[version_1.1]

Note

The exercises in this course will have an associated charge in your AWS account. In this exercise, you create the following resources:

- AWS Identity and Access Management (IAM) policy and user (Policies and users are AWS account features, offered at no additional charge)
- AWS Cloud9 integrated development environment (IDE) instance
- AWS CloudFormation stack
- Amazon DynamoDB table

Familiarize yourself with **AWS Cloud9 pricing**, **AWS CloudFormation pricing**, **Amazon DynamoDB pricing**,and the **AWS Free Tier**.

Setting up

This exercise requires an IAM role that will be used with the AWS Cloud9 instance. It also requires a DynamoDB table that will be set up and then used in a later exercise. The CloudFormation stack will create these resources for you. *You will also need to choose a Region where AWS App Runner is **available**.*

1. Download the following CloudFormation template: exercise-containers.yml. This template will set up the backend resources that are needed to complete the exercise.

   Note: If you have an existing Virtual Private Cloud (VPC) and it has a Classless Inter-Domain Routing (CIDR) block of *10.16.0.0/16*, you must edit the template to change it.

2. Sign in to the AWS Management Console as a user that has administrator permissions.

3. Choose **Services**, and search for and open **CloudFormation**.

4. Choose **Create stack**.

5. For **Specify template**, choose **Upload a template file**.

6. Select **Choose file** and browse to where you downloaded the `exercise-containers` template.

7. Select the `exercise-containers` template and choose **Open**.

8. Choose **Next**.

9. For **Stack name**, enter `exercise-containers`.

10. Choose **Next**, and then choose **Next** again.

11. Select the acknowledgement and choose **Create stack**. *Wait for the stack to complete.*

Exercise 1: Creating the First Container

In this exercise, you create an AWS Cloud9 instance and modify some of the environment settings. You also install the prerequisites and source code that are needed to launch new container instances inside your AWS Cloud9 environment.

Task 1: Creating an AWS Cloud9 instance

In this task, you will create an AWS Cloud9 instance.

1. In the AWS Management Console, choose **Services**, and then search for and open **Cloud9**.

2. Choose **Create environment**.

3. For **Name**, enter `containers-cloud9` and choose **Next step**.

4. Configure the following settings.
    - **Instance type**: *t3.small (2 GiB RAM + 2 vCPU)*
    - **Network settings: VPC settings**: *Containers VPC*

5. Choose **Create**.

Task 2: Modifying and deploying source code to AWS Cloud9

In this task, you will upgrade the AWS Command Line Interface (AWS CLI) **version**, and deploy the source code that's needed to complete the exercise. You will also associate the instance profile with your AWS Cloud9 instance. Finally, you will expand your environment to give it more disk space.

1. In the AWS Cloud9 terminal, upgrade the AWS CLI version by running the following command.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip

unzip awscliv2.zip

sudo ./aws/install

. ~/.bashrc
```

2. Verify the version.

```
aws --version
```

3. Download and extract the source code that you need for this exercise.

```
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/DEV-AWS-MO-Containers
```

```
unzip containers-src.zip
```

4. Replace the current association with the **cloud9-containers-role**.

```
export ASSOC_ID="`aws ec2 describe-iam-instance-profile-associations | grep Assoc

aws ec2 replace-iam-instance-profile-association --iam-instance-profile Name=clou
```

5. Make sure that the **State** is listed as *associated*.

```
aws ec2 describe-iam-instance-profile-associations
```

6. Run the following command.

   This command tells AWS Cloud9 to specifically disable the managed rotated credentials, and instead use the `cloud9-containers-role` that CloudFormation created from the template.

```
aws cloud9 update-environment  --environment-id $C9_PID --managed-credentials-act
```

7. Wait for a couple of minutes, and then run the following command:

```
aws sts get-caller-identity
```

   You should see an Amazon Resource Name (ARN) that matches a role in the CloudFormation template:

```
arn:aws:sts::0123456789012:assumed-role/cloud9-containers-role/i-xxxxxxxxxxxxxxx
```

8. Expand the AWS Cloud9 disk by running the following utility:

```
bash utilities/c9-resize.sh 40
```

Task 3: Building the first container

In this task, you will pull the required images and build your first Docker container.

1. Change directory to `first-container/`.

```
cd first-container/
```

2. Inspect the Dockerfile, and build a container image.

```
docker build -t first-container .
```

3. View the Docker images.

```
docker image ls
```

   The application being served sits on port 8080 in the container image.

4. Publish port 8080 to your AWS Cloud9 host and make it available on port 8080.

```
docker run -d -p 8080:8080 --name webapp first-container
```

5. View the running Docker containers.

```
docker ps
```

6. View the logs that are being captured from the container.

```
docker logs webapp
```

   You will now view the application in a browser.

7. At the top of your AWS Cloud9 instance, choose **Preview** and then choose **Preview Running Application**.

   You should see the running application.

8. Launch a shell inside the container.

   You can use this shell to run commands within the container itself.

```
docker exec -it webapp /bin/sh
```

9. In the container, look at the contents of the **/app** folder and view the contents of **/app/input.txt**.

```
ls /app

cat /app/input.txt
```

10. In the container, view the process list.

```
ps -a
```

11. Escape out of the container (you can escape out of the container by pressing Ctrl+D).

12. Stop and remove the running container.

```
docker stop webapp

docker rm webapp
```

Task 4: Modifying the container with new data

In this task, you will change the `input.txt` file inside the container with new data. You can use a bind mount to mount a local file resource in place of `/app/input.txt` inside the container.

1. Create an `~/input.txt` file with five words, with each word on a new line.

```
printf "cinco\ncuatro\ntres\ndos\nuno" > ~/input.txt
```

2. Launch a container with the new file mounted in place of `/app/input.txt` .

```
docker run -d -p 8080:8080 \
-e MESSAGE_COLOR=#0000ff \
-v ~/input.txt:/app/input.txt \
--name webapp \
first-container
```

   You can configure an application that's running in a container with environment variables. The containerized application takes `MESSAGE_COLOR` as an environment variable.

3. Visit the updated application in a browser by choosing **Preview**, **Preview Running Application.**

4. Force-remove the container.

```
docker rm -f webapp
```

© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. Corrections, feedback, or other questions? Contact us at https://support.aws.amazon.com/#/contacts/aws-training. All trademarks are the property of their owners.