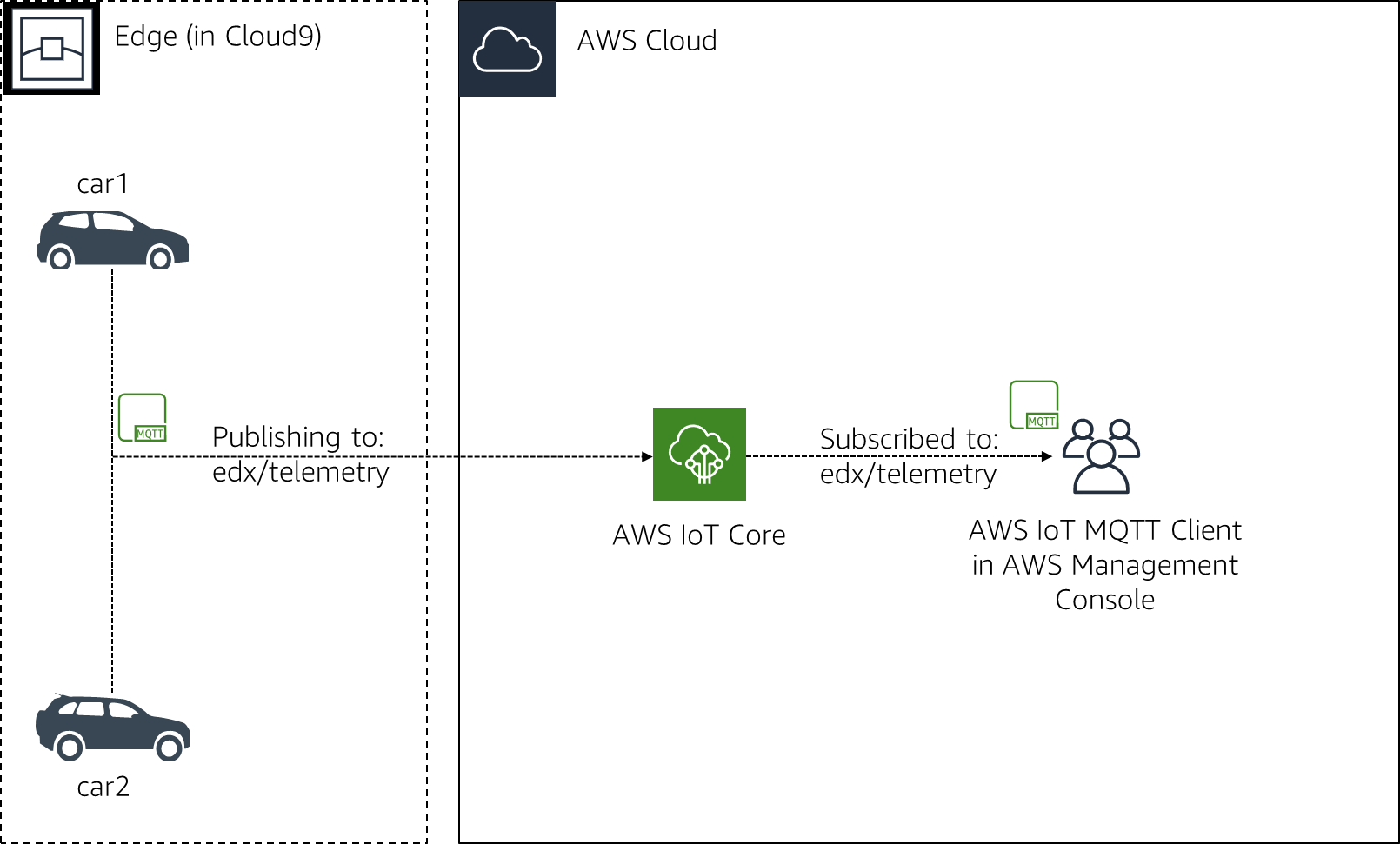
**Exercise 1 - Connect 2 ‘Drone’ Things to AWS IoT Core from AWS Cloud9**

In this exercise, you will create two ‘Drone’ IoT Things and connect them to AWS IoT Core service so they can send telemetry data to an IoT Topic so we can see in real-time the MQ Telemetry Transport (MQTT). To connect the simulated Drones, you will create (i) IoT Thing (ii) Security Certificate and (iii) AWS IoT Core Policy.

The Thing will represent a Drone. The Certificate will be used to authenticate to AWS IoT Core and the Policy will define what your Drone can do once authenticated. The first Drone will be created via the AWS Management Console the second drone created via AWS Command Line Interface (CLI) so you are able to build confidence creating AWS IoT Things via either method.

The Drones will be simulated in an AWS Cloud9 environment that you will need to create. My feedback working with newcomers to IoT environments, is that sometimes having a physical IoT dev board can add too much pressure for quick results, not to mention the obvious cost to create multiple IoT things with a real life IoT board compares to a virtual IoT Thing Board. You will download the code of the Drones, upload their IoT Certificates and start them up to simulate a real life drone. You will use the AWS IoT MQTT Client in the AWS Management Console to subscribe to the telemetry IoT Topic to confirm that the Drones are sending data and that you can receive the data in your Console.

The diagram below shows the resources and data flow that you will create in this exercise.



**1. Create an IAM Policy**

To begin, you will create an IAM User and Policy specific to this class so you can delete that user once you are done with the exercises.

In this section, you will create an IAM customer-managed policy. Customer-managed policies provide more precise control over your policies than policies managed by AWS. This policy will have permissions specific to the AWS resources you need for this course.

1. In the AWS Management Console, click **Services**, and then click **IAM** to open the IAM dashboard.
2. In the left navigation menu, click **Policies**.
3. Click **Create policy**.
4. Click the **JSON** tab.
5. In the editor text box, replace the sample policy with the following:

{

"Version": "2020-01-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"greengrass:\*",

"iot:\*",

"iotanalytics:\*",

"cloud9:\*",

"lambda:\*",

"s3:\*",

"sns:\*",

"iam:\*",

"cognito-identity:\*",

"cognito-sync:\*",

"cognito-idp:\*",

"logs:\*",

"ec2:\*",

"cloudwatch:\*",

"kms:ListAliases",

"kms:DescribeKey",

"cloudformation:DescribeStackResources",

"tag:getResources"

],

"Resource": "\*"

}

]

}

1. Click **Review Policy**.
2. For **Name**, enter DroneIoTPolicy.
3. Click **Create policy**.

You successfully created an IAM policy.

**2. Create an IAM user and attach a policy to the user.**

In this section, you will create an IAM user and attach a policy to the user. If you are familiar with IAM users, you might want to attempt to complete this section before reading the step-by-step instructions.

| **Option** | **Value** |
| --- | --- |
| IAM User Name | DroneIoTUser |
| Access Type | AWS Console access |
| Policy | DroneIoTPolicy |

Expand for step-by-step instructions.

1. In the AWS Management Console, click **Services**, and then click **IAM** to go to the IAM dashboard.
2. In the left navigation menu, click **Users**.
3. Click **Add user**.
4. In the **User name** text box, enter DroneIoTUser.
5. For **Access type**, select **AWS Console access**.
6. For **Console password**, choose **Custom password**. **Note the password**.
7. Remove the **check mark** next to **User must create a new password at next sign-in**.
8. Click **Next: Permissions**.
9. In the **Set permissions** section, click **Attach existing policies directly**.
10. In the search text box for **Filter**, enter DroneIoTUser.
11. Put a check mark next to **DroneIoTPolicy** in the filtered list.
12. Click **Next: Tags**.
13. Click **Next: Review**.
14. Review the information, and click **Create user**. You should see a success message.
15. Note the **sign-in URL** in the success message at the top. This is a special URL for IAM users, which includes your account ID.
16. Click on the **sign-in URL** in the success message at the top. This will log you out.
17. **Sign in** as the DroneIoTUser IAM user.

**3. Create an AWS Cloud9 environment**

In this section, you will create an AWS Cloud9 environment.

1. In the AWS Management Console, click **Services**, and then click **Cloud9** to open the Cloud9 dashboard.
2. Select an appropriate region with the following services available: Amazon Cognito, Amazon EC2, Amazon S3, Amazon SNS, AWS Cloud9, AWS IAM, AWS IoT Analytics, AWS IoT Core, AWS IoT Greengrass, AWS Lambda. **Ireland** is a suitable region for EMEA and what will be used for this workshop.
3. Click **Create environment** at the top-right corner.
4. For **Name**, enter IoTOnAWS.
5. Click **Next step**.
6. On the Configure Settings page, leave the default settings, and click **Next step**.
7. Review the details, and click **Create environment**. This should launch your AWS Cloud9 environment within a few minutes.

Note that this Cloud9 instance will automatically shutdown after 30 minutes if it's not used. All of your work will be saved and brought back to what it was if you were to re-open it.

**4. Setup your Cloud9 Environment, download the Drone code and the AWS IoT CA Public Cert**

In this section, you will install the Node package for the Drone code to work, download the Drone code and setup your repository structure for the Drones.

1. Install the AWS IoT Device SDK Node package by running the following command in your AWS Cloud9 **terminal**. You can find that terminal at the bottom of the page. There is a *bash* tab with the prompt *DroneIoTUser:~/environment: $*. You can adjust the size of that screen like you would in a normal IDE: put your cursor above the tab and select & drag to increase/decrease the space.

npm install aws-iot-device-sdk

You can ignore the warnings that there are no package.json files.

1. Create the repository structure for the Drone application. As there will be 2 Drones, you will create 2 folders. Run the following commands in your AWS Cloud9 **terminal**.

mkdir ~/environment/drone1**;** mkdir ~/environment/drone2

1. Download and copy the application code in each Drone folder by running the following commands in your AWS Cloud9 **terminal**:

cd ~/environment

wget https://github.com/morphman82/iotdroneworkshop/blob/master/droneconfig.js

cp droneconfig.js drone1/

cp droneconfig.js drone2/

rm droneconfig.js

1. Download the AWS IoT Certificate Authority Public Certificate that will be used in the code later and that will sign the IoT Certificates you will create in the next section. To do so, execute the following commands in your Cloud9 **terminal**:
2. cd ~/environment

wget -O root-CA.crt https://www.amazontrust.com/repository/AmazonRootCA1.pem

**5. Create Drone 1 IoT Thing, Certificate and Policy**

In this section you will use the AWS Management Console to create all of the resources required for your Drone 1 to connect to AWS IoT. This includes the Drone Thing, the Certificate and the Policy. We could use the wizard to create many of these resources, but instead, you will do it manually to see how each of the components is attached to each other. You will finish by uploading the Certificate and Private Key that you generated as part of these steps to Cloud9.

**5.1 Create an IoT Thing**

In this section, you will create a Thing representing your Drone.

1. You may have lost your AWS Management Console and only see the Cloud9 console at this moment. To get back to the AWS Management Console without having to retype the URL, click **AWS Cloud9** at the top left corner of the page and select **Go To Your Dashboard**.
2. In the AWS Management Console, click **Services**, and then click **IoT Core** to open the the IoT Console.
3. Make sure you are in the same **Region** as your Cloud9 instance.
4. Click **Get started**.
5. Click **Manage** in the left menu.
6. Click **Register a thing**.
7. Click **Create a single thing** (any of the two buttons with that name works).
8. For **Name**, enter drone1 and click **Next**.
9. Click **Create thing without certificate** so that you skip the creation of the Certificate via the wizard.

The drone1 Thing has now been created.

**5.2 Create an IoT Policy**

In this section, you will create a Policy for authorization purposes that will be used in the next section.

1. Click **Secure** in the left menu.
2. Click **Policies**.
3. Click **Create a policy**.
4. For **Name**, enter IoTPolicy.
5. Click **Advanced mode**.
6. **Replace** the sample policy with the following policy which authorize to Connect to your AWS IoT Core endpoint, to Publish and Subscribe to an IoT Topic, receive messages from AWS IoT once subscribed and use the Discover API from Greengrass which will be used in a later exercise.
7. {
8. "Version": "2020-01-17",
9. "Statement": [
10. {
11. "Effect": "Allow",
12. "Action": [
13. "iot:Connect",
14. "iot:Publish",
15. "iot:Subscribe",
16. "iot:Receive",
17. "greengrass:Discover"
18. ],
19. "Resource": [
20. "\*"
21. ]
22. }
23. ]

}

1. Click **Create**.

You now have a Policy that provides authorizations.

**5.3 Create an IoT Certificate**

In this section, you will create a Certificate that will be used for Authentication.

1. Click **Certificates**.
2. Click **Create a certificate**.
3. Click **Create certificate** to automatically generate a Certificate, a Public Key and a Private Key using AWS IoT's Certificate Authority that you will then download. Note that you could create your own Certificate Authority. You could also create your own Private Key on your end and generate a Certificate Signing Request that you would upload here to be signed by AWS IoT's Certificate Authority. Both could work with the exercise, but you would have to adapt some of the later commands. For simplicity, use the One-click certificate creation.
4. Make sure that you do the following steps before going to the next screen as you would have to restart the creation of this Certificate.
5. Click the **Activate** button to activate the Certificate so it can be used later to connect to AWS IoT Core by your Thing.
6. Click the **Download** link next to **A certificate for this thing**.
7. **Rename** this file to certificate.pem.crt.
8. Click the **Download** link next to **A private key**. Note that you won't need the public key.
9. **Rename** this file to private.pem.key.
10. Click **Done**.

You now have a Certificate and Private Key that can be used to connect to your AWS IoT Core endpoint. However, this is only for authentication, you don't have any authorization yet associated to this Certificate. You may see that the Certificate is inactive, if you refresh the page, it should show as active. This will be done next.

**5.4 Associate the Policy and Thing to your Certificate**

In this section, you will attach a Policy to your Certificate to add authorizations and attach the drone1 Thing that is related to this Certificate.

1. Click on the certificate you just created.
2. Click **Actions > Attach policy**.
3. Put a **check mark** next to **IoTPolicy** and click **Attach**.
4. Click **Actions > Attach thing**.
5. Put a **check mark** next to **drone1** and click **Attach**.
6. If you click on Policies or Things on the left menu, you can see that both the IoTPolicy Policy and drone1 Thing have been attached.

**5.5 Upload your Certificate and Private Key to Cloud9**

In this section, you will upload the Certificate and Private Key that are now associated to drone1 into Cloud9 using the upload feature.

1. You may have closed your Cloud9 tab. If you did so, in the AWS Management Console, click **Services**, and then click **Cloud9** to open the Cloud9 dashboard. Click **Open IDE** under the IoTOnAWS environment card.
2. In the left menu, click the **drone1** folder to select it.
3. Click **File > Upload Local Files...**.
4. Make sure that Upload to folder is: */drone1*.
5. Click **Select files**. Note that the button may appear in grey, it isn't disabled and you can click on it.
6. **Browse** to the **Certificate** you have downloaded and renamed earlier: certificate.pem.crt. **Select it** and click **Open**. The Certificate should now be uploaded to your drone1 folder.
7. Click **Select files** again.
8. Browse to the Private Key you have downloaded and renamed earlier: private.pem.key. **Select it** and click **Open**. The Private Key should now be uploaded to your drone1 folder.
9. Click the **x** icon next to **Upload Files** to close that window.

Both drone1 Certificate and Private Key should now be in the *drone1* folder.

**6. Create Drone 2 IoT Thing, Certificate and Policy**

In this section you will use AWS CLI from the Cloud9 terminal to create all of the resources required for your Drone 2 to connect to AWS IoT. This includes the Drone Thing and the Certificate. The Policy has already been created from the previous step, so you will re-use it.

**6.1 Creation of drone2 IoT Core resources using the CLI**

1. In your Cloud9 **terminal**, enter the following commands to create the *drone2* Thing:
2. cd ~/environment/drone2

aws iot create-thing --thing-name drone2

1. To create the Certificate, enter the following command:

aws iot create-keys-and-certificate --set-as-active --certificate-pem-outfile certificate.pem.crt --private-key-outfile private.pem.key

This command will place the Certificate and Private Key in the certificate.pem.crt file and private.pem.key respectively. It will also output the certificateArn which you will re-use in the next command.

1. To attach the Policy to the Certificate, enter the following command. **Replace certificateArn\_changeme** with the value of the attribute certificateArn from the output of the previous command. It should be similar to: *arn:aws:iot:eu-west1-1:1234567890:cert/0f11db22dafacda87be0940dd5b2e010635916f541461ccf2d1c56ced0f343ee*

aws iot attach-policy --policy-name IoTPolicy --target certificateArn\_changeme

The command should not return anything if it worked.

1. To attach the drone2 Thing to the Certificate, enter the following command. **Replace certificateArn\_changeme** with the value of the attribute certificateArn from the output of the certificate creation command (2 steps above). It should be similar to: *arn:aws:iot:region:1234567890:cert/0f11db22dafacda87be0940dd5b2e010635916f541461ccf2d1c56ced0f343ee*

aws iot attach-thing-principal --thing-name drone2 --principal certificateArn\_changeme

The command should not return anything if it worked.

**6.2 Validation of creation of drone2 IoT resources**

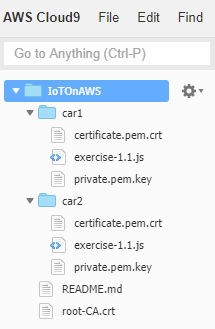
In this section, you will validate that all the resources for the creation of drone2 have been completed successfully. If you are missing any resources, you may want to try the commands above again, but first, **Delete** the drone2 Thing using the Console by selecting it and clicking **Actions > Delete** and try the steps from Section 6.1 again. If you still can't get those to work, use Section 5 to create everything from the AWS Management Console.

1. In the AWS Management Console, click **Services**, and then click **IoT Core** to open the IoT Core console.
2. Click **Manage**.
3. You should see *drone2* listed. If not, refresh the page as your browser may have cached the previous version of this page.
4. Click **drone2**.
5. Click **Security**.
6. You should see a Certificate card.
7. Click on that Certificate card.
8. Click on **Policies**.
9. You should see the IoTPolicy attached.

You have now created the drone2 Thing, its Certificate and Private Key to authenticate and attached the IoTPolicy to authorize the commands to execute later.

**7. Execute the code and validate telemetry**

At this point, you should have a directory structure that looks like the following in your Cloud9 environment:



In this section, you will execute the code of both Drones and validate that telemetry data is sent by both Drones using the AWS IoT MQTT Client in the AWS Management Console.

The code (droneconfig.js) requires one more resource to communicate with AWS IoT and that is your specific AWS IoT Endpoint. It will be stored in a file that will be used in all of the other exercises. To authenticate to your AWS IoT Endpoint, it will use the *certificate.pem.crt* (Certificate), *private.pem.key* (Private Key) and *root-CA.crt* (Certificate Authority public certificate). It then connects to AWS IoT and starts publishing random telemetry data every 10 seconds to the *aws/telemetry* IoT Topic.

Feel free to read the code to understand what is happening. Comments have been added to the code so it can be self explanatory.

**7.1 Execute the code**

In this section, you will first obtain your specific AWS IoT Endpoint that was automatically created for you when you created your first IoT Thing. This normally takes a few minutes to complete. Depending how fast you went with the steps above, you may have to wait a few minutes if the next command doesn't return anything. You will then open two Cloud9 terminals to simulate a connection to AWS IoT from both Drones. From each terminal, you will run the code which will use its appropriate Certificate and Private Key as you placed them in their respective folders.

1. In the Cloud9 **terminal**, enter the following command to get your specific AWS IoT Endpoint which will then be saved in the endpoint.json file:

aws iot describe-endpoint --endpoint-type iot:Data-ATS > ~/environment/endpoint.json

The command above will not output anything if it worked. Feel free to look at that file as it will be visible on the left side of the editor.

1. In the Cloud9 environment, click the **circled +** icon that is next to your current terminal and select **New Terminal**. You now have 2 different terminals.
2. In the **left terminal**, execute the following commands to start the code for drone1. Drone1's Certificate and Private Key, the Root Certificate Authority and your specific AWS IoT Endpoint will all be read from the files you created.
3. cd ~/environment/drone1

node droneconfig.js

You should see the following:

Connected to AWS IoT

Sending Drone telemetry data to AWS IoT for drone1

Sending Drone telemetry data to AWS IoT for drone1

...

1. In the **right terminal**, execute the following commands to start the code for drone2. Drone2's Certificate and Private Key, the Root Certificate Authority and your specific AWS IoT Endpoint will all be read from the files you created.
2. cd ~/environment/drone2

node droneconfig.js

You should see the following:

Connected to AWS IoT

Sending Drone telemetry data to AWS IoT for drone2

Sending Drone telemetry data to AWS IoT for drone2

...

Both Drones are now successfully sending telemetry data to the *aws/telemetry* IoT Topic.

**7.2 Subscribe to the aws/telemetry Topic**

In this section, you will use the AWS IoT MQTT Client in the AWS Management Console to subscribe to the *aws/telemetry* IoT Topic. While connected to this MQTT Client, it does consume connection minutes. The free tier is generous and provides 2,250,000 minutes of connection for free, but you should make sure to disconnect when you are not using it by browsing away from that page.

1. In the AWS Management Console, click **Services**, and then click **IoT Core** to open the IoT Core console.
2. Click **Test** in the left menu. It will open an AWS IoT MQTT Client where you can interact with any Topic that you have access to. This Client will automatically connect to your IoT Endpoint.
3. In the **Subscription topic**, enter aws/telemetry.
4. Click **Subscribe to topic**.

In the next 10 seconds, you should start seeing data being published by both Drones in the interface. You can see which Drone is sending the data by looking at the *device* attribute.

**Move onto Exercise 2 if you wish to send automated email alerts when a drone battery falls below 25%**

**Or if finishing just this Exercise, then proceed to step 8 below and delete the resources, to minimize costs on your account**

**8. Delete the resources created in this exercise**

While there are no connections nor data being transmitted to the IoT service, there will not be any charge for this exercise for that service.

The Cloud9 environment uses a t2.micro EC2 instance and an 8GiB Elastic Block Storage (EBS) volume which is what you are being charged on for the usage of Cloud9. The t2.micro has 750 hours of free utilization per month under the free tier for the first 12 months after opening your account. You have 30GiB of space for EBS covered under the free tier for the first 12 months of the opening of your account. If you are no longer under the free tier, you will incur a charge while the EC2 instance is running for the t2.micro instance and a charge for the EBS volume as long as your environment exists.

All of the other exercises will use the resources created as part of this exercise. The recommendation is to keep the IoT resources for the future exercises and to let Cloud9 stop by itself after 30 minutes of inactivity so you can keep the environment. If you decide not to keep some of those resources, you will have to do this exercise again before the other exercises.

If you would prefer to finish the workshop at this stage follow the steps 8.x below:

**8.1 Stop the Drones**

1. **Press Ctrl-c** in each of the Cloud9 **terminal** to stop them from interacting with AWS IoT.

**8.2 Stop the MQTT Client**

1. **Navigate away** from the **AWS IoT MQTT Client** page to disconnect from the client.

**8.3 Stop the Cloud9 environment**

The Cloud9 environment will automatically shut down after 30 minutes of inactivity. For your Cloud9 environment to be considered inactive, you need to close the browser tab. All of the settings will be saved.

1. Close the **browser tab** where your environment was running.

As the operating system is Amazon Linux, you are billed by the second during those 30 minutes of inactivity. If you are under the free tier, this would be covered. If you are no longer under the free tier, you can force a stop of the EC2 instance that runs your Cloud9 environment. This will have no effect on the future exercises.

1. In the AWS Management Console, click **Services**, and then click **EC2** to open the EC2 console.
2. Click **Instances** in the left menu.
3. Select the EC2 Instance that has a name that starts with **aws-cloud9-IoTOnAWS**.
4. Click **Actions > Instance State > Stop**

Congratulations! You have successfully completed this exercise.