Deploy Services on AWS (Deprecated)

Contact info

Tan | ts864@cornell.edu

Overview

Currently, the way services are deployed is basically running their Docker containers on AWS EC2 instances. Within each container, a Flask service is hosted using Gunicorn instead of native Flask to make the service more fault-tolerant.

Test and production environments for our mobile and dashboard services are deployed separately on the EC2 instances below.

Environment	IP address	Domain	Port	Sample URL	Key pair file
Mobile (Online) **Test**	54.163.4.203	**`mobile-test.diaper-project.com`** (deprecated: on-test.diaper.cf)	5001	https://mobile-test.diaper-project.com:5001/api/monitoring	DIAPER confidential > AWS EC2 key files > DIAPER-test-key Box folder link
Dashboard (Offline) **Test**	54.243.201.107	**`dashboard-test.diaper-project.com`**	5000	https://dashboard-test.diaper-project.com:5000/env/	
Mobile (Online) **Production**	35.168.248.57	**`mobile-prod.diaper-project.com`** (deprecated: on-prod.diaper.cf)	5001	https://mobile-prod.diaper-project.com:5001/api/monitoring	DIAPER confidential > AWS EC2 key files > DIAPER-production-key Box folder link
Dashboard (Offline) **Production**	3.228.124.129	**`dashboard-prod.diaper-project.com`**	5000	https://dashboard-prod.diaper-project.com:5000/env/	

Procedure

The procedure for deploying is the same for all environments. First, download the key pair file corresponding to the instance and run

```
chmod 400 /path/to/DIAPER-*-key.pem
```

Then, ssh into the corresponding EC2 instance using

```
ssh -i /path/to/DIAPER-*-key.pem ec2-user@<domain>
```

See How to ssh into AWS / BioHPC if you're having trouble

Pull your Docker image and other relevant file from GitHub. Instead of your own git account and password, the username is diapertestemail@gmail.com , and the password is the token in the login secrets.

Once pulled, navigate to the folder containing docker-compose files and run the corresponding command as explained below:

```
// For test environments.
./deploy.sh -n "Your Name" -m "Reason for deployment" test

// For production environments.
./deploy.sh -n "Your Name" -m "Reason for deployment" prod

// For local development on your laptop.
// These two commands are equivalents (i.e. default is docker-compose.yml)
sudo docker-compose -f docker-compose.yml up -d --force-recreate
sudo docker-compose up -d --force-recreate
```

For test and production environments, make sure to fill your name and reason for deployment. All test and production deployments will be logged in deploy mentHistory.log under the same directory.

The commands above run your container in detached mode so that your service doesn't block the console. However, once a container is run in detached mode, all of its runtime errors will not be reported to the console. Thus, as the last step of deployment, you need to manually check that no runtime errors occurred during the launching of your service by looking at its log using

```
sudo docker-compose logs -f
```

If there are no runtime errors, you will get an output similar to the one below.

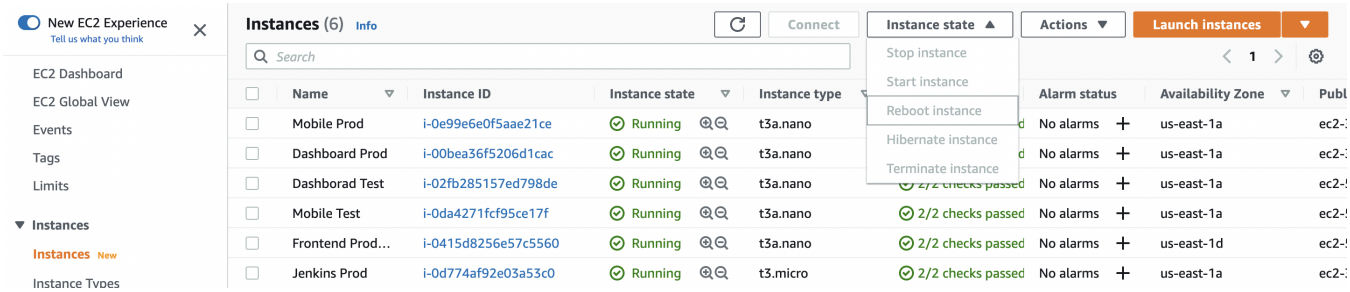
```
Attaching to dev-dashboard_dashboard-backend_1
dashboard-backend_1  |  * Environment: production
dashboard-backend_1  |    WARNING: Do not use the development server in a production environment.
dashboard-backend_1  |    Use a production WSGI server instead.
dashboard-backend_1  |  * Debug mode: off
dashboard-backend_1  |  * Running on https://0.0.0.0:5000/ (Press CTRL+C to quit)
```

Now you can log out and the service will continue running on the EC2 instance.

Troubleshooting

Server is down

1. Reboot the instance(shown below)



2. If the problem persists, connect to the relevant server and run the deployment script:

```
// For test environments.
./deploy.sh -n "Your Name" -m "Reason for deployment" test

// For production environments.
./deploy.sh -n "Your Name" -m "Reason for deployment" prod
```

The particular issue may be highlighted during the execution of the above command.

~~BioHPC database timeout~~ (Obsolete)

~~If you are experiencing timeout when connecting to the BioHPC database, it' probably because the EC2 instance isn't connected to Cornell's VPN. Check whether the VPN is connected with~~

```
ps -A | grep openconnect
```

~~If the output is non-empty, then the VPN is connected. If not, you can connect to the VPN using~~

```
sudo openconnect -b cuvpn.cuvpn.cornell.edu --reconnect-timeout 600
```

~~and enter necessary information as prompted.~~