CPD Lab 8: IoT and DevOps

Introduction

In this lab, for IoT you will first register a device on IoT AWS to send and receive messages. This is followed by another task in which you will make use of AWS MQTT client to publish message which triggers an SNS topic to send an SMS.

For DevOps, you will use CloudFormation (Infrastructure as code) to create an EC2 instance with LAMP stack. In another task you will use AWS DevOps to create a complete CodePipeline.

Tasks:

- Task 1: Registering a Device on AWS IoT Core
- Task 2: Configure and Test IoT Rule for Sending SMS
- Task 3: Creating a CloudFormation Template
- Task 4: Creating a CodePipeline Using CodeCommit and CodeDeploy
- Task 5 (**Optional**): Creating a Four Stage Pipeline
- Task 6: Releasing the Allocated Resources

Task 1: Registering an AWS Device on IoT Core

In this task you will cover the steps as listed in 'Getting Started with AWS IoT Core' https://docs.aws.amazon.com/iot/latest/developerguide/iot-gs.html

In this task you will register a Windows (or another operating system) computer as a device with AWS IoT core to send and receive messages using AWS IoT SDK.

- From under 'Services' in AWS Console select 'IoT Core'
- From the left navigation menu, click 'Onboard'
- Click 'Get Started' for 'Onboard a device'
- In 'Connect to AWS IoT', click 'Get Started'
- For 'Choose a platform', select 'Windows'
- For 'Choose a AWS IoT Device SDK' select 'Python'

  Note the displayed prerequisites at the bottom of the screen, "the device should have Python and Git installed and a TCP connection to the public internet on port 8883'

- Click 'Next'
- Under 'Register a thing', provide 'Name' as 'MyThing' and click 'Next Step'
- On next screen, 'Connect to AWS IOT', under 'Download connection kit for Windows', click the 'Windows' button which will download the connection kit, Click 'Next Step'
- On next screen, 'Configure and test your device'

To configure and test the device, perform the following steps.

Step 1: Unzip the connection kit on the device

```
unzip connect_device_package.zip
```

Step 2: Add execution permissions

```
Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope Process
```

Step 3: Run the start script. Messages from your thing will appear below

```
.\start.ps1
```

```
Waiting for messages from your device
```

- Unzip the downloaded connection kit downloaded above
- The files are:
  - MyThing.cert
  - MyThing.private.key
  - MyThing.public.key
  - Start.ps1
- On running the script start.ps1 in PowerShell, it downloads a folder <aws-iot-device-sdk-python> and root-CA from AWS
- The output in the PowerShell window is:

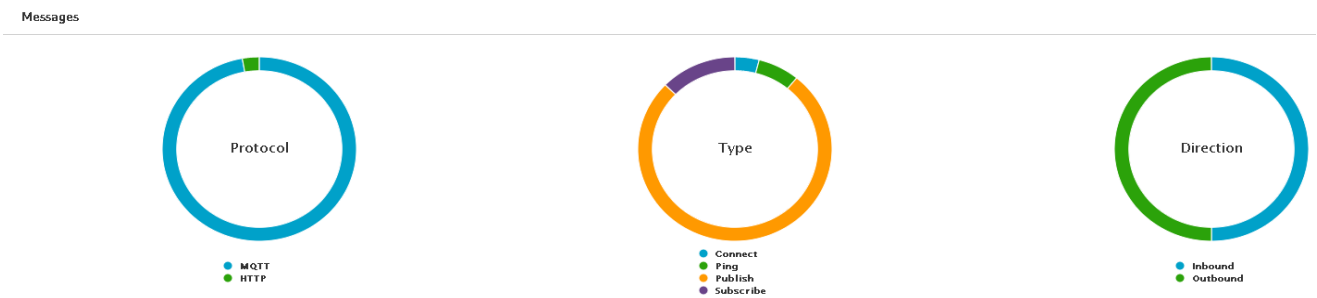
```
PS C:\Users\sna2\downloads\connect_device_package> .\start.ps1

Downloading AWS IoT Root CA certificate from AWS...

Cloning the AWS SDK...\n
Cloning into 'aws-iot-device-sdk-python'...
remote: Enumerating objects: 354, done.
Receiving objects: 100% (354/354), 197.61 KiB | 784.00 KiB/s, done.
remote: Total 354 (delta 0), reused 0 (delta 0), pack-reused 354
Resolving deltas: 100% (157/157), done.

Running pub/sub sample application...
```

- On the AWS console on the monitoring tab, you can see the messages



- Terminate the script by pressing Ctrl-C

- Delete the IoT device

Task 2: Configure and Test IoT Rule for Sending SMS

In this task you will use the IoT MQTT Client (instead of an actual IoT device) to send MQTT messages and configure an IoT rule to send Simple Notification Service (SNS) notification to a mobile phone number. You will follow the tutorial at:

https://docs.aws.amazon.com/iot/latest/developerguide/view-mqtt-messages.html
https://docs.aws.amazon.com/iot/latest/developerguide/config-and-test-rules.html

For this task you can use eu-west-1 (Ireland) as a region because SMS messaging is not available for eu-west-2 (London) region.
https://docs.aws.amazon.com/sns/latest/dg/sns-supported-regions-countries.html

Step 1: Send message from MQTT Test Client

You can use the AWS IoT MQTT client to better understand the MQTT messages sent by a device.

- Under 'Services' select 'IoT Core'
- In the left navigation menu, select 'Test'
- In 'Subscription Topic' enter 'my/Topic'
- Click 'Subscribe to topic'
- To emulate an IoT device sending a message to this topic
  - In the 'Publish section', select my/Topic
  - In the message payload section enter the following message

```
{
    "message": "Hello, world",
    "clientType": "MQTT client"
}
```

- Click on 'Publish to topic'
- The message gets published

Step 2: Configure and Test Rules

An AWS IoT rule can be configured to take some action such as invoking a Lambda function, writing data to S3 etc. In this step, you will create and configure a rule to send the data received from the MQTT Test Client (or a device) to an Amazon SNS topic.

Create an Amazon SNS topic

- Under 'Services' in AWS Console select 'SNS'
- In the left navigation menu, select 'Topics'

- Click ‘Create Topic’
- Enter name as ‘MyIoTSNSTopic’
- Enter ‘Display name’ as ‘My IoT SNS Topic’
- Click ‘Create topic’

Subscribe to the Amazon SNS topic

- In the left navigation menu, select ‘Subscription’
- Click ‘Create subscription’
- For ‘Topic ARN’ enter the ARN of the SNS topic that you have created
- For ‘Protocol’ select ‘SMS’
- Enter a UK mobile number (with the country code) under ‘Endpoint’
- Click ‘Create subscription’
- You can quickly confirm that an SMS will be received by the configured mobile number by clicking ‘Text messaging’ in the left navigation pane and clicking ‘Publish text message’ and filling in the details

Create a rule that sends a message to the Amazon SNS topic when a message is received from your device

You will create a rule that uses SNS to send an SMS to the mobile phone number.

- Navigate to AWS IoT console
- In the left navigation menu, select ‘Act’
- Click ‘Create a rule’
- Enter ‘MyIoTRule’ as name
- Under ‘Rule Query Statement’

```
SELECT * FROM ‘my/topic’
```

- Under ‘Set one or more actions’, click ‘Add action’
- Select ‘Send a message as an SNS push notification’
- Click ‘Configure action’
- Under ‘Configure action’ click ‘Select’ to expand the ‘SNS target’ and select the SNS topic created earlier.
- Under ‘Message format’ select JSON
- For ‘Choose or create a role to grant AWS IoT to perform this action’
    - Click ‘Create Role’
    - Enter ‘MySNSRole’ as name and click ‘create role’
    - Choose the role and click ‘Add action’
    - On ‘Create a Rule’ click ‘Create rule’

Test the rule using the MQTT client

AWS IoT MQTT client can be used to test the rule.

- In the left navigation menu, select 'Test'
- Under 'Publish' section, enter 'my/Topic' under 'Specify a topic and a message to publish with a QoS of 0'
- In the message payload section, enter

```
{
"default": "Hello, from AWS IoT console",
"message": "Hello, from AWS IoT console"
}
```

- Click 'Publish to topic'
- An SMS message should be received on the previously entered mobile phone number.

Deleting the resources

- Delete the SNS Topic
- Delete the rule created in AWS IoT

Task 3: Creating a CloudFormation Template

AWS CloudFormation allows you to quickly and easily deploy your infrastructure resources and applications in AWS.

In this task you will experiment with CloudFormation and briefly look at CloudFormation designer and CloudFormer.

- Under 'Services' select 'CloudFormation'
- Click on 'Stack' and 'Create Stack'
- There are three options presented as below, out of which we will be using a sample template but will look at few other options:
    - Template is ready
    - Use a sample template
    - Create template in Designer – Select it and click 'Create template in designer' and in the designer experiment with drag and drop of few resources and note the corresponding template
- Select 'Use a sample template' and under 'Select a sample template' select in the dropdown under 'Tools', 'CloudFormer' and click 'View in Designer'. Observe to resources in the designer.
- Click back to proceed to the previous screen
- Select 'Use a sample template', and select 'LAMP Stack' from the drop-down under 'Select a sample template'
- Click 'Next'
- Provide 'Stack name' as 'MyApacheServer', and fill in details for DBPassword, DBRootPassword, and DBUser
- Change 'InstanceType' from t2.small to t2.micro
- Provide a KeyName

- Click ‘Next’
- In ‘Configure stack options’, Navigate to ‘Advanced options’ to observe the various settings and click ‘Next’
- In ‘Review MyApacheServer’ look through the settings
- Click ‘Create stack’
- From the ‘CloudFormation’ page you can see the information under StackInfo, Events, Resources, and Parameters. You can view the template under the ‘template’ tab. You can find and connect to the website URL under ‘Outputs’
- You can also view the EC2 instance and the security group created by the template under ‘Instances’ and ‘Security Groups’ in the EC2 Dashboard
- Under ‘CloudFormation’ select ‘MyApacheServer’ and click ‘Delete’, confirm by clicking ‘Delete stack’
- This would delete both the EC2 instance and security group created by the template

Task 4: Creating a CodePipeline Using CodeCommit and CodeDeploy

In this task you will create a CodePipeline by following the AWS tutorial at:

https://docs.aws.amazon.com/codepipeline/latest/userguide/tutorials-simple-codecommit.html

Pre-requisites

Follow the steps (only follow steps 1, 2 and 3) for “Setup for HTTPS Users Using Git Credentials” at https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up-gc.html to:

1. Create an IAM user
2. Git installation on your local machine

Step 1: Create a CodeCommit Repository

You will need a CodeCommit repository and a local repository from where you can push the code to CodeCommit repository. The pipeline will be triggered when you push a change to CodeCommit repository.

CodeCommit Repository

- From the ‘Services’ search for ‘CodeCommit’
- Note the region where you want to create the repository and pipeline
- Select ‘Repositories’ from the left navigation menu and click ‘Create repository’

- Provide 'Repository name' as 'MyRepo'
- Click 'Create'

Local Repository

- Create a temporary folder for local code repository e.g., C:\Temp
- Using Command window, change to c:\temp
- On the AWS console, click 'Clone URL' and select from the drop-down ' Clone HTTPS'
- In the command window in the local machine enter 'git clone' and paste the URL copied from the AWS console

```
C:\Temp>git clone https://git-codecommit.eu-west-
2.amazonaws.com/v1/repos/MyRepo
```

- You will be asked for the credentials saved in pre-requisite above.

Step 2: Add sample code to your CodeCommit repository

- Run Download sample code SampleApp_Linux.zip from https://docs.aws.amazon.com/codepipeline/latest/userguide/samples/SampleApp_Linux.zip
- Unzip the zipped file into the local directory that you have created in Step1, Task4, c:\temp taking care to place the unzipped files and folders into your local repository
- Change to your local repo using command Window

```
C:\temp\MyRepo
```

- Run the command

```
git add –A
```

- Run the following command to commit the files with a commit message

```
git commit –m "Add sample application files"
```

- Run the following command to push the files from your local repository to the CodeCommit repository

```
git push
```

The files are now ready to be included in the CodePipeline

Step 3: Create an EC2 Linux Instance and Install the CodeDeploy Agent

In this step, you create the EC2 instance where you deploy a sample application. You will install the CodeDeploy agent on the EC2 instance, the agent is a software package that enables an instance to be used in CodeDeploy deployments.

Create an instance role

- Open IAM console on AWS Console
- From the left navigation menu, select 'Roles'
- Click on 'Create Role'
- Under 'Select type of trusted entity' click 'AWS service'
- Under 'Choose a use case', select 'EC2' and then click 'Next:Permissions'
- Under 'Attach permissions policies', search for 'AmazonEC2RoleforAWSCodeDeploy' in the search field
- Select the policy and click 'Next:Tags' and the click 'Next:Review'
- For 'Role Name' enter 'EC2InstanceRole' and click 'Create Role'

Launch an EC2 instance

- Open EC2 Dashboard on AWS Console
- Click on 'Launch Instance'
- Step 1: Choose an AMI
    - Click 'Select' in front of 'Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0cb790308f7591fa6'
- Step 2: Choose an Instance type
    - Select t2.micro and click 'Next: Configure Instance Details'
- Step 3: configure instance details
    - Enter 1 for 'Number of Instances'
    - For 'Auto-assign Public IP', select 'Enable' from the drop-down
    - For IAM role, select the IAM role created above 'EC2InstanceRole'
    - Open 'Advanced Details' and in the 'User data' enter

```
#!/bin/bash
yum -y update
yum install -y ruby
yum install -y aws-cli
cd /home/ec2-user
aws s3 cp s3://aws-codedeploy-us-east-2/latest/install . --
          region us-east-2
chmod +x ./install
./install auto
```

This would install CodeDeploy agent on the EC2 instance once it gets created.

    - Click 'Next: Add storage'
- Step 4: Add storage
    - Click 'Next: Add Tags'
- Step 5: Add Tags
    - Click 'Add Tag' and enter 'Name' for Key and 'MyCodePipelineDemo' for Value

    - Click 'Next: Configure Security Group'
- Step 6: Configure Security Group
    - For 'Assign a security group' select 'Create a new security group'
    - For SSH, under 'Source' select 'My IP'
    - Click 'Add Rule', select HTTP for 'Type' and for 'Source' select 'My IP'
    - Click 'Review and Launch'
- Step 7: Review Instance Launch
    - Click 'Launch'
    - In 'Select an existing key pair or create a new key pair', choose an existing KeyPair or create a new one and click 'Launch Instance'

Step 4: Create an Application in CodeDeploy

An application is a resource that contains the software application you want to deploy.

Create a CodeDeploy service role

- Open IAM console on AWS Console
- From the left navigation menu, select 'Roles'
- Click on 'Create Role'
- Under 'Select type of trusted entity' click 'AWS service'
- Under 'Choose a use case', select 'CodeDeploy' and then click 'Next:Permissions'
- Under 'Attach permissions policies', search for 'AWSCodeDeployRole' in the search field
- The policy 'AWSCodeDeployRole' is pre-selected, click 'Next:Tags' and the click 'Next:Review'
- For 'Role Name' enter 'MyCodeDeployRole' and click 'Create Role'

Create an application in CodeDeploy

- Select 'CodeDeploy' under 'Services in AWS Console
- Select 'Applications' from the left navigation menu and click 'Create Application'
- Enter 'MyDemoApplication' for the 'application name'
- Select 'EC2/On-premises' from the 'Compute platform' drop-down
- Click 'create application'

Create a deployment group in CodeDeploy

A deployment group is a resource that defines deployment-related settings like which EC2 instances to deploy to and how fast to deploy them.

- Click 'Create deployment group' in 'MyDemoApplication' page
- For 'Enter a deployment group name' enter 'MyDemoDeploymentGroup'

- For ‘Enter a service role’ select the role created earlier ‘MyCodeDeployRole’
- Select ‘In-place’ under ‘Deployment type’
- Select ‘Amazon EC2 Instances’ under ‘Environment Configuration’
- In the ‘Tag group 1’, for ‘Value’ select the Key/value created earlier, Name/MyCodePipelineDemo
- Under ‘Deployment settings’, for ‘Deployment configuration’ select ‘CodeDeployDefault.OneAtATime’
- Under ‘Load balancer’, clear the check mark for ‘Enable load balancing’
- Expand the ‘Advanced-optional’
    - If any alarms are listed under ‘Alarms’ then check ‘ignore alarm configuration’
- Click ‘Create deployment group’

Step 5: Create your First Pipeline in CodePipeline

The pipeline that you will create in this step will run automatically whenever code is pushed to the CodeCommit repository.

- Select ‘CodePipeline’ under ‘Services’ in AWS Console
- From the left navigation menu, select Pipelines’ and click ‘Create Pipeline’
- Step 1: Choose pipeline settings
    - For ‘Pipeline name’ enter ‘MyFirstPipeline’
    - Under ‘Service role’, select ‘New service role’
    - Click ‘Next’
- Step 2: Add source stage
    - From the drop-down for ‘Source provider’, select ‘AWS CodeCommmit’
    - For ‘Repository name’ select the CodeCommit repository name created in Step 1.
    - For ‘Branch name’, select ‘master’
    - Click ‘Next’
- Step 3: Add build stage
    - Click ‘Skip build stage’ and then click ‘Skip’again in the dialog box as the code being deployed does not require a build service. However, if the source code needs to be built before it is deployed to EC2 instances then ‘CodeBuild’ can be configured in this step.
- Step 4: Add deploy stage
    - Under ‘Deploy provider’ select ‘AWS CodeDeploy’
    - Under ‘Application name’ select ‘MyDemoApplication’
    - Under ‘DeploymentGroup’ select ‘MyDemoDeploymentGroup’
    - Click ‘Next’
- Step 5: Review
    - Review and Click ‘Create pipeline’

The pipeline starts running after it is created and downloads the code from your CodeCommit repository and creates a CodeDeploy deployment to your EC2 instance.

To verify the results:

- After 'Succeeded' is displayed for the pipeline status, choose Details. This opens the CodeDeploy console. If Succeeded is not displayed see Troubleshooting CodePipeline.
- Copy the public DNS of the EC2 instance to a browser and you should see the message, "Congratulations-this application was deployed using AWS CodeDeploy'

Step 6: Modify Code in your CodeCommit Repository

The pipeline runs whenever code changes are made to your CodeCommit repository. In this step, you make changes to the HTML file in the local repository and the CodePipeline runs as you push these changes, with the changes visible at the web address you accessed earlier.

- Replace the index.html in the local computer repository with the file provided in "CPD Lab8 Resources.zip"

- Change directory to your local repo on the local computer:

  cd C:\temp\MyDemoRepo

- Run the following commands to push the changes to CodeCommit repository

  ```
  git commit –am "Updated sample application files"

  git push
  ```

- You can verify that the pipeline ran successfully by viewing the initial progress of the pipeline. The status of each stage changes from No executions yet to In Progress, and then to either Succeeded or Failed. The running of the pipeline should be complete within a few minutes.
- Refresh the browser to display the updated webpage.

Step 7: Clean Up Resources

- Delete the CodePipeline: on AWS console, select the CodePipeline and click 'delete' resources
- Delete the CodeCommit repository
- Delete the CodeDeploy deployment group and CodeDeploy application
- Delete the CodeCommit repository
- Terminate the EC2 instance

Task 5: (Optional) Creating a four stage pipeline

This tutorial will walk you through the creation of a four-stage pipeline that uses a GitHub repository for your source, a Jenkins build server to build the project, and a

CodeDeploy application to deploy the built code to a staging server. After the pipeline is created, you will edit it to add a stage with a test action to test the code, also using Jenkins.

https://docs.aws.amazon.com/codepipeline/latest/userguide/tutorials-four-stage-pipeline.html

Task 6: Releasing the allocated resources

Make sure that you have released all the resources created in tasks above including:

- Delete the IoT device from Task 1
- Delete the SNS Topic and the rule created in Task 2
- Delete the stack, EC2 instance and security group from Task 3
- Delete the CodePipeline, CodeCommit repository, CodeDeploy deployment group, CodeDeploy application, CodeCommit repository, and terminate the EC2 instance from Task 4

Links

The links below are for your reference only in case further information is required to help complete tasks above:

Task 1

1. Registering a Device on AWS IoT Core
   https://docs.aws.amazon.com/iot/latest/developerguide/iot-gs.html

Task 2

2. Configure and Test IoT Rue for Sending SMS
   https://docs.aws.amazon.com/iot/latest/developerguide/view-mqtt-messages.html
   https://docs.aws.amazon.com/iot/latest/developerguide/config-and-test-rules.html

Task 3

3. Creating a CloudFormation Template
   https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-guide.html

Task 4

4. Creating a Code Pipeline Using CodeCommit and CodeDeploy
   https://docs.aws.amazon.com/codepipeline/latest/userguide/tutorials-simple-codecommit.html