Real-time Demo Workshop

CrateDB

Table of Contents

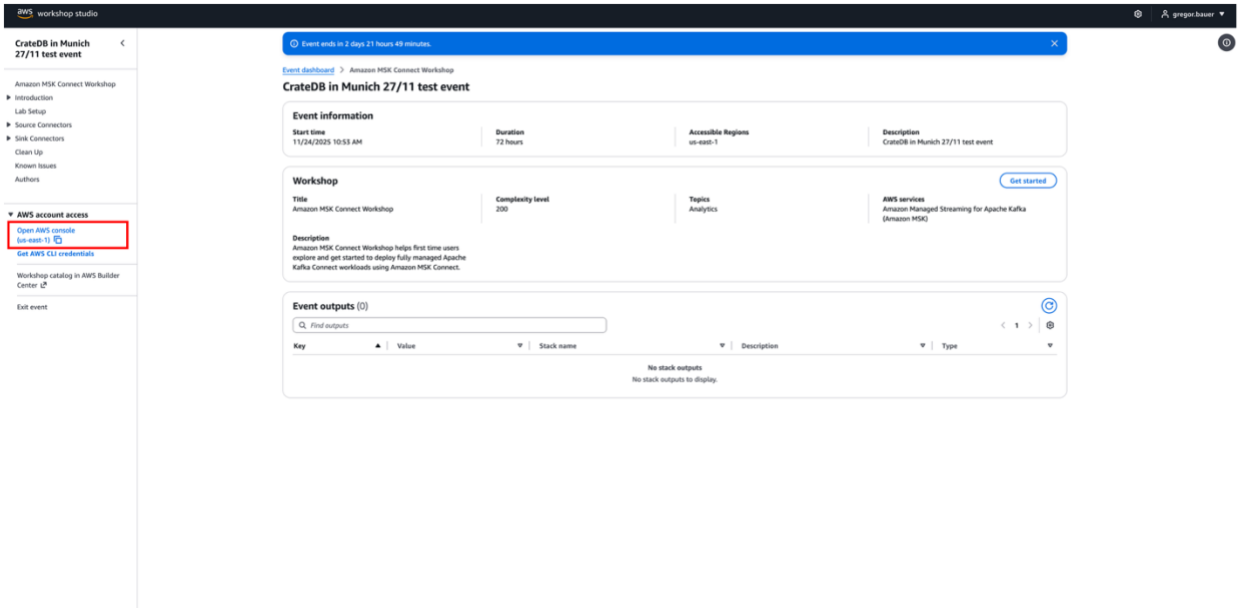
Workshop User Instructions

Creating AWS Workshop Pre-Requisites

Note: Please use the exact names provided in this guide for naming instances otherwise build steps might fail. Use copy paste to avoid errors.

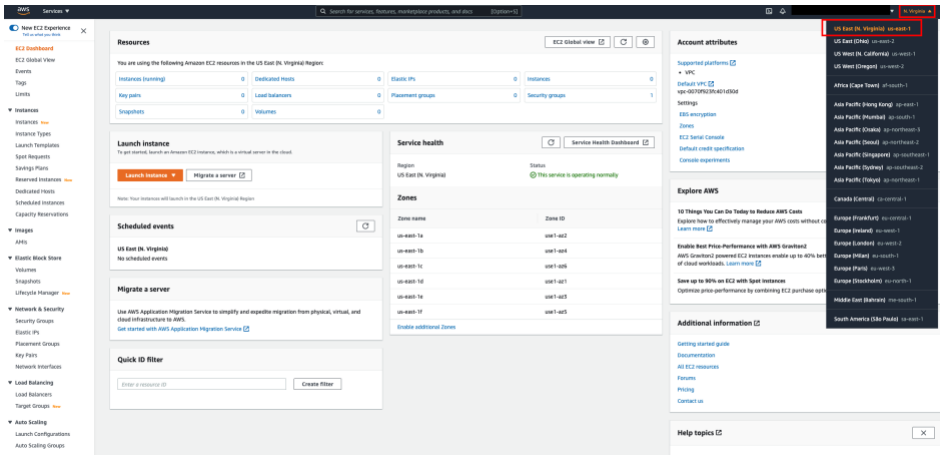
Note: If you already have AWS console it's best to use another browser

1. Click on the provided link and add the email you registered for the event to get an OTP
2. After entering the OTP you received go to the Amazon Console

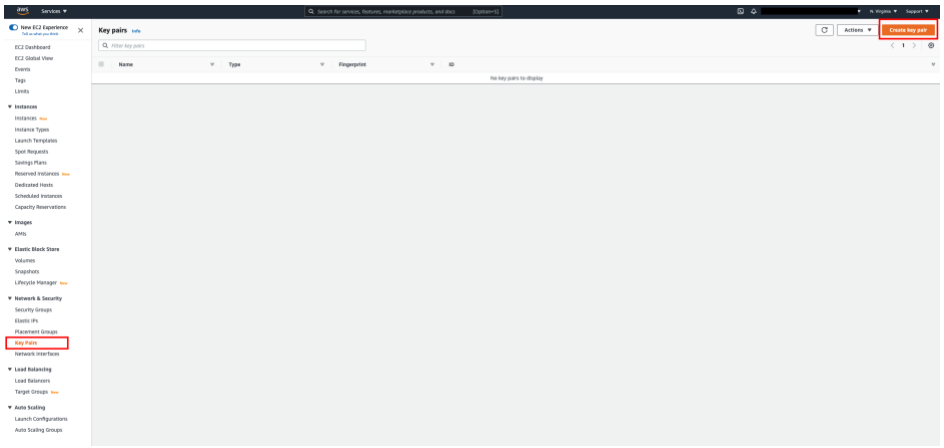


3. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/

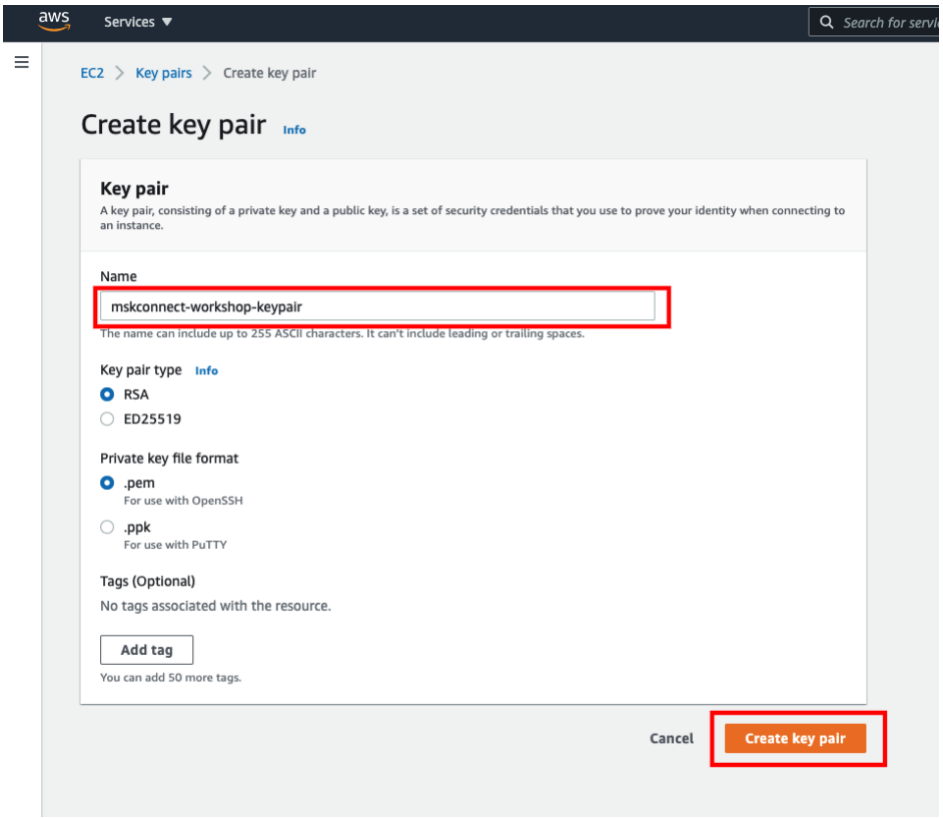
Choose region **us-east-1** this will be the region that you are using for the workshop.



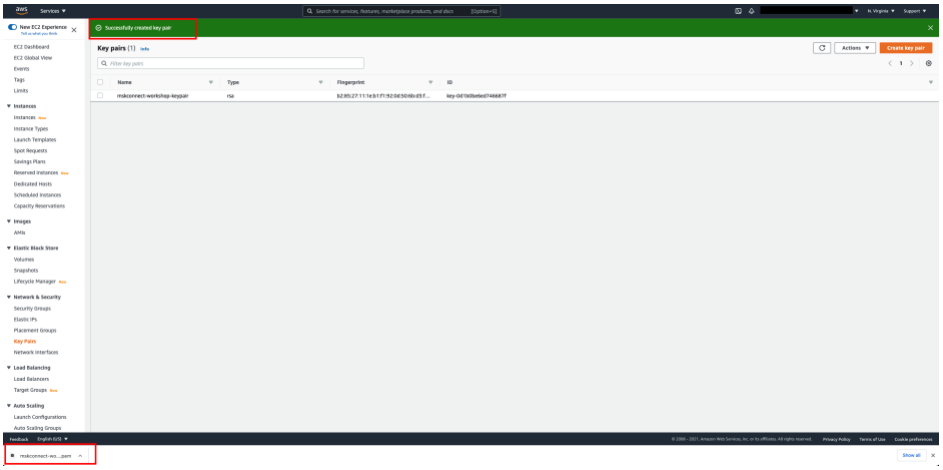
2. Choose **Key Pairs** in the navigation bar on the left and click on **Create key pair**.



3. Provide a name for the key pair (e.g. **mskconnect-workshop-keypair**) and click on **Create key pair**.

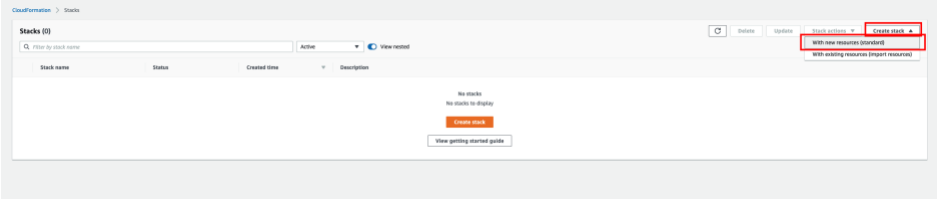


4. You should see the message **Successfully created key pair**, and confirm the download of the generated **.pem** file to your local machine.

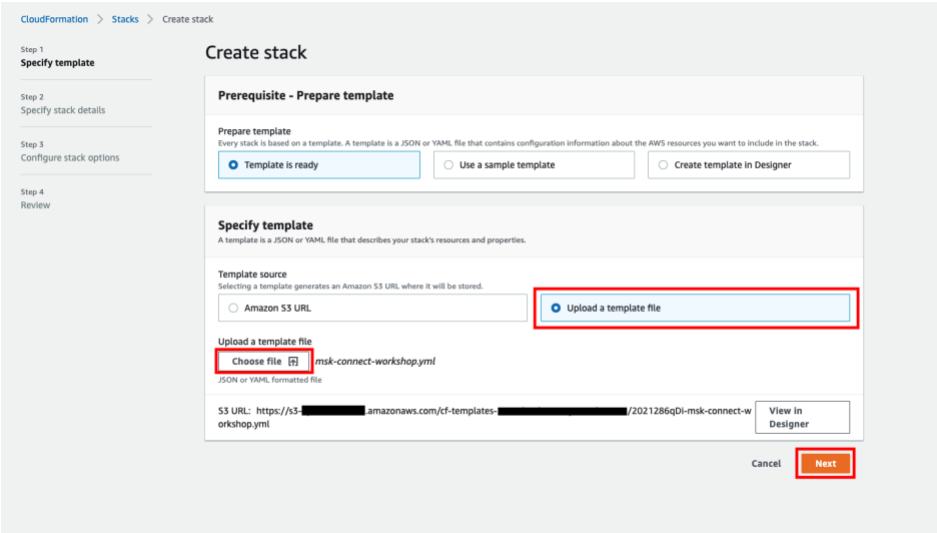


Deploy required AWS resources via CloudFormation

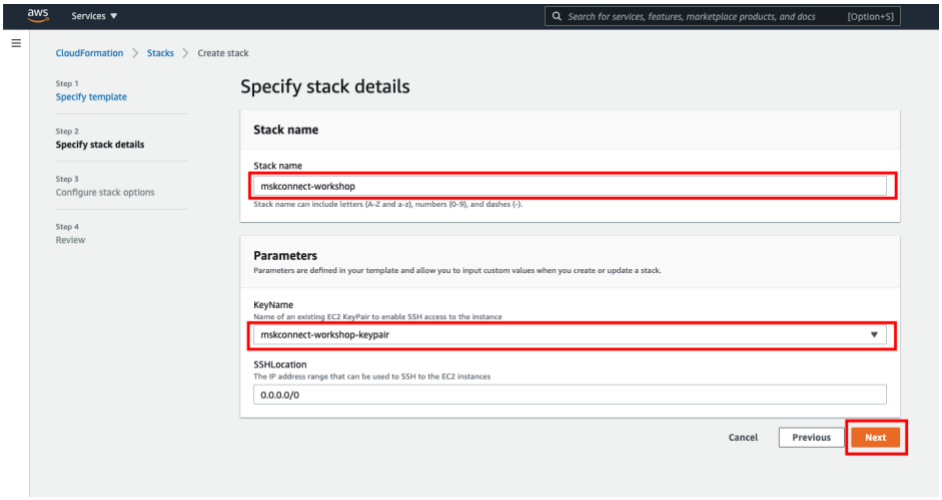
5. Download the CloudFormation template here: msk-connect-workshop-3.6.0.yml
6. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/
7. Click on **Create stack** and select **With new resources (standard)**.



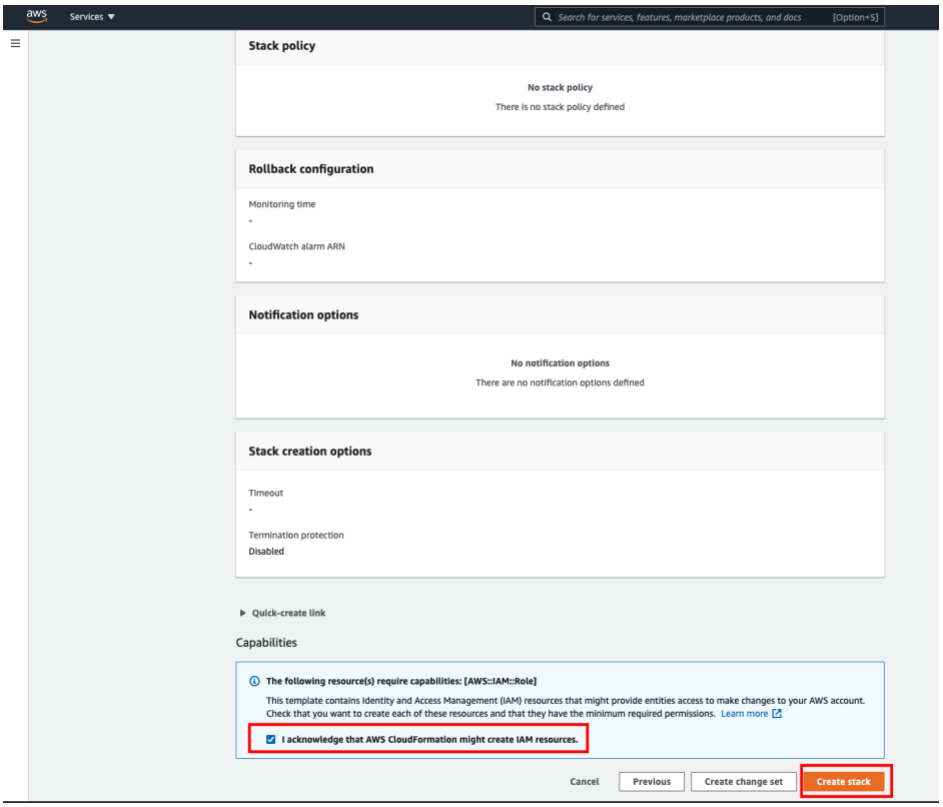
8. Choose **Upload a template file** option and upload the CloudFormation template from your local machine, then click **Next**.



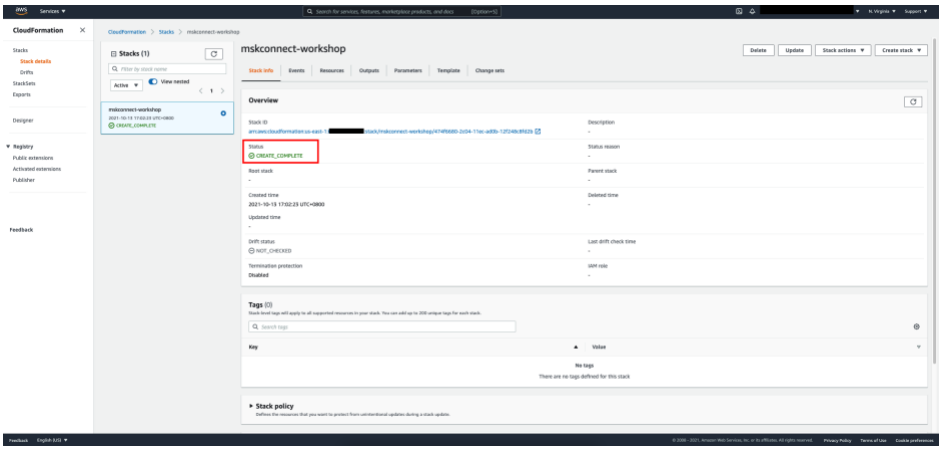
9. Enter **mskconnect-workshop** as the stack name, choose the **mskconnect-workshop-keypair** under **KeyName**, and then click **Next**.



10. Leave the default settings on the step **Configure stack options** and click **Next**.
11. Scroll down to the bottom of the **Review mskconnect-workshop** screen and check the box to acknowledge the creation of IAM resources. Click **Create stack** to trigger the creation of the CloudFormation stack.



12. Wait until the stack creation is completed before proceeding to the next steps. It may take a while for the MSK cluster creation (~30 to 45mins).



What does the above CloudFormation template do?

The CloudFormation stack creates the following resources:

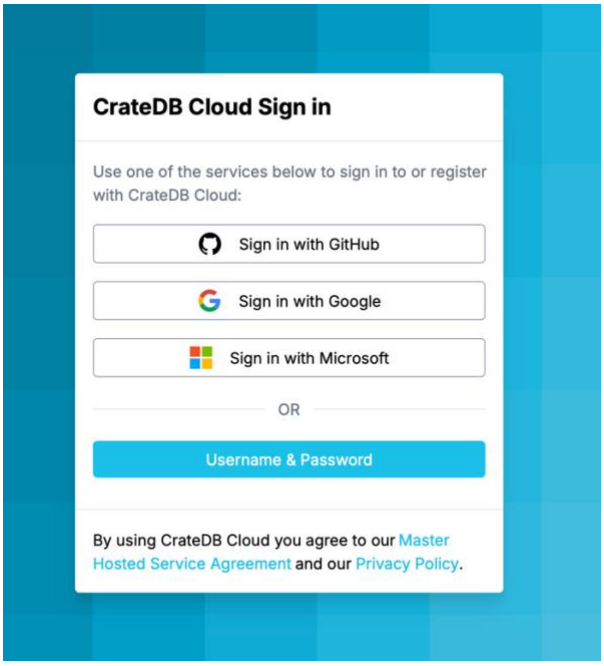
- 2 CloudWatch Log Groups for MSK Cluster and MSK Connect connectors respectively, each with 7 days log retention period
- 1 S3 bucket for storing required Apache Kafka Connect plugin files, and other testing results in some workshop modules
- 1 VPC with 1 Public subnet and 3 Private subnets, and the required networking components such as Route Tables, Internet Gateway, NAT Gateway & Elastic IP, VPC Gateway Endpoint for S3
- 1 m5.large EC2 instance with Java 1.8.0 and Apache Kafka version 3.6.0 installed, and the required Security Group, IAM Role, EC2 Instance Profile
  Note: we will not be using this and instead creating a new instance.
- 1 MSK Cluster with 3 kafka.m5.large broker nodes configured with Apache Kafka version 3.6.0, and the required Security Group

Note that the encryption options are disabled in order to reduce the chance of errors in the workshop, this is not a best practice, and you should not use the provided CloudFormation template in any production environments.

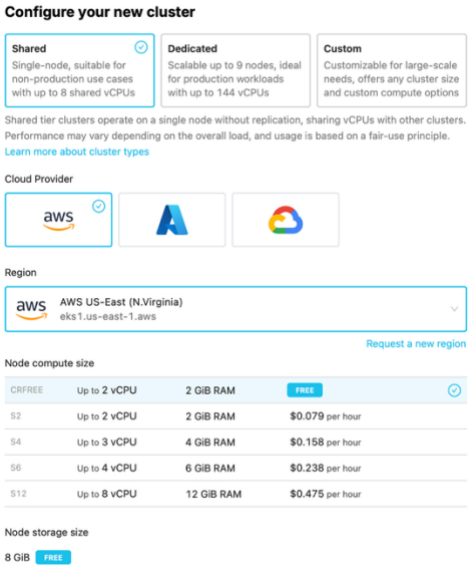
Note: While this is running, this would be a good time to create the CrateDB cluster in the next section.

Creating a CrateDB Cluster

1. Go to https://cratedb.com
2. Click on the **Start Free** button
3. You have several options to create an account, use the one that is appropriate for you. You won't need to enter any payment details.



4. Once you have an account, log in and deploy a **CRFREE** cluster



   - Select a **Shared** cluster (this is the only way to get a free cluster)
   - Select **AWS** as the cloud provider

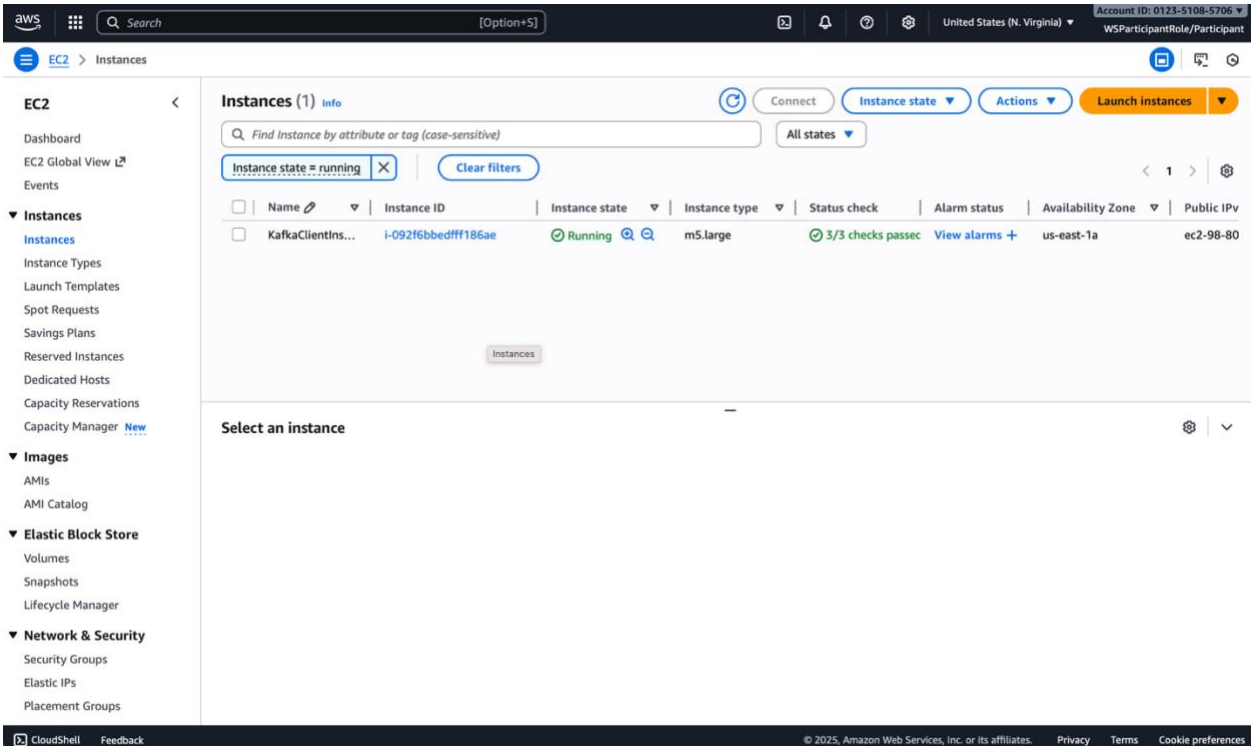
   - Select **AWS US-East** as the region (the same as the workshop runs within)
   - Ensure **CRFREE** is selected as compute size
   - Click **Deploy Cluster** to create it, which will take a short amount of time
5. It's **very** important that you save the login credentials, so download or make a note now before moving on.
6. We need to create the destination table for the climate data, run the following in the Console:

```
CREATE TABLE IF NOT EXISTS "demo"."climate_data" (
   "timestamp" TIMESTAMP WITHOUT TIME ZONE,
   "geo_location" GEO_POINT,
   "data" OBJECT(DYNAMIC) AS (
      "temperature" DOUBLE PRECISION,
      "u10" DOUBLE PRECISION,
      "v10" DOUBLE PRECISION,
      "pressure" DOUBLE PRECISION,
      "latitude" DOUBLE PRECISION,
      "longitude" DOUBLE PRECISION,
      "foo" BIGINT
   )
)
```

7. The host information you will need later, so make a note or leave the webpage open for now.

Creating AWS Workshop Environment

Open the EC2 dashboard https://console.aws.amazon.com/ec2/ go to **Instances**

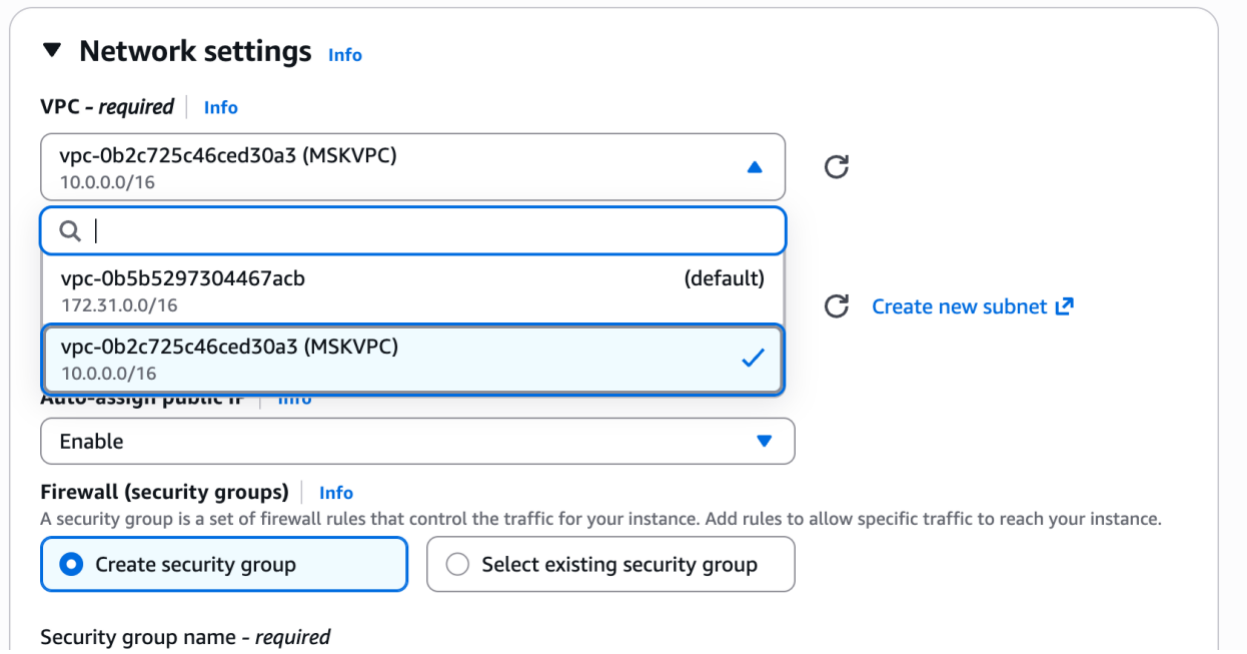


"

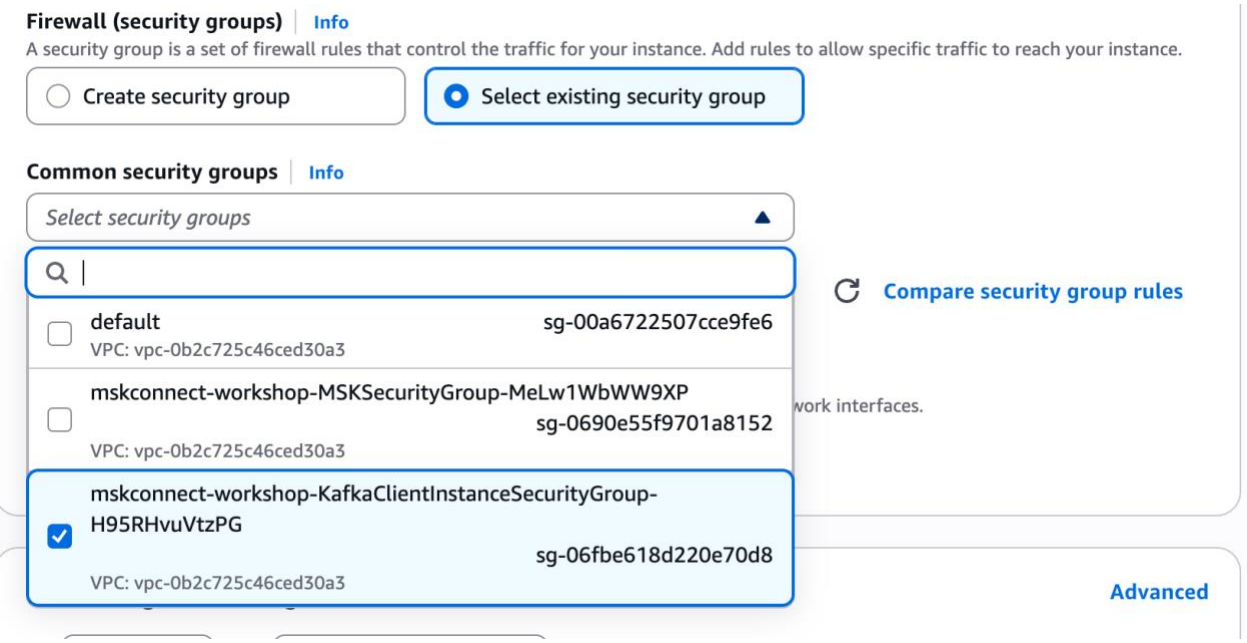
Note: We need to create a new EC2 instance, do not use the one created as part of the CloudFormation process as it's ARM based, and you will need an x86 instance.

Click the **Launch instances** button and choose the following:

- Name: **cratedb-workshop**
- AMI: **Amazon Linux**
- Architecture: **x86**
- Instance type: **m5a.xlarge**
- Key-pair name: **Choose the one previously created**
- Network Settings: Click on **Edit**
    - VPC: Find and select the workshop **MSKVPC** (same as existing EC2)



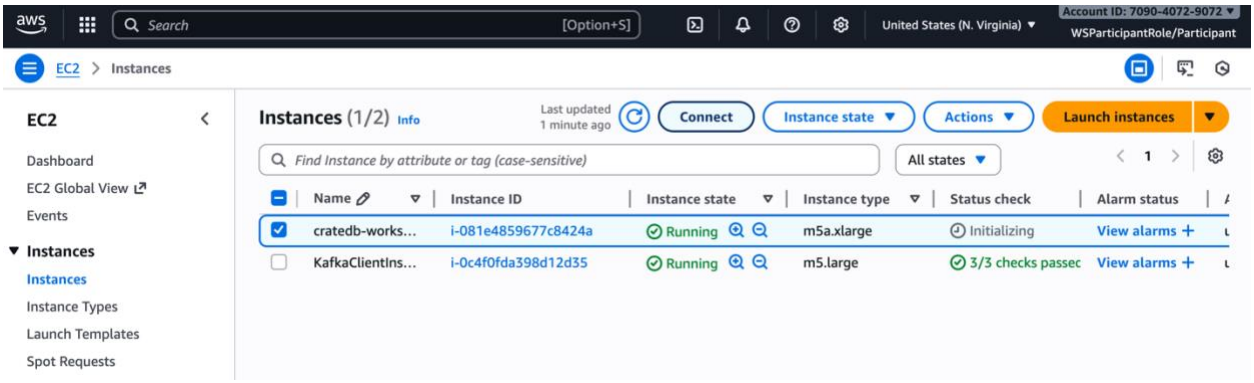
- Security group: Existing one starting `mskconnect-workshop-KafkaClientInstanceSecurityGroup` (same as existing EC2)

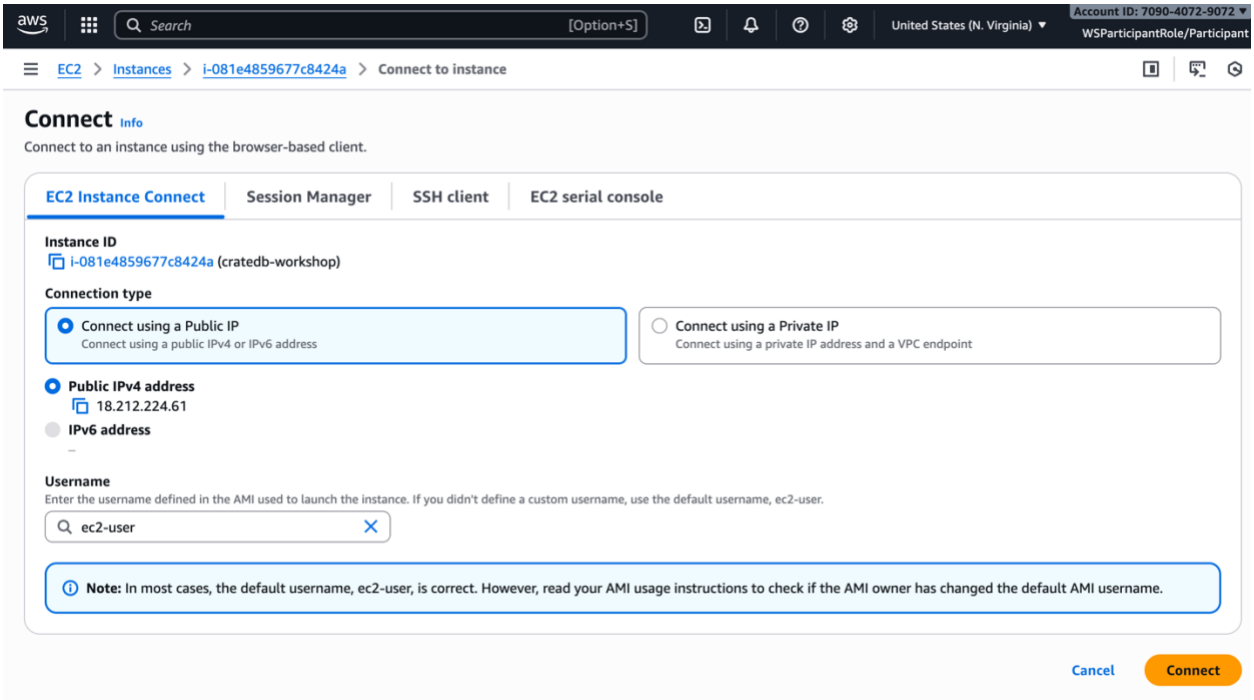


- Configure storage: **30GB**
- Ensure it's created in a **public** subnet, not private
- Ensure a public elastic IP is assigned

Create the instance and wait for it to be fully ready before attempting to connect

- Go to the EC2 instance and select the create one
- Click on the **Connect** button and then **Connect** again





Installing the demo source code

1. Ensure the git command-line tool is installed:

   ```
   sudo dnf install git
   ```

2. Clone the repository in the EC2 console (using HTTPS, not SSH). We use a particular "unauthenticated" branch (this removes the SASL authentication code, which is not needed).

   ```
   git clone https://github.com/crate/realtime-demo.git -b unauthenticated
   ```

3. Change into the directory:

   ```
   cd realtime-demo
   ```

Running the data producer

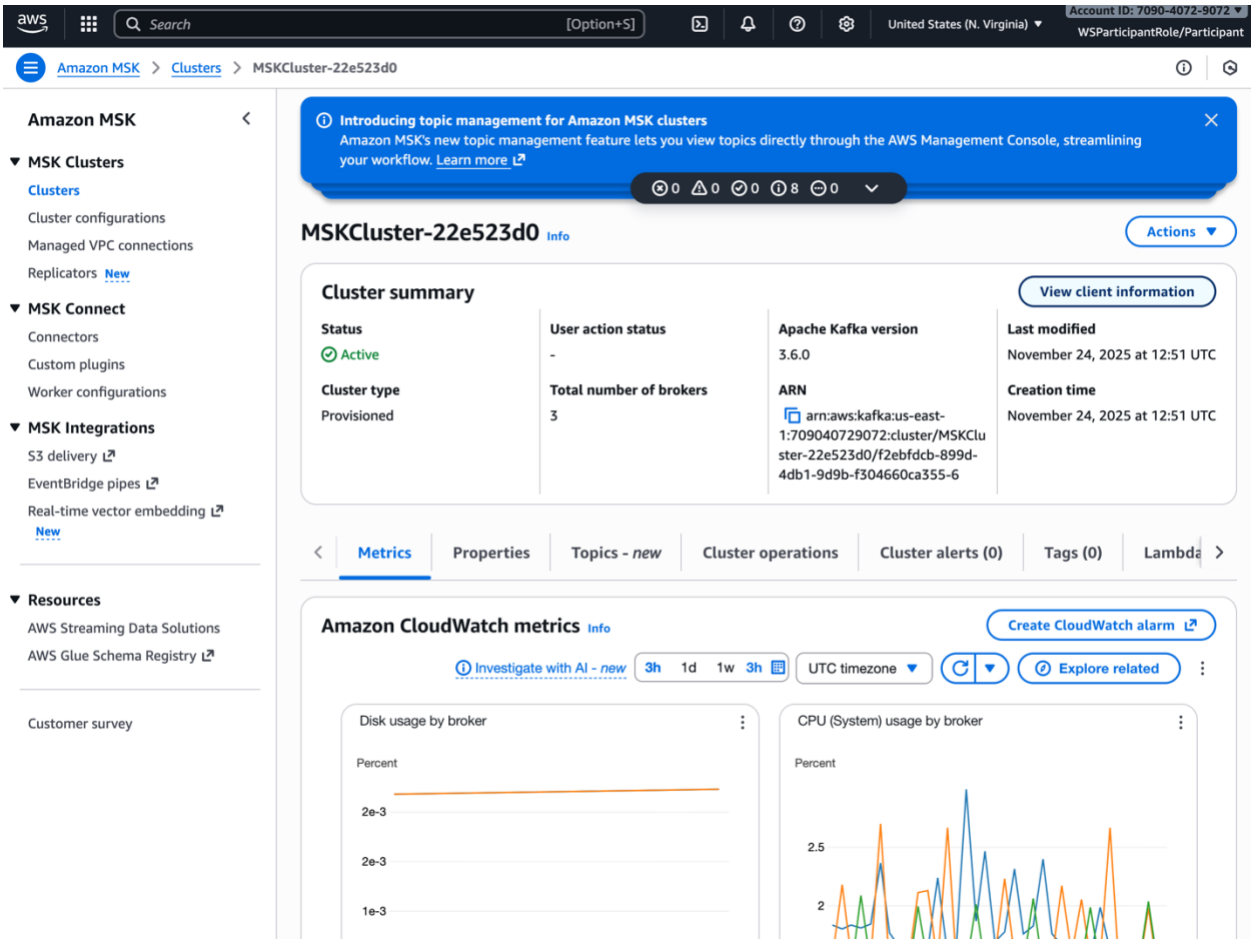
To run the producer, we need to set up a virtual Python environment and install dependencies:

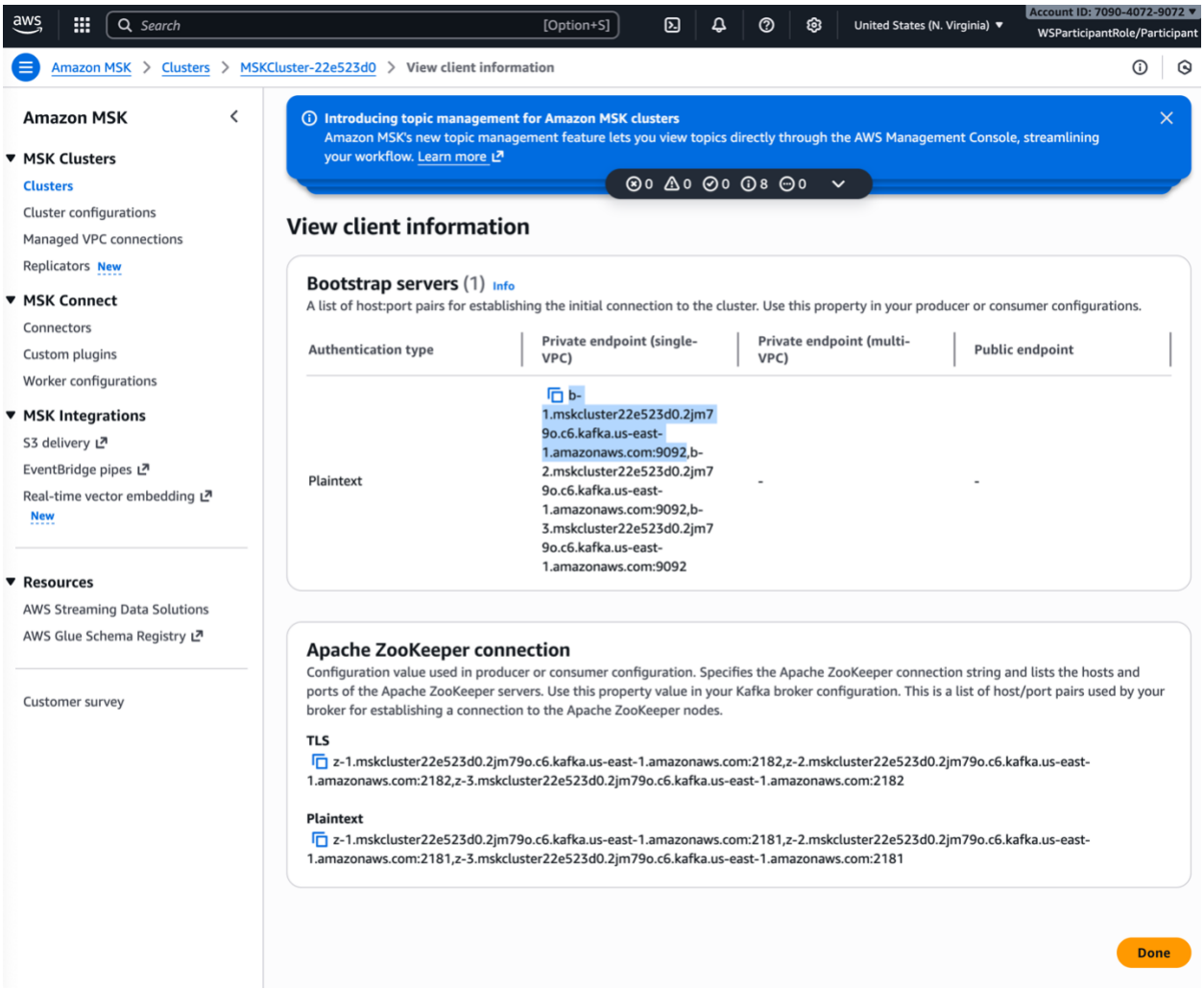
```
cd data
python3 -m venv .venv
source .venv/bin/activate
pip3 install -U -r requirements.txt
```

Next, copy the example .env file

```
cp .env.example .env
```

Afterwards change the bootstrap server. Go to **MSK**, click on the MSK Cluster, then click on "View client information". Then copy one of the Bootstrap servers. Note that they are comma separated. Only select one and make sure it's fully copied.


Search
[Option+S]
United States (N. Virginia)
WSParticipantRole/Participant
Amazon MSK
Clusters
MSKCluster-22e523d0
Amazon MSK
MSK Clusters
Clusters
Cluster configurations
Managed VPC connections
Replicators New
MSK Connect
Connectors
Custom plugins
Worker configurations
MSK Integrations
S3 delivery
EventBridge pipes
Real-time vector embedding
New
Resources
AWS Streaming Data Solutions
AWS Glue Schema Registry
Customer survey
Introducing topic management for Amazon MSK clusters
Amazon MSK's new topic management feature lets you view topics directly through the AWS Management Console, streamlining your workflow. Learn more
MSKCluster-22e523d0 Info
Actions
Cluster summary
View client information
Status
Active
User action status
-
Apache Kafka version
3.6.0
Last modified
November 24, 2025 at 12:51 UTC
Cluster type
Provisioned
Total number of brokers
3
ARN
arn:aws:kafka:us-east-1:709040729072:cluster/MSKCluster-22e523d0/f2ebfdcb-899d-4db1-9d9b-f304660ca355-6
Creation time
November 24, 2025 at 12:51 UTC
Metrics
Properties
Topics - new
Cluster operations
Cluster alerts (0)
Tags (0)
Amazon CloudWatch metrics Info
Create CloudWatch alarm
Investigate with AI - new
3h
1d
1w
3h
UTC timezone
Explore related
Disk usage by broker
Percent
2e-3
2e-3
1e-3
CPU (System) usage by broker
Percent
2.5
2



Next modify the env file in the console with the bootstrap server you copied

```
vi .env
```

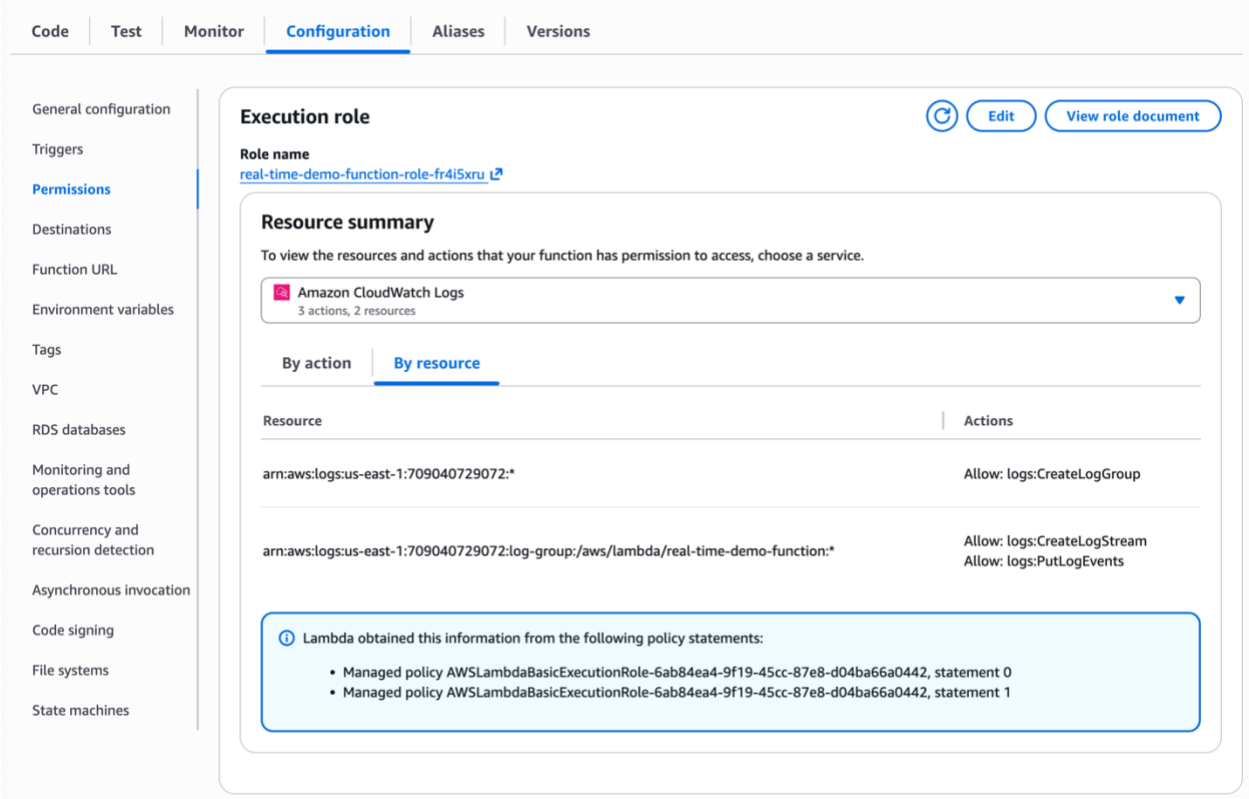
To run the producer, simply execute it.

```
python3 producer.py
```

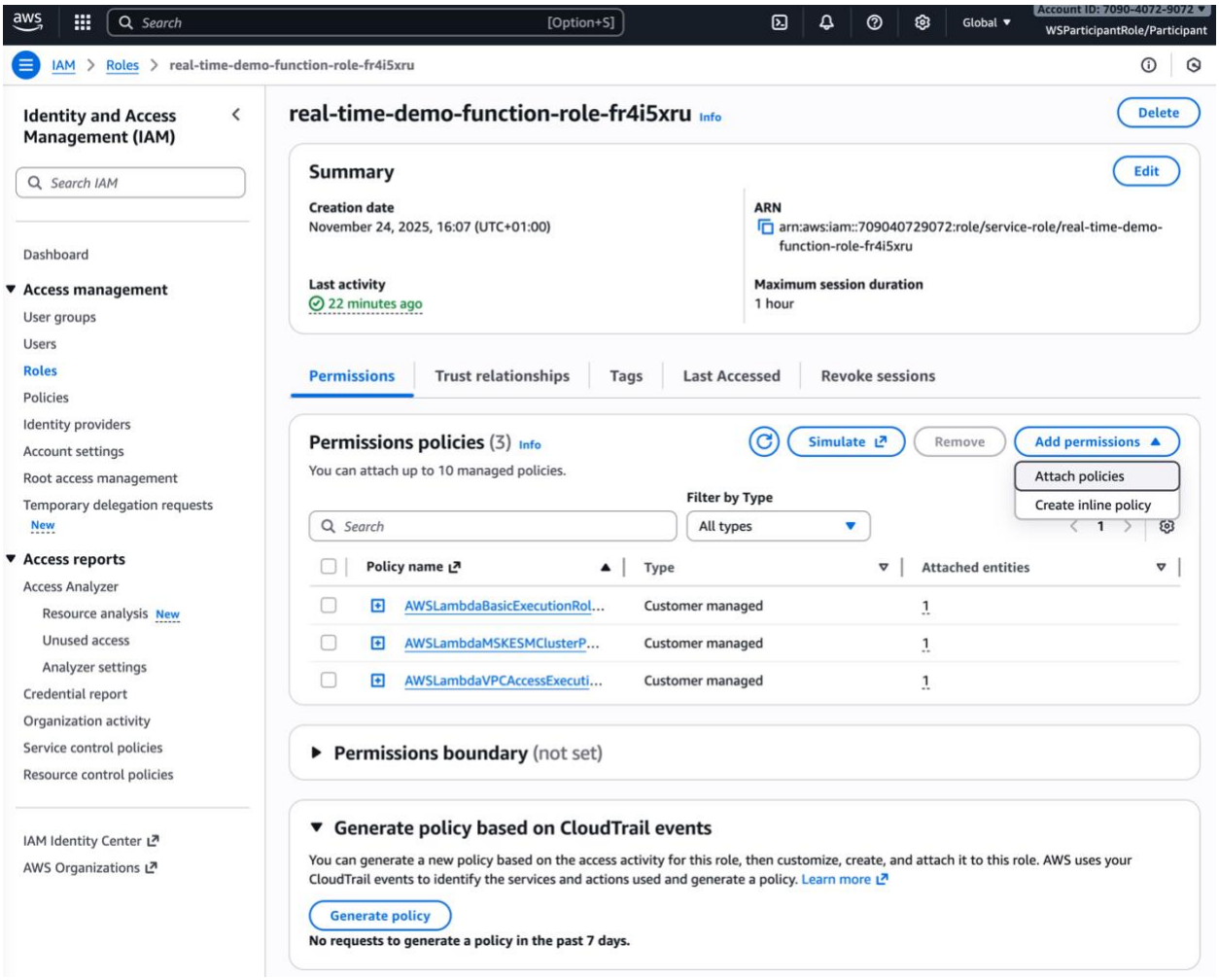
AWS Lambda function for Data Consumer

- Create a Lambda function using the **Author from scratch** option
- Give it the name `real-time-demo-function`
- Choose the Python 3.13 runtime (**not** 3.14)
- Either architecture is fine
- Under **Additional Configurations**:

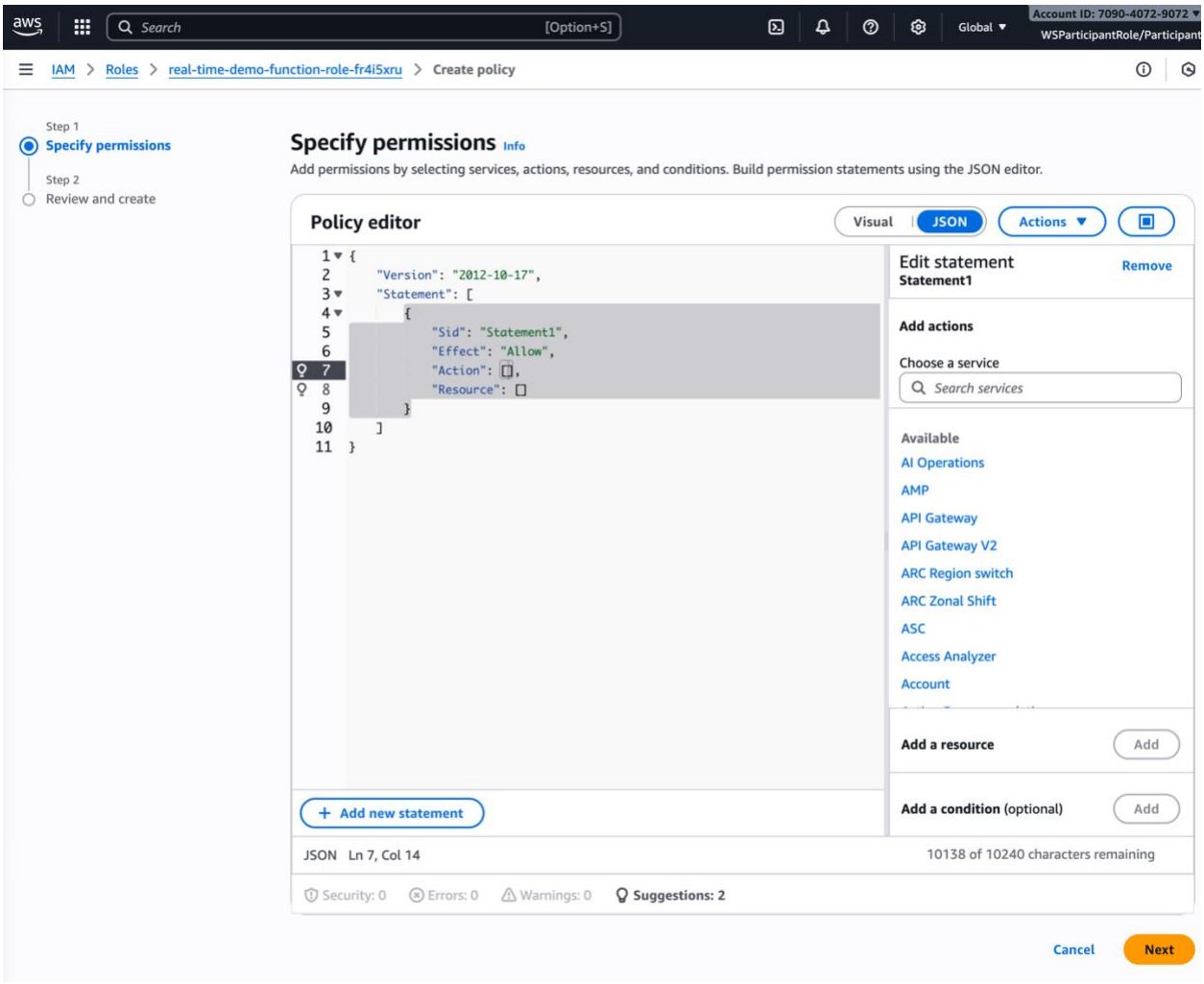
  - Choose the VPC created earlier (MSKVPC)
  - Select the **private** subnets (not public)
  - Select the previously created security group starting with **mskconnect-workshop-MSKSecurityGroup**
- Create the Lambda!
- We need to add some variables
- Go to **Configuration**
- Under **Environment Variables** enter following:
  - CRATEDB_HOST (URL excluding prefix and suffixes, just the hostname)
  - CRATEDB_DB (`crate`)
  - CRATEDB_PASS (password you got when your cratedb cluster was created)
  - CRATEDB_USER (`admin`)
  - CRATEDB_PORT (`4200`)
  - SOURCE_TOPIC (`dev-1-0`) (Note the -0 suffix)
- Some additional security permissions are needed to allow our Lambda to access other resources like the MSK cluster.
  - Go to: **Configuration**, then P**ermissions**, then click on the **Role name** at the top to open the Lambda execution role.



- Add the following permissions to the Lambda execution role:
  - `AdministratorAccess` (never do this in a production environment!)



- After that add a new inline policy to allow the Lambda execution role to access MSK (this rule allows access to all MSK clusters)

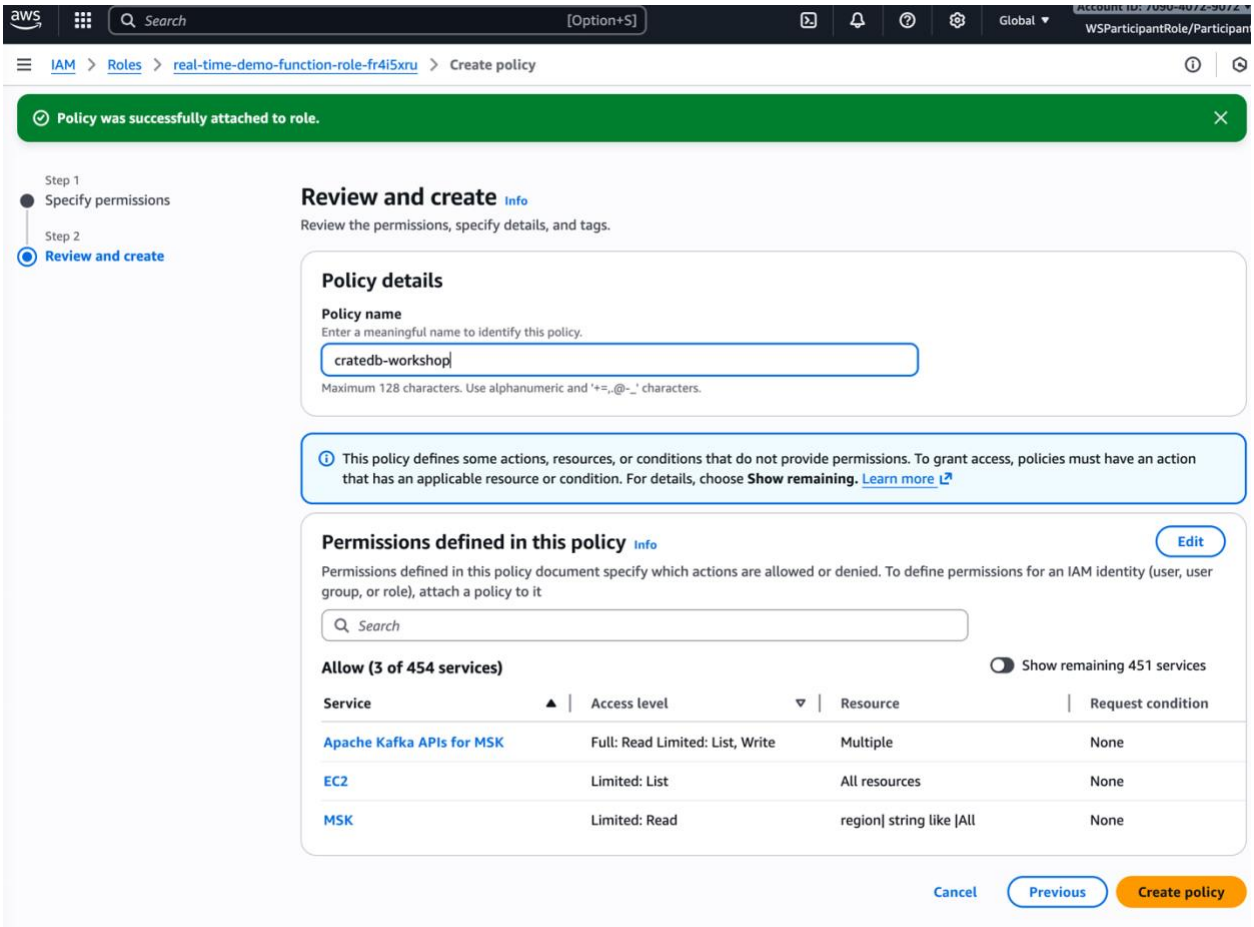


- Select JSON and paste the code below

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kafka:DescribeCluster",
        "kafka:DescribeClusterV2",
        "kafka:GetBootstrapBrokers"
      ],
      "Resource": "arn:aws:kafka:*:*:cluster/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kafka-cluster:Connect",
        "kafka-cluster:AlterCluster",
        "kafka-cluster:DescribeCluster",
        "kafka-cluster:DescribeTopic",
        "kafka-cluster:ReadData",
        "kafka-cluster:AlterGroup",
        "kafka-cluster:DescribeGroup"
```

```
      ],
      "Resource": [
        "arn:aws:kafka:*:*:cluster/*",
        "arn:aws:kafka:*:*:topic/*",
        "arn:aws:kafka:*:*:group/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs"
      ],
      "Resource": "*"
    }
  ]
}
```

- Click on Next and name this **cratedb-workshop**
- Create the policy



Now we upload the Lambda code, this has been provided in the form of a ZIP file release to make it simple to update.

- The link is: https://github.com/crate/realtime-demo/releases/tag/Resources
- Once downloaded go to the **Lambda** page and select the **Code** section
- Go to **Upload from** button which allows the ZIP file to be uploaded.

Create a trigger for the AWS Lambda function

Next step is to create a trigger from MSK when a new value in our demo topic is created. To do this:

- Go to created Lambda Function
- Go to **Configuration**
- Go to **Triggers**
- Select **Add Trigger**
- Select **MSK** as the source
- Find the cluster created earlier (likely named real-time-demo)
- The topic name is **`dev-1`**
- **Provisioned mode** should be **disabled**
- Use **IAM** authentication
- Set **Starting position** to **At timestamp**
- Starting position is timestamp **`2025-10-14`**
- Under additional settings, batch size determines how quickly data is ingested, set to **`100`**
- Click on **Add**

Installing Grafana onto AWS EC2 instance

Update packages

```
sudo dnf update -y
```

Add the Grafana repository

```
sudo tee /etc/yum.repos.d/grafana.repo <<EOF
[grafana]
name=Grafana OSS
baseurl=https://packages.grafana.com/oss/rpm
repo_gpgcheck=1
enabled=1
gpgcheck=1
gpgkey=https://packages.grafana.com/gpg.key
EOF
```

Install Grafana

```
sudo dnf install grafana -y
```

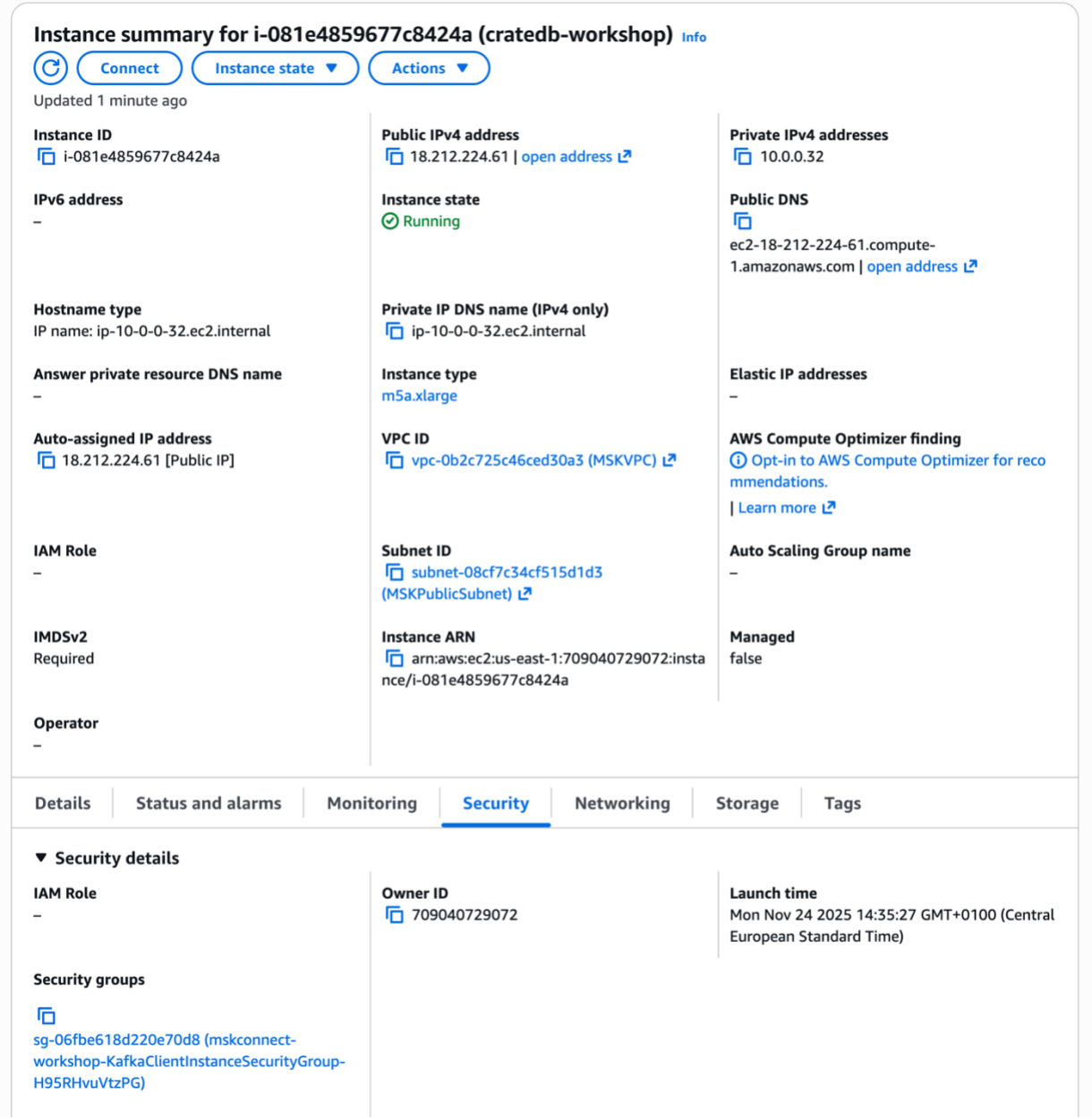
Enable and start the service

```
sudo systemctl enable grafana-server
sudo systemctl start grafana-server
```

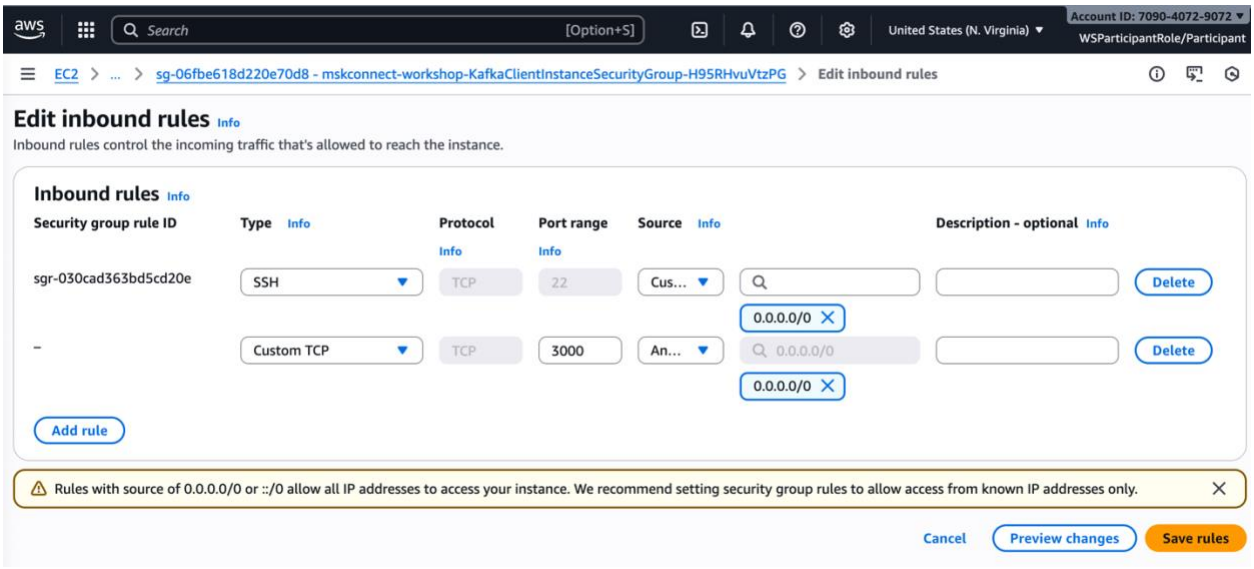
Open Grafana port

Edit the security group for the EC2 instance and open port 3000 for the Grafana dashboards.

- Go to EC2
- Select the running workshop instance



- Click on the security groups
- Click on **Edit inboud rules**
- Click on **Add rule**
- Select:
    - Type: Customer TCP
    - Port range: 3000
    - Source: Anywhere-IPv4



- Click on **Save rules**
- Go back to EC2
- Select the running workshop instance
- Make note of the public IPv4 address of the EC2 instance

Import dashboards

Your dashboard should be accessible at http://ip:3000 (where IP is the assigned external IP address of the EC2 instance).

To login, the default Grafana username and passwords are admin/admin, then either skip or set a new password.

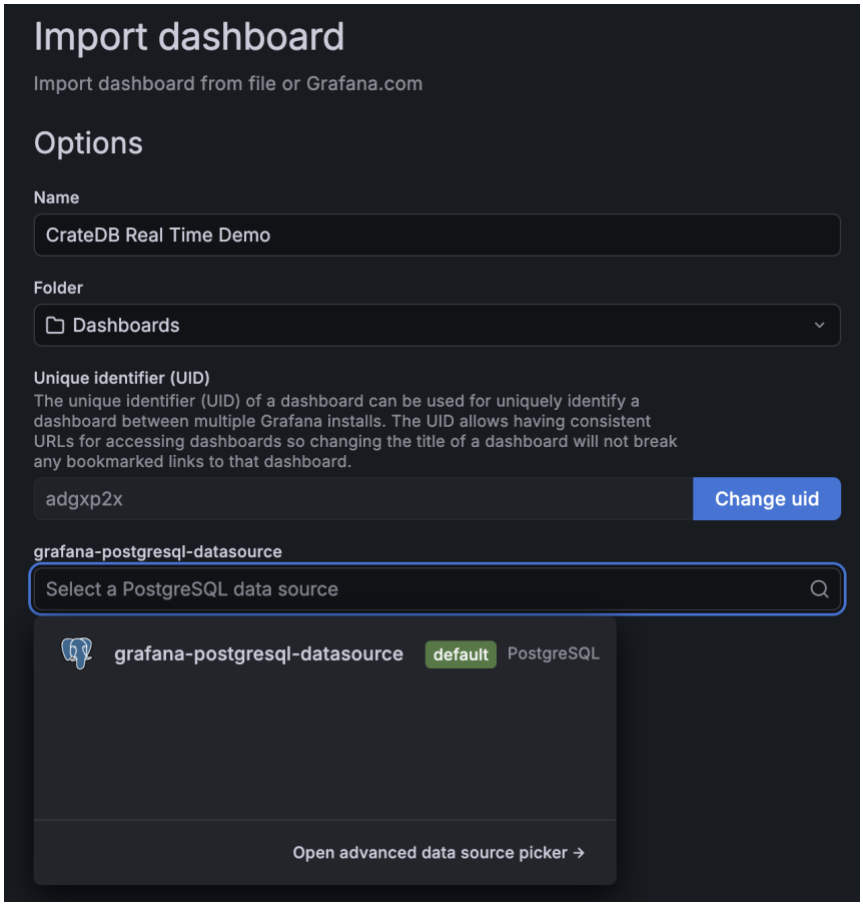
First, we will create a connection to the CrateDB cluster by configuring a corresponding connection.

- Go to **Connections** on the menu on the left
- Click on **Add new connection**
- Search for **PostgreSQL** and add those details from the beginning
    - Name: `grafana-postgresql-datasource`
    - Host URL: (the URL of you CrateDB cloud cluster without any protocol or port)
    - Database name: `crate`

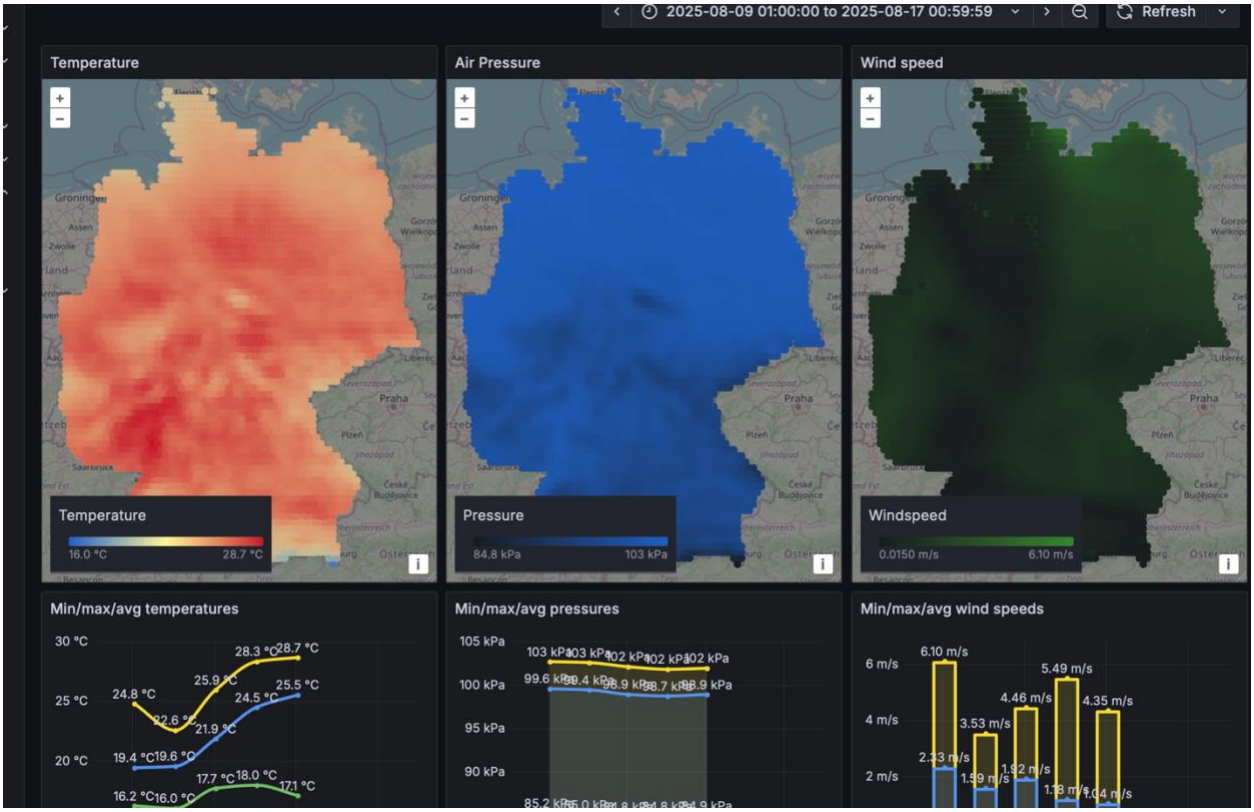
    - Username: `admin`
    - Password: (password from the CrateDB instance)
- Save and test

To set up the dashboard, download both JSON files from https://github.com/crate/realtime-demo/releases/tag/Dashboards-v1.1 and use the import functionality in Grafana.

- Go to **Dashboards**
- Click on **New** and select **Import**
- Select one of the downloaded two JSON documents
- Click on **Import**
- Select the previously created `grafana-postgresql-datasource`



- Upload the second one as well



All complete!