Assignment 3 - DevOpamis

⦁ Create a free tier account on AWS, Add alerts on billing

⦁ Setup 2FA and Create IAM user, policy etc.

⦁ Deploy the Django application on docker based EC2 (Deploy using Ansible)

⦁ Deploy front end application on S3 which would be accessible only through CDN.

⦁ Jenkins job to deploy BE and FE using Jenkins pipeline jobs (Jenkins should be hosted locally)

⦁ Manage the infrastructure(EC2, vpc, nat, igw etc) using terraform.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Steps to complete the tutorial:

**Objective 1**: Create a free tier account on AWS, Add alerts on billing (skipped notes for now)

**Objective 2**: Setup 2FA and Create IAM user, policy etc. [CAN BE TAUGHT AS A MINI-PROJECT WITH IAM]

**Prerequisites**: Knowledge of AWS IAM, Install “Google Authenticator” on your mobile device

* Create a user:

Sample user details (for my personal reference)

User 1: Jasbir

Password: jasbir1!

Link: https://856088786762.signin.aws.amazon.com/console

Account ID Number: 856088786762

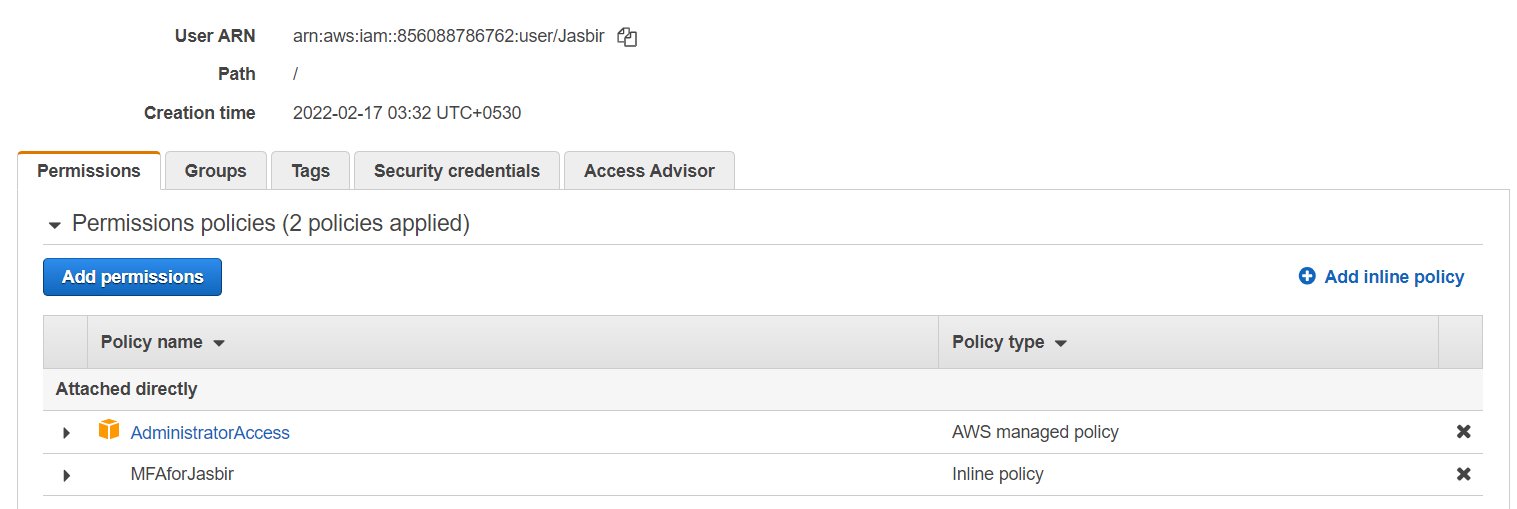
(To find your account ID number, log in via your IAM account, click on the user's name on the top-right of the screen)

Make sure that your user has “AdministratorAccess” permission granted. Also, you need special permissions for MFA set-up, which can be found in the link below:

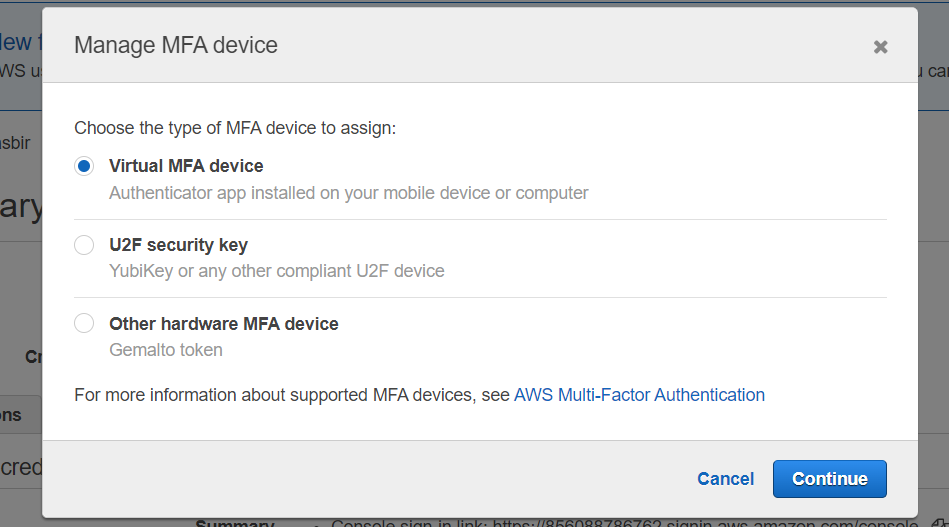
<https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_enable_physical.html>

There are multiple ways to set-up MFA for an IAM user, one of those is hardware-based.

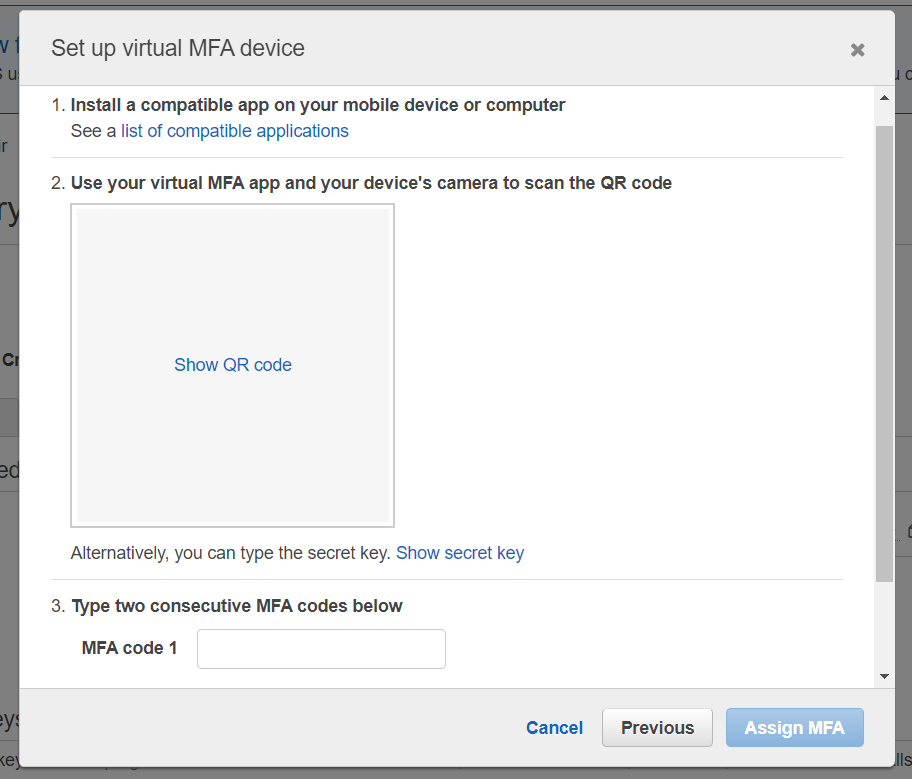
Go to your user’s details in the IAM section of AWS, click on “Add inline policy”, then click on “JSON” tab, and paste the code that you see in the link, and apply the policy, which will enable your user to set-up MFA and disable it.



Next, click on Security credentials tab, and select “Virtual MFA Device”



Next, Download “Google Authenticator” on your mobile device, and scan the QR code on your screen

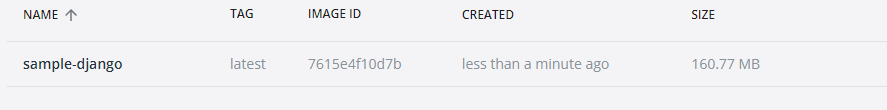


To sync your devices, write two consecutive codes (as seen on your mobile device) in the MFA code 1 and 2 sections

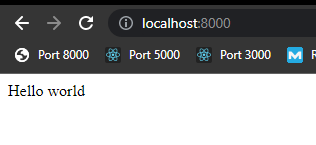
Now, MFA will be required during sign-on for this user, for which you can see your OTP on the Authenticator app

**Objective 3**: Deploy the Django application on docker based EC2 (Deploy using Ansible)

Prerequisites: [Create a sample django application](https://www.youtube.com/watch?v=ZsJRXS_vrw0&ab_channel=IDGTECHtalk)

* Next, Dockerize the Django application
  + Go to the root directory of the application
  + Run “pip freeze > requirements.txt”
    - This will create a new file “requirements.txt”, and write all the dependencies required to run the Django application
    - Create “Dockerfile” in the root directory. This will contain the list of commands to be run by the application
    - Now, we need to choose an [OS that our container will use, which supports python](https://hub.docker.com/_/python) (we have chosen 3.10-slim-buster for this experiment)
    - Now we need to specify the application folder
      * WORKDIR /app
    - Now we have created and moved into the “app” folder
    - Copy the dependencies from requirements.txt to a new file called requirements.txt
      * COPY requirements.txt requirements.txt
    - Install the dependencies
      * RUN pip3 install -r requirements.txt
    - Copy all the files used in our Django project
      * COPY . .
      * Basically, we are asking Docker to copy everything from this directory (first ".") to the app directory (second ".")
    - Write the command used to run the Django project, and we want to make it accessible from outside the container
      * CMD ["python3", "manage.py", "runserver", “0.0.0.0:8000”]
    - Ignore the virtual environment folder in a “.dockerignore” file
    - Build the image using the command:
      * docker build -t sample-django .
      * Open “Images” tab in Docker Desktop. You should see the following:
      * 
      * docker run -p 8000:8000 sample-django
        + This is to create the container
        + -p flag is used to map the port from the container to the machine on which the application is running
        + Verify container formation by going to the “Containers/Apps” section on Docker desktop



* Go to Port 8000. Your application should be running
* 
* Steps to push this image to a repository to [Docker Hub](https://docs.docker.com/docker-hub/repos/)

**Objective 4**: Deploy front-end application on S3 which would be accessible only through CDN.

Steps:

<https://github.com/arindamkeswani/DevOps/tree/main/S3xCDN>

(IGNORE: OAI: arindam-s3xcdn.s3.us-east-1.amazonaws.com)

**Objective 5**: Jenkins job to deploy BE and FE using Jenkins pipeline jobs (Jenkins should be hosted locally)

Notes are present [here](https://docs.google.com/document/d/1iAMfC-jk8EjW9Mh-iZdPACgHPwvqtR_5HteNzIvvQ68/edit?pli=1#bookmark=id.utuja7f2k6kx)

**Objective 6**: Manage the infrastructure(EC2, vpc, nat, igw etc) using terraform.

Notes are present [here](https://github.com/arindamkeswani/DevOps/tree/main/terraform/project1)