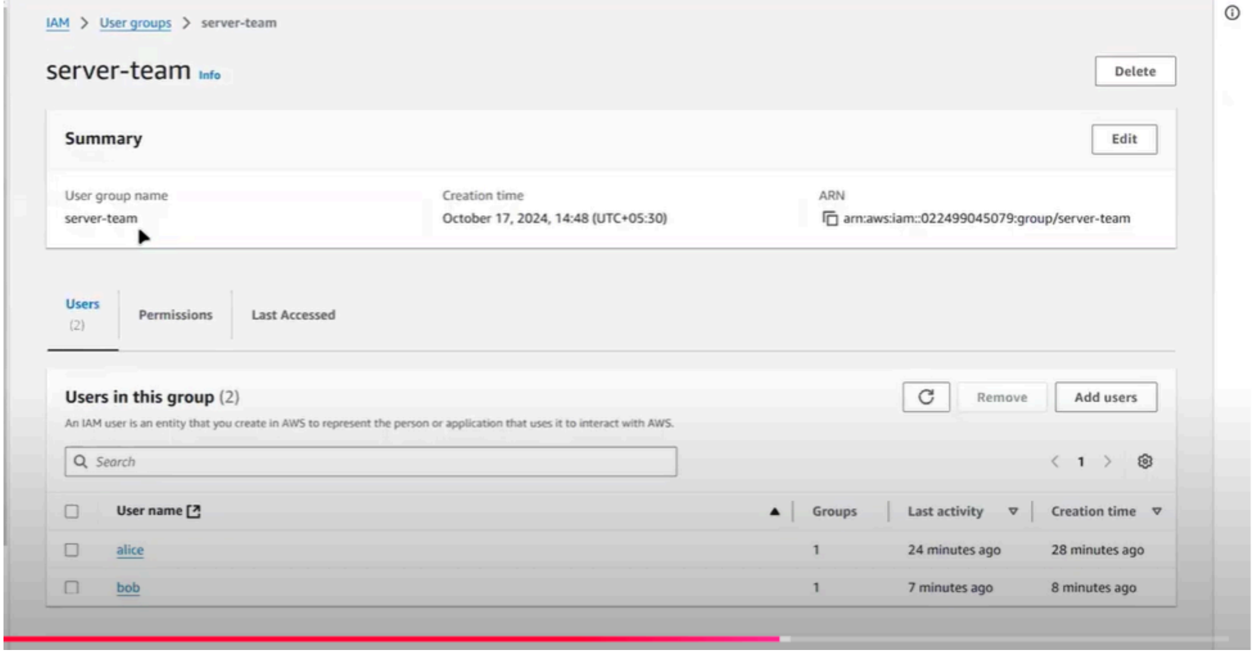
Apply tags to hosted zones to help organize and identify them.

No tags associated with the resource.

You can add up to 50 more tags.

Error , full user needs to give access

Permission in user groups



IAM > User groups > server-team

server-team Info

Delete

Summary

Edit

User group name: server-team

Creation time: October 17, 2024, 14:48 (UTC+05:30)

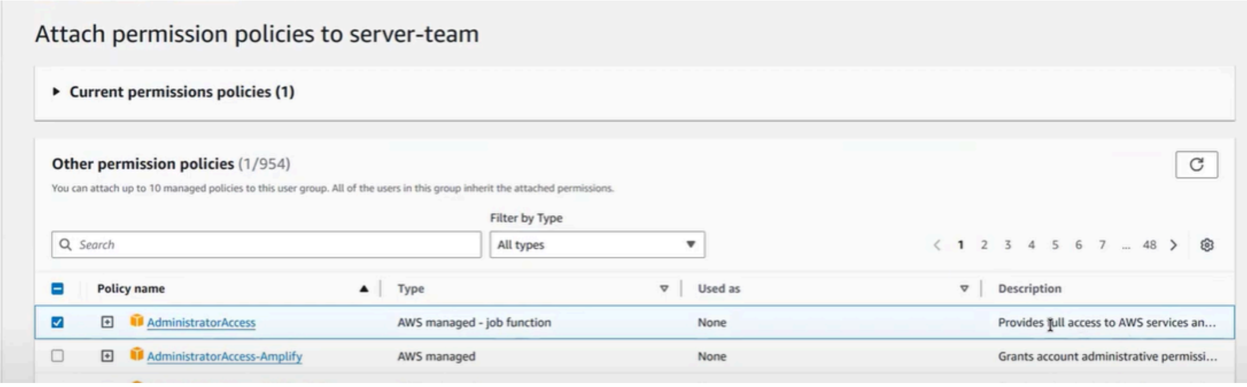
ARN: arn:aws:iam::022499045079:group/server-team

Users (2) | Permissions | Last Accessed

Users in this group (2)

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

User name	Groups	Last activity	Creation time
alice	1	24 minutes ago	28 minutes ago
bob	1	7 minutes ago	8 minutes ago



Attach permission policies to server-team

▸ Current permissions policies (1)

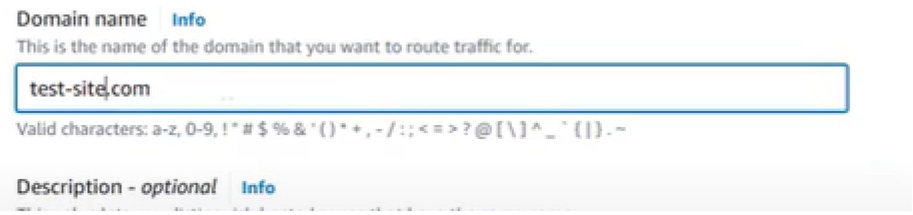
Other permission policies (1/954)

You can attach up to 10 managed policies to this user group. All of the users in this group inherit the attached permissions.

Filter by Type: All types

	Policy name	Type	Used as	Description
☑	AdministratorAccess	AWS managed - job function	None	Provides full access to AWS services an...
□	AdministratorAccess-Amplify	AWS managed	None	Grants account administrative permissi...

Now create host zone for bob



Domain name Info

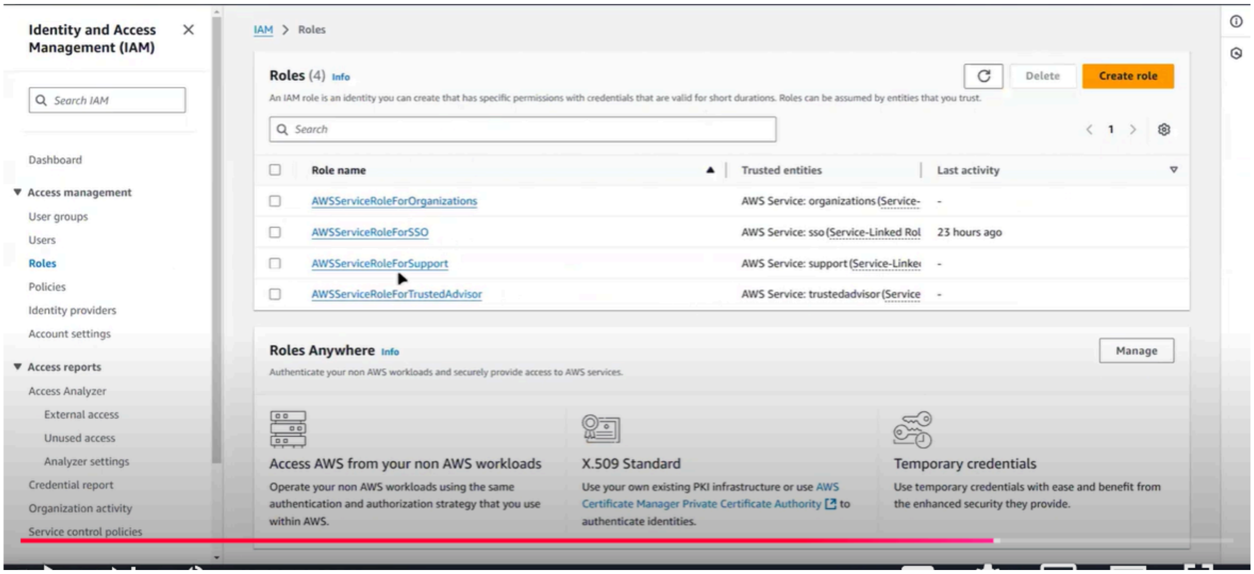
This is the name of the domain that you want to route traffic for.

test-site.com

Valid characters: a-z, 0-9, ! " # $ % & ' ( ) * + , - / : ; < = > ? @ [\ ] ^ _ ` { | } . ~

Description - *optional* Info

Create role in your user id



Roles (4) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

□	Role name	Trusted entities	Last activity
□	AWSServiceRoleForOrganizations	AWS Service: organizations (Service-	-
□	AWSServiceRoleForSSO	AWS Service: sso (Service-Linked Rol	23 hours ago
□	AWSServiceRoleForSupport	AWS Service: support (Service-Linke	-
□	AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service	-

Roles Anywhere Info

Authenticate your non AWS workloads and securely provide access to AWS services.

Access AWS from your non AWS workloads

Operate your non AWS workloads using the same authentication and authorization strategy that you use within AWS.

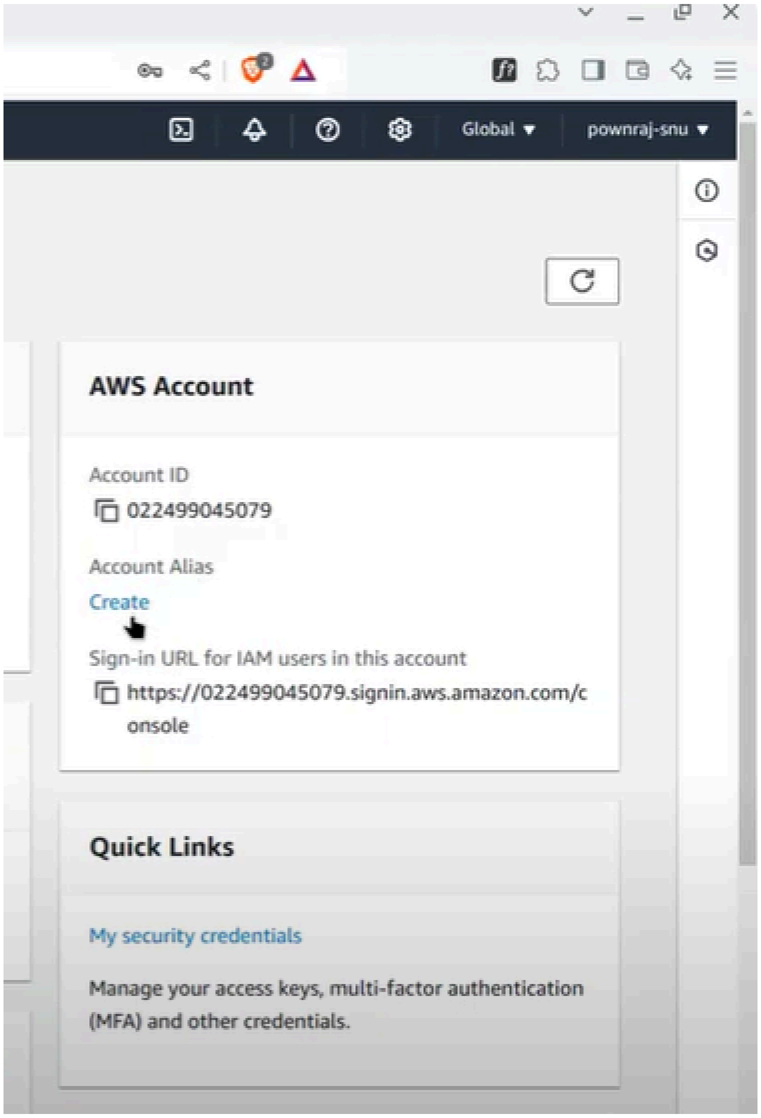
X.509 Standard

Use your own existing PKI infrastructure or use AWS Certificate Manager Private Certificate Authority to authenticate identities.

Temporary credentials

Use temporary credentials with ease and benefit from the enhanced security they provide.

Create alias for your account



AWS Account

Account ID

022499045079

Account Alias

Create

Sign-in URL for IAM users in this account

https://022499045079.signin.aws.amazon.com/console

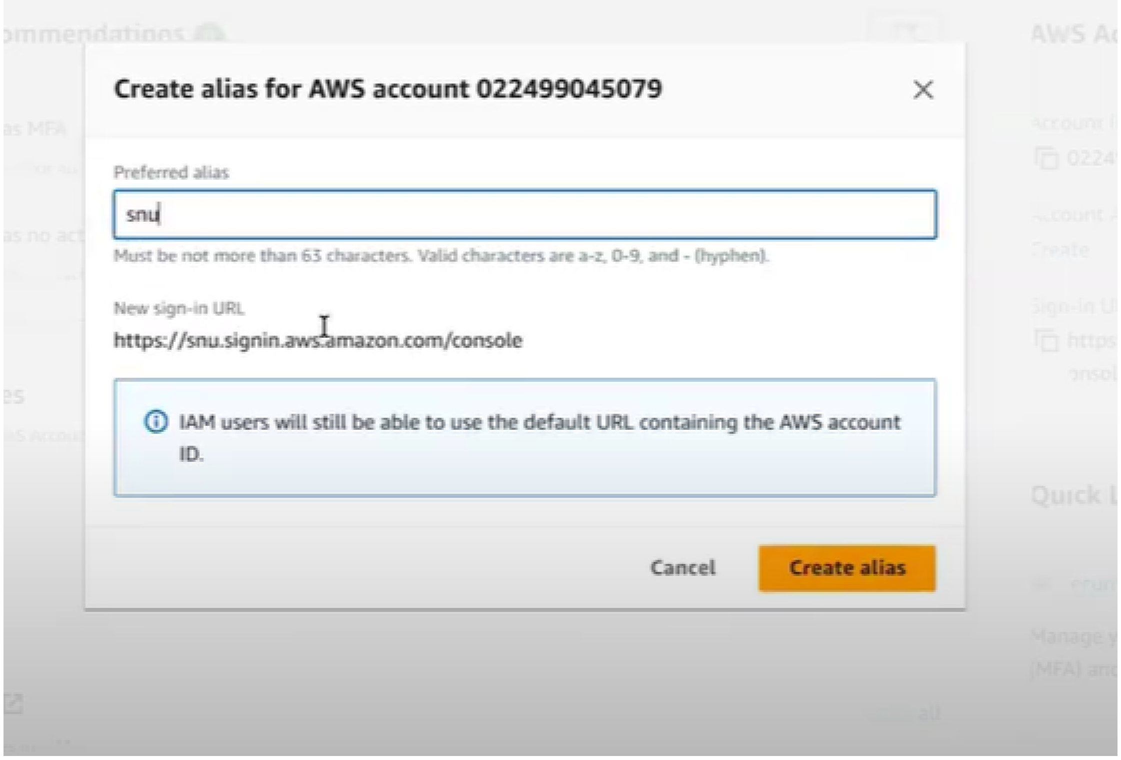
Quick Links

My security credentials

Manage your access keys, multi-factor authentication (MFA) and other credentials.

Sign out of bob

Sign into alice



Create alias for AWS account 022499045079

Preferred alias

snu

Must be not more than 63 characters. Valid characters are a-z, 0-9, and - (hyphen).

New sign-in URL

https://snu.signin.aws.amazon.com/console

IAM users will still be able to use the default URL containing the AWS account ID.

Instructions:

IAM :

Here's a structured guide to create IAM users and configure AWS access and services based on your requirements. I'll assume you have AWS administrative access to start this process:

Step-by-Step Guide

Step 1: Set Up IAM Groups and Users

1. **Log in to AWS Management Console**:
   - Go to the IAM Console.
2. **Create the Server Admin Group (Full EC2 Access)**:
   - In the IAM Console, select **Groups** > **Create Group**.
   - Name it **ServerAdmin**.
   - Attach the **AmazonEC2FullAccess** policy.
   - Click **Create Group**.
3. **Create Users Alice and Bob and Add to ServerAdmin Group**:
   - Go to **Users** > **Add Users**.
   - Add **Alice** and **Bob** as separate users.
   - Select **AWS Management Console Access** and configure their passwords.
   - Under **Permissions**, add **Alice** and **Bob** to the **ServerAdmin** group.
   - Click **Next**, review, and **Create Users**.
4. **Create the DNS Admin Group (Full Route 53 Access)**:
   - Go back to **Groups** > **Create Group**.
   - Name it **DNSAdmin**.
   - Attach the **AmazonRoute53FullAccess** policy.
   - Click **Create Group**.
5. **Create Users Cathy and David and Add to DNSAdmin Group**:
   - Go to **Users** > **Add Users**.
   - Add **Cathy** and **David** as separate users.
   - Select **AWS Management Console Access** and set their passwords.
   - Under **Permissions**, add **Cathy** and **David** to the **DNSAdmin** group.
   - Review and **Create Users**.
6. **Create the Billing User (Eve)**:
   - Go to **Users** > **Add User**.
   - Name the user **Eve**.
   - Select **AWS Management Console Access**.
   - Assign the **AWSBillingReadOnlyAccess** policy directly to **Eve**.
   - Complete and **Create User**.
7. **Create Your User with Full Access**:
   - Go to **Users** > **Add User**.
   - Name the user as **your_name** (replace with your actual name or preferred username).
   - Select **AWS Management Console Access**.
   - Attach the **AdministratorAccess** policy for full AWS access.
   - Review and **Create User**.

Step 2: Create Alias for Account Login URL

1. In **IAM Console**, go to **Account Settings** (you may need administrative permissions).

2. Under **Account Alias**, click **Create Account Alias**.
3. Set an alias (e.g., `companyname-alias`).
4. Save, and use the alias in your URL:
   - Format: `https://<alias>.signin.aws.amazon.com/console`
5. Use this URL to log in to verify that the alias works.

Step 3: Log In as Alice and Launch EC2 Instance

1. **Log in** to the AWS Console as **Alice**.
2. Go to the **EC2 Dashboard**.
3. Click **Launch Instance**:
   - Name: `Web Server`
   - Select an AMI, e.g., Amazon Linux 2.
   - Choose an instance type (e.g., t2.micro for free-tier eligibility).
   - Configure instance details as needed.
   - **Assign Elastic IP**:
     - Under **Network settings**, select **Assign a public IP**.
4. **Configure Security Group**:
   - In the Security Group, ensure ports such as **80 (HTTP)** and **22 (SSH)** are open to the appropriate IP ranges (e.g., `0.0.0.0/0` for testing, though it's more secure to restrict IPs in a production environment).
5. Review and **Launch** the instance.

Step 4: Verify Port 80 Accessibility

1. Once the instance is running, copy the **public IP** of the instance.
2. Visit Port Checker.
3. Enter the public IP and **Port 80** to check if it is open.

This setup will help ensure that the IAM and EC2 configurations work as specified. Let me know if you run into any specific issues during the setup!

Rest of codes

https://chatgpt.com/share/672901af-72b8-8013-8cf7-673d3607fbb1