Raspberry PI - Weather Station

User Manual - AWS Version

December 3, 2018

"Patience is a key element of success"

Joynal Abedin

Version History

Date	Version	Author	Comments
12/3/2018	v1.0	Joynal Abedin	Initial Formatting
12/5/2018	v1.1	Joynal Abedin	Completed Prerequisite, SensorData
12/5/2018	v1.1.2	Joynal Abedin	Update SensorData, Include Disclaimer
12/6/2018	v1.1.3	Joynal Abedin	Completed Verify-API, Updated SensorData, Updated Disclaimer

Table of Contents

Disclaimer

Use this manual at your own risk and responsibility. I will not be responsible for any AWS Costs or consequences that may result from following this manual. I'd highly encourage you to read the AWS Free Tier policy and user agreement before continuing with this document.

AWS Free Tier Policy: https://aws.amazon.com/free/
User agreement: https://aws.amazon.com/agreement/

Manual usage

The symbol with one vertical stroke which is also known as dollar symbol ($) will represent commands in terminal. So, if you see a line followed by $, it means you have to open up a terminal and execute that line of command.

This guide will also assume that you have sufficient understanding of your own system and how to work around terminal which you have learned in 4420 which is the prerequisite of this course.

Prerequisite

Get an AWS Account

If you already have an account with AWS, skip this step

1. Visit the Amazon Web Service at https://aws.amazon.com
2. Choose Create an AWS Account.

   Note: If you've signed in to AWS recently, it might say Sign In to the Console. Otherwise it should say Create an AWS Account.

   If the option "Create a new AWS Account" isn't displayed, first choose to Sign in to a different account, and select "Create a new AWS Account".

3. Type the requested account information, and then choose Continue.
4. Choose Personal .
5. Type the requested personal information.
6. Read and check the AWS Customer Agreement.
7. Choose Create Account and Continue
8. Add a payment method (if you cross the free tier limit, AWS may start charging you. Here's more detail about free tier https://aws.amazon.com/free)
9. Verify your phone number
10. Choose the Basic support plan
11. If all the above steps went well, congratulations! Now, you have an AWS Account.

Install AWS CLI on your machine

Requirements

- Python 2 version 2.6.5+ or Python 3 version 3.3+
- Windows, Linux, macOS or Unix

Install the AWS CLI on macOS/Linux

Step 1: Install Python 3

*If you already have it installed, skip this step

```
$ python --version
```

or

```
$ python3 --version
```

If you don't have python installed. Get it from here
https://www.python.org/downloads/

Once python is being installed, run the following:

```
$ pip --version
```

If you already have pip, skip the below step for pip installation.

Step 2: Install pip:

```
$ curl -O https://bootstrap.pypa.io/get-pip.py
```

```
$ python get-pip.py --user
```

Add the executable path to your PATH variable: ~/.local/bin

Find your shell profile script.
Add an export command to your profile script

```
$ export PATH=~/.local/bin:$PATH
```

You can now close the terminal and re-open it again to load the profile to your session. Or you can simply do `$ source ~/.bash_profile`

Move on to the next step if you have pip installed successfully.

Step 3: Install AWS CLI

Depend on your system, if it's called pip or pip3, change the below command accordingly.
For me it's pip3.

```
$ pip3  install awscli
```

```
$ aws --version
```

Output: AWS CLI [version number]

Free Tips: If you don't see a version number outputted, then you'd need to add it to your command line path.

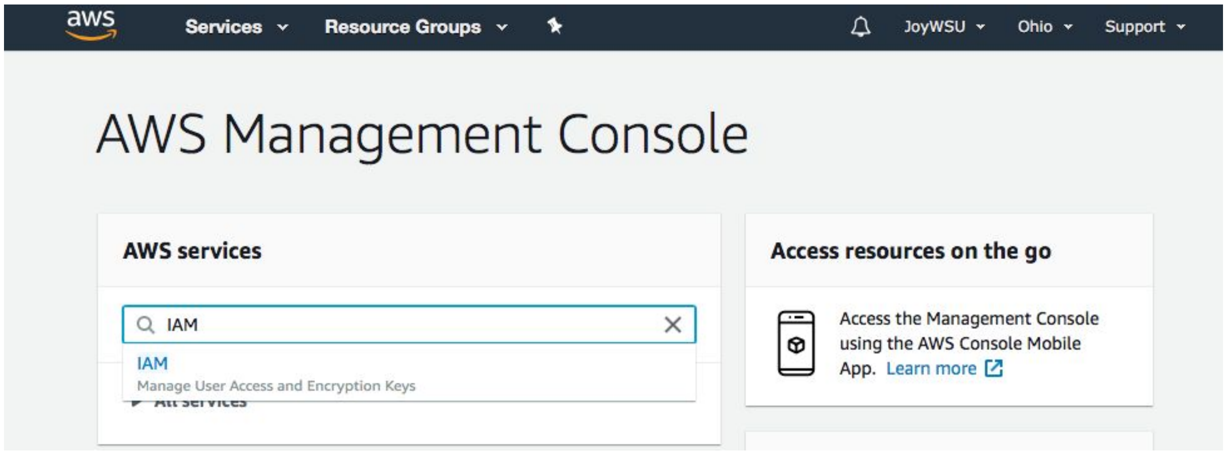
Install the AWS CLI on Windows

MSI Installer for Windows people here:
https://docs.aws.amazon.com/cli/latest/userguide/awscli-install-windows.html

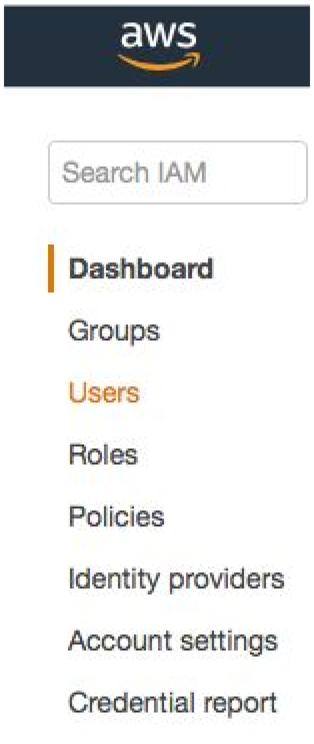
Create an IAM User

Go to https://aws.amazon.com/console
Click on "Sign In to the Console"
Enter your email address
Click next
Enter your password
Click on "Sign in"

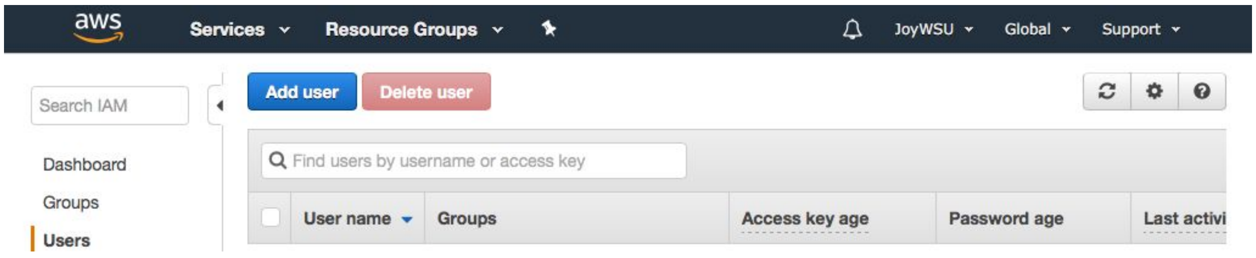
From the Home page
Search "IAM" in AWS services and click on the result where it says "IAM"


aws
Services
Resource Groups
JoyWSU
Ohio
Support
AWS Management Console
AWS services
IAM
IAM
Manage User Access and Encryption Keys
All services
Access resources on the go
Access the Management Console using the AWS Console Mobile App. Learn more

From the left menu choose “Users”



Click on “Add user”



Set user details:

Type the username “AWS-ADMIN-WSU”

Select AWS access type:

Access type: check both “Programmatic access”

On the bottom right click “Next: Permissions”

Set permissions

Select Attach existing policies directly
In the Filter policies, search “AdministratorAccess”
Select the Policy Name that says “AdministratorAccess”

On the bottom right click “Next: Tags”
On the bottom right click “Next: Review”
On the bottom right click “Create User”

Click on Download.csv button
Note down the value from the column “Access key ID” and “Secret access key”

Configure your CLI on terminal

```
$ aws configure --profile joyWSU

AWS Access Key ID [None]: your_access_key_id
AWS Secret Access Key [None]: your_secret_access_key
Default region name [None]: us-east-1
Default output format [None]:
```

Create S3 Bucket

```
$ aws s3api create-bucket --bucket raspberry-pi-wsu-senior --region
us-east-1
```

```
Response:

{ "Location": "/raspberry-pi-wsu-senior" }
```

SensorData

Upload code to S3 Bucket

```
$ aws s3 cp sensordata.zip s3://raspberry-pi-wsu-senior
```

Create new policy

Make sure your in the policies directory in the deployment_package

```
$ aws iam create-policy --policy-name sensordata-policy --policy-document
file://sensordata.json
```

```
Response: {
    "Policy": {
        "PolicyName": "sensordata-policy",
        "PolicyId": "ANPAJHNSZ2KAFXC6VLEMK",
        "Arn": "arn:aws:iam::667671192261:policy/sensordata-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2018-12-04T03:21:03Z",
        "UpdateDate": "2018-12-04T03:21:03Z"
    }
}
```

From the response, copy the ARN for later usage
For example:
Arn: “arn:aws:iam::667671192261:policy/sensordata-policy”

Create IAM Role

```
$ aws iam create-role --role-name WSU-SENSOR-ROLE
--assume-role-policy-document file://trusty.json

---------------------------------------------------------------------------
Response: {
    "Role": {
        "Path": "/",
        "RoleName": "WSU-SENSOR-ROLE",
        "RoleId": "AROAI2K2XLICPSCLLS376",
        "Arn": "arn:aws:iam::667671192261:role/WSU-SENSOR-ROLE",
        "CreateDate": "2018-12-04T03:26:31Z",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "lambda.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }
    }
}
```

Copy the **ARN** and **Role** name for later usage

For example:
“RoleName”: “WSU-SENSOR-ROLE”
“Arn”: “arn:aws:iam::667671192261:role/WSU-SENSOR-ROLE”

Note: Make sure to replace the red highlighted portion with the appropriate arn name from the previous steps.

Attach the policy to the role

```
$ aws iam attach-role-policy --role-name WSU-SENSOR-ROLE --policy-arn
arn:aws:iam::667671192261:policy/sensordata-policy
```

Get VPC info

Make note of the VpcId from the response

```
$ aws ec2 describe-vpcs --filters Name=isDefault,Values=true

Response: {
    "Vpcs": [
        {
            "CidrBlock": "172.31.0.0/16",
            "DhcpOptionsId": "dopt-3ed09845",
            "State": "available",
            "VpcId": "vpc-0310bf79",
            "OwnerId": "667671192261",
            "InstanceTenancy": "default",
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-a8fe45c4",
                    "CidrBlock": "172.31.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ],
            "IsDefault": true
        }
    ]
}
```

Make note of all of the SubnetId from the response. You'll use it later. They're highlighted with red font followed by "SubnetId" parameter

```
$ VPC=vpc-0310bf79
$ aws ec2 describe-subnets --filters Name=vpc-id,Values=$VPC
Name=default-for-az,Values=true

Response: {
    "Subnets": [
        {
            "AvailabilityZone": "us-east-1f",
            "AvailabilityZoneId": "use1-az5",
            "AvailableIpAddressCount": 4088,
            "CidrBlock": "172.31.48.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": true,
            "State": "available",
            "SubnetId": "subnet-b44713bb",
            "VpcId": "vpc-0310bf79",
            "OwnerId": "667671192261",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn":
"arn:aws:ec2:us-east-1:667671192261:subnet/subnet-b44713bb"
        },
        {
            "AvailabilityZone": "us-east-1c",
            "AvailabilityZoneId": "use1-az6",
            "AvailableIpAddressCount": 4088,
            "CidrBlock": "172.31.32.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": true,
            "State": "available",
            "SubnetId": "subnet-958b14c9",
            "VpcId": "vpc-0310bf79",
            "OwnerId": "667671192261",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn":
"arn:aws:ec2:us-east-1:667671192261:subnet/subnet-958b14c9"
        },
        {
            "AvailabilityZone": "us-east-1e",
            "AvailabilityZoneId": "use1-az3",
            "AvailableIpAddressCount": 4091,
            "CidrBlock": "172.31.64.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": true,
            "State": "available",
            "SubnetId": "subnet-7872d446",
            "VpcId": "vpc-0310bf79",
            "OwnerId": "667671192261",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn":
"arn:aws:ec2:us-east-1:667671192261:subnet/subnet-7872d446"
        },
        {
            "AvailabilityZone": "us-east-1d",
            "AvailabilityZoneId": "use1-az1",
            "AvailableIpAddressCount": 4091,
            "CidrBlock": "172.31.0.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": true,
            "State": "available",
            "SubnetId": "subnet-ef980a88",
            "VpcId": "vpc-0310bf79",
            "OwnerId": "667671192261",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn":
"arn:aws:ec2:us-east-1:667671192261:subnet/subnet-ef980a88"
        },
        {
            "AvailabilityZone": "us-east-1a",
            "AvailabilityZoneId": "use1-az2",
            "AvailableIpAddressCount": 4090,
            "CidrBlock": "172.31.80.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": true,
            "State": "available",
            "SubnetId": "subnet-078f1629",
            "VpcId": "vpc-0310bf79",
            "OwnerId": "667671192261",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn":
"arn:aws:ec2:us-east-1:667671192261:subnet/subnet-078f1629"
        },
        {
            "AvailabilityZone": "us-east-1b",
            "AvailabilityZoneId": "use1-az4",
            "AvailableIpAddressCount": 4091,
            "CidrBlock": "172.31.16.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": true,
            "State": "available",
            "SubnetId": "subnet-881c3ec2",
            "VpcId": "vpc-0310bf79",
            "OwnerId": "667671192261",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn":
"arn:aws:ec2:us-east-1:667671192261:subnet/subnet-881c3ec2"
        }
    ]
}
```

Get the Security Group ID

Make note of the “GroupID” parameter’s value.

```
$ aws ec2 describe-security-groups --filters Name=vpc-id,Values=$VPC
--group-names default

Response: {
    "SecurityGroups": [
        {
            "Description": "default VPC security group",
            "GroupName": "default",
            "IpPermissions": [
                {
                    "FromPort": 3306,
                    "IpProtocol": "tcp",
                    "IpRanges": [
                        {
                            "CidrIp": "172.31.42.214/32",
                            "Description": "ec2-for-AMI"
                        },
                        {
                            "CidrIp": "68.01.01.123/32",
                            "Description": "joy home"
                        }
                    ],
                    "Ipv6Ranges": [],
                    "PrefixListIds": [],
                    "ToPort": 3306,
                    "UserIdGroupPairs": [
                        {
                            "Description": "ASG Security group",
                            "GroupId": "sg-0a0876f20bed41e27",
                            "UserId": "667671192261"
                        },
                        {
                            "Description": "vpc security",
                            "GroupId": "sg-2554ce65",
                            "UserId": "667671192261"
                        }
                    ]
                }
            ],
            "OwnerId": "667671192261",
            "GroupId": "sg-2554ce65",
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "Ipv6Ranges": [],
                    "PrefixListIds": [],
                    "UserIdGroupPairs": []
                }
            ],
            "VpcId": "vpc-0310bf79"
        }
    ]
}
```

Create new Lambda function

```
$ aws lambda create-function --function-name sensordata --runtime python3.6
--role arn:aws:iam::667671192261:role/WSU-SENSOR-ROLE --handler
lambda_function.handler --code
S3Bucket=raspberry-pi-wsu-senior,S3Key=sensordata.zip --vpc-config
SubnetIds=subnet-b44713bb,subnet-958b14c9,subnet-ef980a88,subnet-078f1629,s
ubnet-881c3ec2,SecurityGroupIds=sg-2554ce65

Response: {
    "FunctionName": "sensordata",
    "FunctionArn":
"arn:aws:lambda:us-east-1:667671192261:function:sensordata2",
    "Runtime": "python3.6",
    "Role": "arn:aws:iam::667671192261:role/WSU-SENSOR-ROLE",
    "Handler": "lambda_function.handler",
    "CodeSize": 1984640,
    "Description": "",
    "Timeout": 3,
    "MemorySize": 128,
    "LastModified": "2018-12-05T19:38:51.311+0000",
    "CodeSha256": "Z1AHTgsmJOQQDttuLZXujdsOwSIg0qdJekuE7tsbGtE=",
    "Version": "$LATEST",
    "VpcConfig": {
        "SubnetIds": [
            "subnet-958b14c9",
            "subnet-ef980a88",
            "subnet-078f1629",
            "subnet-881c3ec2",
            "subnet-b44713bb"
        ],
        "SecurityGroupIds": [
            "sg-2554ce65"
        ],
        "VpcId": "vpc-0310bf79"
    },
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "RevisionId": "d23924a6-f40f-4d8d-9568-80e03942cf33"
}
```

Setup API Gateway

Create the API

```
$ aws apigateway create-rest-api --name sensordata-API
--endpoint-configuration types=REGIONAL

---------------------------------------------------------------------------
Response: {
    "id": "pkqwe266xg",
    "name": "sensordata-API",
    "createdDate": 1543900164,
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "REGIONAL"
        ]
    }
}
```

From the previous step, save the “id” parameter and replace it below with API value. Now, you can get the API root resource ID as well.

```
$ API=pkqwe266xg
$ aws apigateway get-resources --rest-api-id $API

Response: {
    "items": [
        {
            "id": "eu9ur0jp06",
            "path": "/"
        }
    ]
}
```

Copy the id from the response into the clipboard.

Create a Resource in the API

Set the variable ROOT_RESOURCE to the id that you copied from previous step.

```
$ ROOT_RESOURCE=eu9ur0jp06
$ aws apigateway create-resource --rest-api-id $API --path-part
'sensordata' --parent-id $ROOT_RESOURCE

Response: {
    "id": "txgaza",
    "parentId": "eu9ur0jp06",
    "pathPart": "sensordata",
    "path": "/sensordata"
}
```

Create POST Method on the Resource

```
$ RESOURCE=txgaza
$ aws apigateway put-method --rest-api-id $API --resource-id $RESOURCE
--http-method POST --authorization-type "NONE" --api-key-required

Response: {
    "httpMethod": "POST",
    "authorizationType": "NONE",
    "apiKeyRequired": true
}
```

```
$ aws apigateway put-integration --rest-api-id $API --resource-id $RESOURCE
--http-method POST --type AWS_PROXY --integration-http-method POST --uri
arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambd
a:us-east-1:667671192261:function:test1/invocations

Response: {
    "type": "AWS_PROXY",
    "httpMethod": "POST",
    "uri":
"arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lamb
da:us-east-1:667671192261:function:test1/invocations",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "cacheNamespace": "txgaza",
    "cacheKeyParameters": []
}
```

```
$ aws apigateway put-method-response --rest-api-id $API --resource-id $RESOURCE
--http-method POST --status-code 200 --response-parameters
"method.response.header.Access-Control-Allow-Origin=true" --response-models
application/json=Empty

Response: {
    "statusCode": "200",
    "responseParameters": {
        "method.response.header.Access-Control-Allow-Origin": true
    },
    "responseModels": {
        "application/json": "Empty"
    }
}
```

```
$ aws apigateway put-integration-response --rest-api-id $API --resource-id
$RESOURCE --http-method POST --status-code 200 --selection-pattern ""
--response-templates '{"application/json": "{\"json\": \"Empty\"}"}'
--response-parameters
'{"method.response.header.Access-Control-Allow-Origin":"'"'"'*'"'"'"}'


Response: {
    "statusCode": "200",
    "selectionPattern": "",
    "responseParameters": {
        "method.response.header.Access-Control-Allow-Origin": "'*'"
    },
    "responseTemplates": {
        "application/json": "{\"json\": \"Empty\"}"
    }
}
```

```
$ aws apigateway create-deployment --rest-api-id $API --stage-name final
--stage-description 'Final Stage' --description 'Deployment for the weather
station pi'

Response: {
    "id": "uc7mww",
    "description": "Deployment for the weather station pi",
    "createdDate": 1543986142
}
```

```
$ aws lambda add-permission --function-name test1 --statement-id
apigateway-testt-1 --action lambda:InvokeFunction --principal
apigateway.amazonaws.com --source-arn
"arn:aws:execute-api:us-east-1:667671192261:8h35no6zsg/*/POST/sensordata"


Response: {
    "Statement":
"{\"Sid\":\"apigateway-testt-1\",\"Effect\":\"Allow\",\"Principal\":{\"Serv
ice\":\"apigateway.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction\",\"
Resource\":\"arn:aws:lambda:us-east-1:667671192261:function:test1\",\"Condi
tion\":{\"ArnLike\":{\"AWS:SourceArn\":\"arn:aws:execute-api:us-east-1:6676
71192261:8h35no6zsg/*/POST/sensordata\"}}}"
}
```

```
$ aws apigateway create-api-key --name 'PI-API-Key' --description 'for
development' --enabled --stage-keys restApiId=$API,stageName=final

Response: {
    "id": "a56g1r3mzc",
    "value": "g7gwWjktqX5rfQLQS7tIs2nXmzPYSpem4SH0W5ze",
    "name": "PI-API-Key",
    "description": "for development",
    "enabled": true,
    "createdDate": 1543986612,
    "lastUpdatedDate": 1543986612,
    "stageKeys": [
        "8h35no6zsg/final"
    ]
}
```

```
$ aws apigateway create-usage-plan --name "pi-api-key-usagePlan"
--api-stages apiId=$API,stage=final --description "usage plan for pi api
key" --throttle burstLimit=10,rateLimit=150 --quota
limit=500,offset=0,period=MONTH

Response:{
    "id": "u9dvxc",
    "name": "pi-api-key-usagePlan",
    "description": "usage plan for pi api key",
    "apiStages": [
        {
            "apiId": "8h35no6zsg",
            "stage": "final"
        }
    ],
    "throttle": {
        "burstLimit": 10,
        "rateLimit": 150.0
    },
    "quota": {
        "limit": 500,
        "offset": 0,
        "period": "MONTH"
    }
}
```

```
$ USAGEID=u9dvxc
$ KEYID=a56g1r3mzc
$ aws apigateway create-usage-plan-key --usage-plan-id $USAGEID --key-type
"API_KEY" --key-id $KEYID

Response: {
    "id": "cn02s869kj",
    "type": "API_KEY",
    "name": "PI-API-Key"
}
```

Verify-API

Upload code to S3 Bucket

```
$ aws s3 cp verifyAPI.zip s3://raspberry-pi-wsu-senior
```

Create new policy

Make sure your in the policies directory in the deployment_package

```
$ aws iam create-policy --policy-name verifyapi-policy --policy-document
file://VerifyAPI.json
```

```
Response: {
    "Policy": {
        "PolicyName": "verifyapi-policy",
        "PolicyId": "ANPAJCETVWRAMPUBBXMXW",
        "Arn": "arn:aws:iam::667671192261:policy/verifyapi-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2018-12-06T02:50:59Z",
        "UpdateDate": "2018-12-06T02:50:59Z"
    }
}
```

From the response, copy the ARN for later usage
For example:
Arn: “arn:aws:iam::667671192261:policy/sensordata-policy”

Create IAM Role

```
$ aws iam create-role --role-name WSU-VERIFY-ROLE
--assume-role-policy-document file://trusty.json

---------------------------------------------------------------------------
Response: {
    "Role": {
        "Path": "/",
        "RoleName": "WSU-VERIFY-ROLE",
        "RoleId": "AROAJBKSEHFMYO5NVKGBC",
        "Arn": "arn:aws:iam::667671192261:role/WSU-VERIFY-ROLE",
        "CreateDate": "2018-12-06T02:51:32Z",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "lambda.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }
    }
}
```

Copy the **ARN** and **Role** name for later usage

For example:
"RoleName": "WSU-SENSOR-ROLE"
"Arn": "arn:aws:iam::667671192261:role/WSU-SENSOR-ROLE"

Note: Make sure to replace the red highlighted portion with the appropriate arn name from the previous steps.

Attach the policy to the role

```
$ aws iam attach-role-policy --role-name WSU-VERIFY-ROLE --policy-arn
arn:aws:iam::667671192261:policy/verifyapi-policy
```

Get VPC info

Make note of the VpcId from the response

```
$ aws ec2 describe-vpcs --filters Name=isDefault,Values=true

Response: {
    "Vpcs": [
        {
            "CidrBlock": "172.31.0.0/16",
            "DhcpOptionsId": "dopt-3ed09845",
            "State": "available",
            "VpcId": "vpc-0310bf79",
            "OwnerId": "667671192261",
            "InstanceTenancy": "default",
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-a8fe45c4",
                    "CidrBlock": "172.31.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ],
            "IsDefault": true
        }
    ]
}
```

Make note of all of the SubnetId from the response. You’ll use it later. They’re highlighted with red font followed by “SubnetId” parameter

```
$ VPC=vpc-0310bf79
$ aws ec2 describe-subnets --filters Name=vpc-id,Values=$VPC
Name=default-for-az,Values=true

Response: {
    "Subnets": [
        {
            "AvailabilityZone": "us-east-1f",
            "AvailabilityZoneId": "use1-az5",
            "AvailableIpAddressCount": 4088,
            "CidrBlock": "172.31.48.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": true,
            "State": "available",
            "SubnetId": "subnet-b44713bb",
            "VpcId": "vpc-0310bf79",
            "OwnerId": "667671192261",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn":
"arn:aws:ec2:us-east-1:667671192261:subnet/subnet-b44713bb"
        },
        {
            "AvailabilityZone": "us-east-1c",
            "AvailabilityZoneId": "use1-az6",
            "AvailableIpAddressCount": 4088,
            "CidrBlock": "172.31.32.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": true,
            "State": "available",
            "SubnetId": "subnet-958b14c9",
            "VpcId": "vpc-0310bf79",
            "OwnerId": "667671192261",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn":
"arn:aws:ec2:us-east-1:667671192261:subnet/subnet-958b14c9"
        },
        {
            "AvailabilityZone": "us-east-1e",
            "AvailabilityZoneId": "use1-az3",
            "AvailableIpAddressCount": 4091,
            "CidrBlock": "172.31.64.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": true,
            "State": "available",
            "SubnetId": "subnet-7872d446",
            "VpcId": "vpc-0310bf79",
            "OwnerId": "667671192261",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn":
"arn:aws:ec2:us-east-1:667671192261:subnet/subnet-7872d446"
        },
        {
            "AvailabilityZone": "us-east-1d",
            "AvailabilityZoneId": "use1-az1",
            "AvailableIpAddressCount": 4091,
            "CidrBlock": "172.31.0.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": true,
            "State": "available",
            "SubnetId": "subnet-ef980a88",
            "VpcId": "vpc-0310bf79",
            "OwnerId": "667671192261",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn":
"arn:aws:ec2:us-east-1:667671192261:subnet/subnet-ef980a88"
        },
        {
            "AvailabilityZone": "us-east-1a",
            "AvailabilityZoneId": "use1-az2",
            "AvailableIpAddressCount": 4090,
            "CidrBlock": "172.31.80.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": true,
            "State": "available",
            "SubnetId": "subnet-078f1629",
            "VpcId": "vpc-0310bf79",
            "OwnerId": "667671192261",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn":
"arn:aws:ec2:us-east-1:667671192261:subnet/subnet-078f1629"
        },
        {
            "AvailabilityZone": "us-east-1b",
            "AvailabilityZoneId": "use1-az4",
            "AvailableIpAddressCount": 4091,
            "CidrBlock": "172.31.16.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": true,
            "State": "available",
            "SubnetId": "subnet-881c3ec2",
            "VpcId": "vpc-0310bf79",
            "OwnerId": "667671192261",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn":
"arn:aws:ec2:us-east-1:667671192261:subnet/subnet-881c3ec2"
        }
    ]
}
```

Get the Security Group ID

Make note of the “GroupID” parameter’s value.

```
$ aws ec2 describe-security-groups --filters Name=vpc-id,Values=$VPC
--group-names default

Response: {
  "SecurityGroups": [
    {
      "Description": "default VPC security group",
      "GroupName": "default",
      "IpPermissions": [
        {
          "FromPort": 3306,
          "IpProtocol": "tcp",
          "IpRanges": [
            {
              "CidrIp": "172.31.42.214/32",
              "Description": "ec2-for-AMI"
            },
            {
              "CidrIp": "68.01.01.123/32",
              "Description": "joy home"
            }
          ],
          "Ipv6Ranges": [],
          "PrefixListIds": [],
          "ToPort": 3306,
          "UserIdGroupPairs": [
            {
              "Description": "ASG Security group",
              "GroupId": "sg-0a0876f20bed41e27",
              "UserId": "667671192261"
            },
            {
              "Description": "vpc security",
              "GroupId": "sg-2554ce65",
              "UserId": "667671192261"
            }
          ]
        }
      ],
      "OwnerId": "667671192261",
      "GroupId": "sg-2554ce65",
      "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "Ipv6Ranges": [],
                    "PrefixListIds": [],
                    "UserIdGroupPairs": []
                }
            ],
            "VpcId": "vpc-0310bf79"
        }
    ]
}
```

Create new Lambda function

```
$ aws lambda create-function --function-name verifyapi --runtime python3.6
--role arn:aws:iam::667671192261:role/WSU-VERIFY-ROLE --handler app.handler
--code S3Bucket=raspberry-pi-wsu-senior,S3Key=verifyAPI.zip --vpc-config
SubnetIds=subnet-b44713bb,subnet-958b14c9,subnet-ef980a88,subnet-078f1629,s
ubnet-881c3ec2,SecurityGroupIds=sg-2554ce65

Response: {
    "FunctionName": "verifyapi",
    "FunctionArn":
"arn:aws:lambda:us-east-1:667671192261:function:verifyapi",
    "Runtime": "python3.6",
    "Role": "arn:aws:iam::667671192261:role/WSU-VERIFY-ROLE",
    "Handler": "app.handler",
    "CodeSize": 1853710,
    "Description": "",
    "Timeout": 3,
    "MemorySize": 128,
    "LastModified": "2018-12-06T02:57:39.875+0000",
    "CodeSha256": "zACPXrpoUvAG1wilb9xc7+oH9+IRcoD+xtoc4S+zSR4=",
    "Version": "$LATEST",
    "VpcConfig": {
        "SubnetIds": [
            "subnet-958b14c9",
            "subnet-ef980a88",
            "subnet-078f1629",
            "subnet-881c3ec2",
            "subnet-b44713bb"
        ],
        "SecurityGroupIds": [
            "sg-2554ce65"
        ],
        "VpcId": "vpc-0310bf79"
    },
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "RevisionId": "e34e5b2c-d6c6-4d22-8c60-f944c6f15c40"
}
```

Setup API Gateway

Create the API

```
$ aws apigateway create-rest-api --name verifyapi-API
--endpoint-configuration types=REGIONAL

---------------------------------------------------------------------------
Response: {
    "id": "m97wiv8zwh",
    "name": "verifyapi-API",
    "createdDate": 1544065128,
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "REGIONAL"
        ]
    }
}
```

From the previous step, save the “id” parameter and replace it below with API value. Now, you can get the API root resource ID as well.

```
$ API=m97wiv8zwh
$ aws apigateway get-resources --rest-api-id $API

Response: {
    "items": [
        {
            "id": "b0gf9enoh5",
            "path": "/"
        }
    ]
}
```

Copy the id from the response into the clipboard.

Create a Resource in the API

Set the variable ROOT_RESOURCE to the id that you copied from previous step.

```
$ ROOT_RESOURCE=b0gf9enoh5
$ aws apigateway create-resource --rest-api-id $API --path-part
'sensordata' --parent-id $ROOT_RESOURCE

Response: {
    "id": "148fw7",
    "parentId": "b0gf9enoh5",
    "pathPart": "verifyapi",
    "path": "/verifyapi"
}
```

Create Method on the Resource

```
$ RESOURCE=148fw7
$ aws apigateway put-method --rest-api-id $API --resource-id $RESOURCE
--http-method POST --authorization-type "NONE" --api-key-required
```

```
Response: {
    "httpMethod": "POST",
    "authorizationType": "NONE",
    "apiKeyRequired": true
}
```

```
$ aws apigateway put-integration --rest-api-id $API --resource-id $RESOURCE
--http-method POST --type AWS_PROXY --integration-http-method POST --uri
arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambd
a:us-east-1:667671192261:function:verifyapi/invocations

Response: {
    "type": "AWS_PROXY",
    "httpMethod": "POST",
    "uri":
"arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lamb
da:us-east-1:667671192261:function:verifyapi/invocations",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "cacheNamespace": "148fw7",
    "cacheKeyParameters": []
}
```

```
$ aws apigateway put-method-response --rest-api-id $API --resource-id $RESOURCE
--http-method POST --status-code 200 --response-parameters
"method.response.header.Access-Control-Allow-Origin=true" --response-models
application/json=Empty

Response: {
    "statusCode": "200",
    "responseParameters": {
        "method.response.header.Access-Control-Allow-Origin": true
    },
    "responseModels": {
        "application/json": "Empty"
    }
}
```

```
$ aws apigateway put-integration-response --rest-api-id $API --resource-id
$RESOURCE --http-method POST --status-code 200 --selection-pattern ""
--response-templates '{"application/json": "{\"json\": \"Empty\"}"}'
--response-parameters
'{"method.response.header.Access-Control-Allow-Origin":"'"'*'"'"}'

Response: {
    "statusCode": "200",
    "selectionPattern": "",
    "responseParameters": {
        "method.response.header.Access-Control-Allow-Origin": "'*'"
    },
    "responseTemplates": {
        "application/json": "{\"json\": \"Empty\"}"
    }
}
```

```
$ aws apigateway create-deployment --rest-api-id $API --stage-name final
--stage-description 'Final Stage' --description 'Deployment for the weather
station pi'

Response: {
    "id": "zhni8q",
    "description": "Deployment for the weather station pi",
    "createdDate": 1543986142
}
```

```
$ aws lambda add-permission --function-name verifyapi --statement-id
apigateway-testt-1 --action lambda:InvokeFunction --principal
apigateway.amazonaws.com --source-arn
"arn:aws:execute-api:us-east-1:667671192261:8h35no6zsg/*/POST/verifyapi"


Response:{
    "Statement":
"{\"Sid\":\"apigateway-testt-1\",\"Effect\":\"Allow\",\"Principal\":{\"Serv
ice\":\"apigateway.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction\",\"
Resource\":\"arn:aws:lambda:us-east-1:667671192261:function:verifyapi\",\"C
ondition\":{\"ArnLike\":{\"AWS:SourceArn\":\"arn:aws:execute-api:us-east-1:
667671192261:8h35no6zsg/*/POST/verifyapi\"}}}"
}
```

```
$ aws apigateway create-api-key --name 'PI-VERIFY-Key' --description 'for
development' --enabled --stage-keys restApiId=$API,stageName=final

Response: {
    "id": "7n2exwj8s5",
    "value": "3wjsITLRVM1iDuPCxCKWc47gWFbTqnqz4JM44NAa",
    "name": "PI-Verify-Key",
    "description": "for development",
    "enabled": true,
    "createdDate": 1543986612,
    "lastUpdatedDate": 1543986612,
    "stageKeys": [
        "m97wiv8zwh/final"
    ]
}
```

```
$ aws apigateway create-usage-plan --name "pi-api-key-usagePlan"
--api-stages apiId=$API,stage=final --description "usage plan for pi api
key" --throttle burstLimit=10,rateLimit=150 --quota
limit=500,offset=0,period=MONTH

Response:{
    "id": "mhujyv",
    "name": "pi-api-key-usagePlan",
    "description": "usage plan for pi api key",
    "apiStages": [
        {
            "apiId": "m97wiv8zwh",
            "stage": "final"
        }
    ],
    "throttle": {
        "burstLimit": 10,
        "rateLimit": 150.0
    },
    "quota": {
        "limit": 500,
        "offset": 0,
        "period": "MONTH"
    }
}
```

```
$ USAGEID=mhujyv
$ KEYID=7n2exwj8s5
$ aws apigateway create-usage-plan-key --usage-plan-id $USAGEID --key-type
"API_KEY" --key-id $KEYID

Response: {
    "id": "7n2exwj8s5j",
    "type": "API_KEY",
    "name": "PI-Verify-Key"
}
```

Elastic Load Balancer

Setup S3 Bucket
Setup Lambda
Create IAM Role
Get the other stuffs

AWS RDS - MySQL

As of today December 5th, 2018, AWS RDS only allows dbs encryption on t2.small (lowest cost). No free tier option available for encryption. So, this will cost you money. Check AWS cost usage for exact price (it may change time to time).

Prerequisite: SensorData, Verify-API sections are completed

```
$ aws s3 cp sensordata s3://raspberry-pi-wsu-senior
```

Create new policy

Make sure your in the policies directory in the deployment_package

```
$ aws iam create-policy --policy-name RDS-Policy --policy-document
file://RDS.json

Response: {
    "Policy": {
        "PolicyName": "RDS-Policy",
        "PolicyId": "ANPAIC24NBSCG6A3T5AS6",
        "Arn": "arn:aws:iam::667671192261:policy/RDS-Policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2018-12-05T17:05:43Z",
        "UpdateDate": "2018-12-05T17:05:43Z"
    }
}
```

From the response, copy the ARN for later usage
For example:
Arn: “arn:aws:iam::667671192261:policy/RDS-policy”

Create IAM Role

```
$ aws iam create-role --role-name WSU-RDS-ROLE
--assume-role-policy-document file://trusty.json


Response: {
    "Role": {
        "Path": "/",
        "RoleName": "WSU-RDS-ROLE",
        "RoleId": "AROAIFLRPDAESBHCFKORS",
        "Arn": "arn:aws:iam::667671192261:role/WSU-RDS-ROLE",
        "CreateDate": "2018-12-05T17:09:13Z",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "lambda.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }
    }
}
```

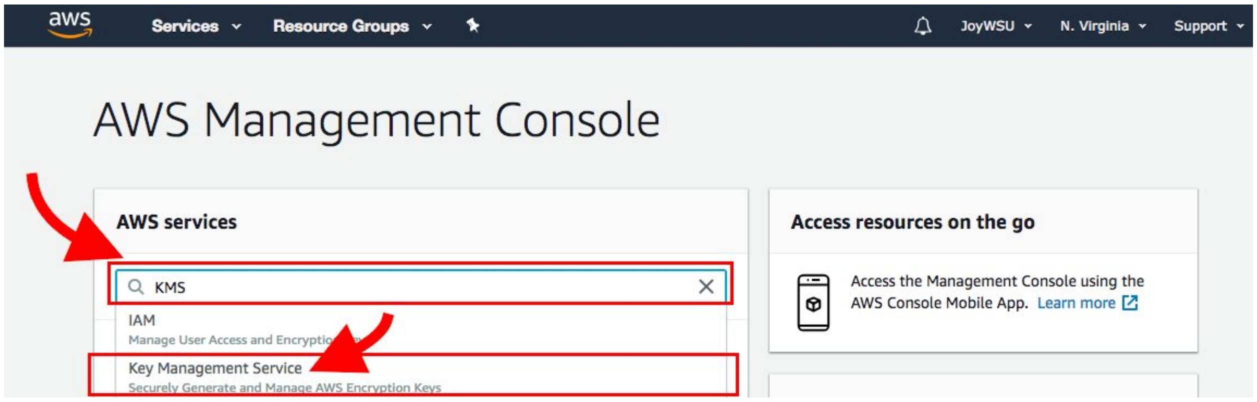
Note: Make sure to replace the red highlighted portion with the appropriate arn name from the previous steps.

Attach the policy to the role

```
$ aws iam attach-role-policy --role-name WSU-RDS-ROLE --policy-arn
arn:aws:iam::667671192261:policy/RDS-policy
```

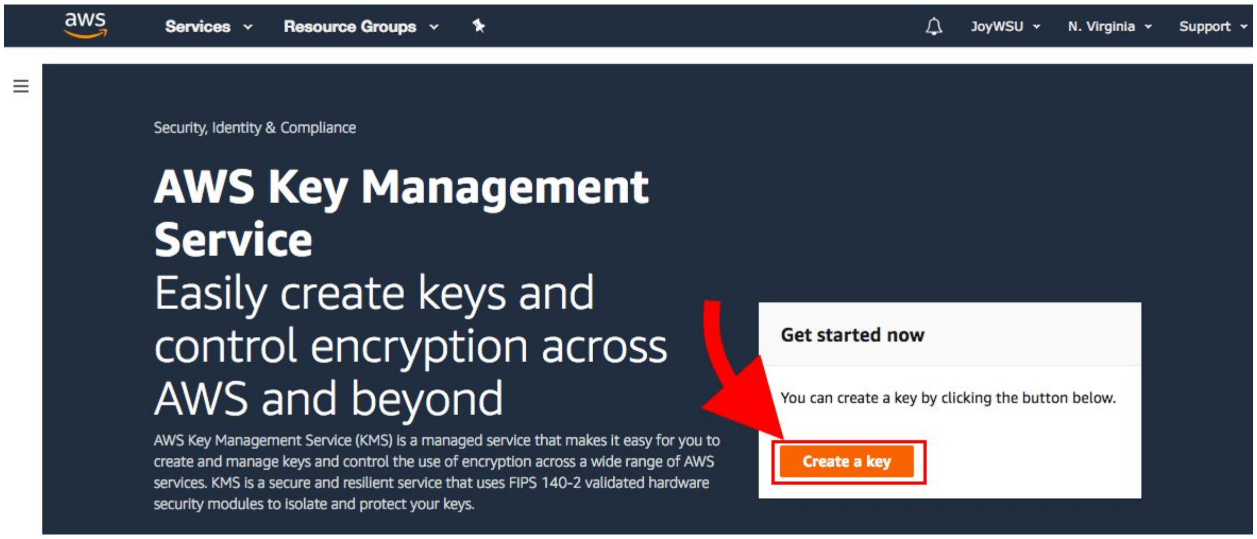
Create KMS Key

Search for ‘KMS’



Access the Management Console using the AWS Console Mobile App.

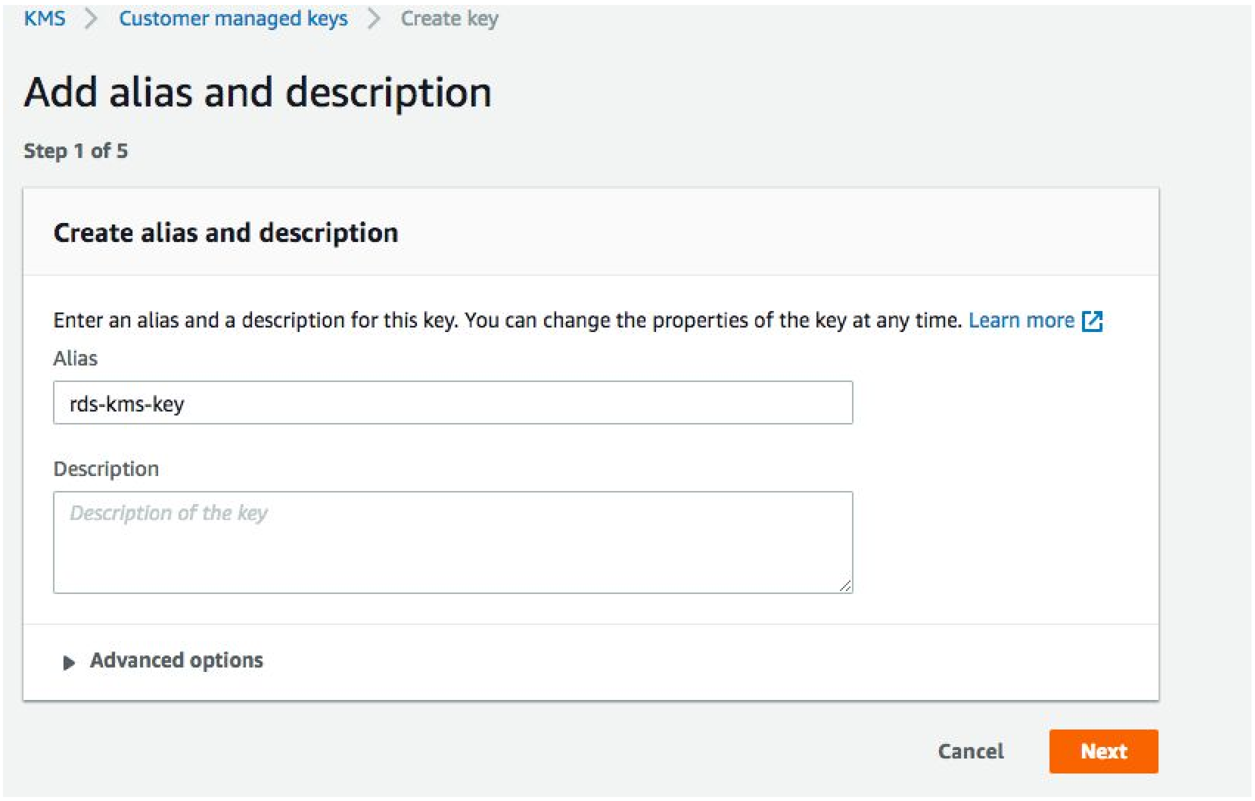
Click on Create a key


aws
Services
Resource Groups
JoyWSU
N. Virginia
Support
Security, Identity & Compliance
AWS Key Management Service
Easily create keys and control encryption across AWS and beyond
Get started now
Create a key

AWS Key Management Service (KMS) is a managed service that makes it easy for you to create and manage keys and control the use of encryption across a wide range of AWS services. KMS is a secure and resilient service that uses FIPS 140-2 validated hardware security modules to isolate and protect your keys.

You can create a key by clicking the button below.

Provide a Display name for the key: “rds-kms-key”



KMS > Customer managed keys > Create key

Add alias and description

Step 1 of 5

Create alias and description

Enter an alias and a description for this key. You can change the properties of the key at any time. Learn more

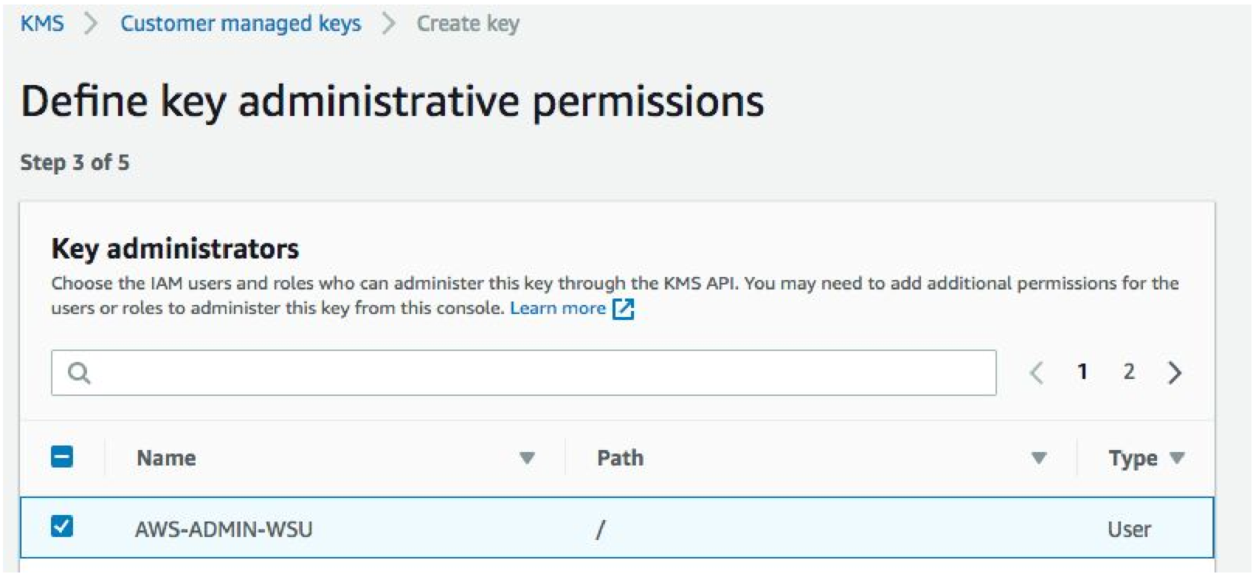
Alias: rds-kms-key

Description: Description of the key

▶ Advanced options

Click on Next.
Add Tags (not required), click on next.

Select ‘AWS-ADMIN-WSU’



KMS > Customer managed keys > Create key

Define key administrative permissions

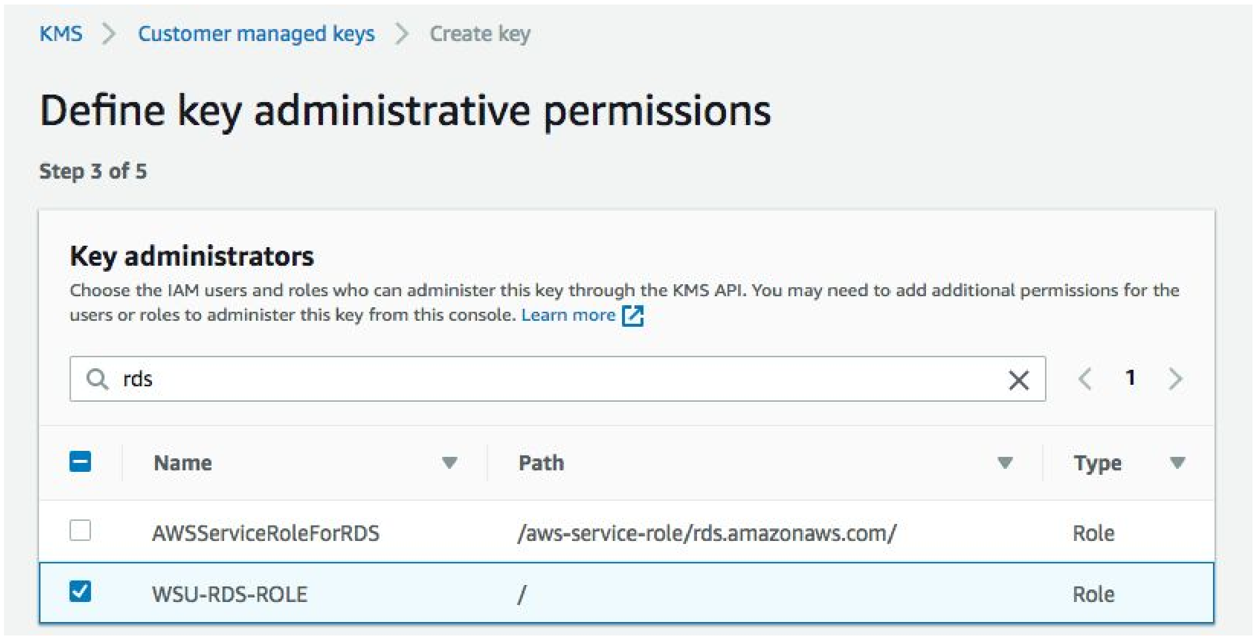
Step 3 of 5

Key administrators

Choose the IAM users and roles who can administer this key through the KMS API. You may need to add additional permissions for the users or roles to administer this key from this console. Learn more

	Name	Path	Type
✓	AWS-ADMIN-WSU	/	User

And search for “WSU-RDS-ROLE” and select it.



Define key administrative permissions

Step 3 of 5

Key administrators

Choose the IAM users and roles who can administer this key through the KMS API. You may need to add additional permissions for the users or roles to administer this key from this console. Learn more

rds

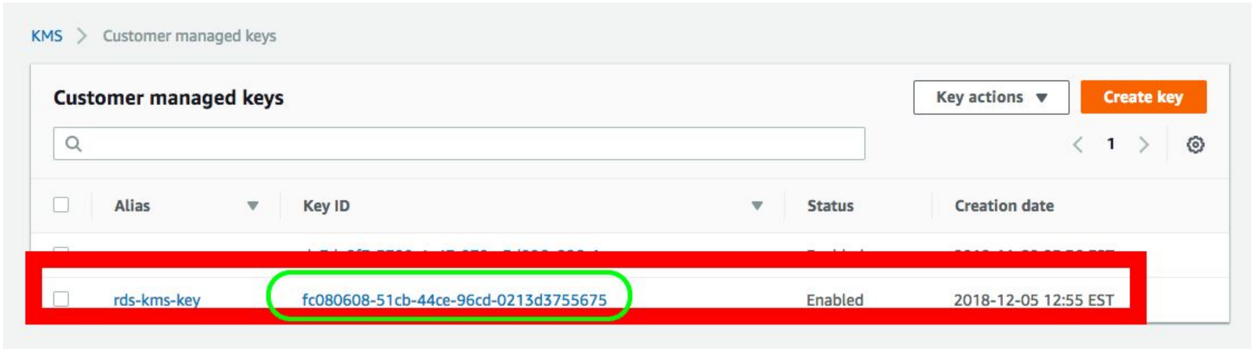
	Name	Path	Type
☐	AWSServiceRoleForRDS	/aws-service-role/rds.amazonaws.com/	Role
✓	WSU-RDS-ROLE	/	Role

Click on “Next”
Select the “AWS-ADMIN-WSU” and “WSU-RDS-ROLE” for Define key usage permissions as well. And click on “Next”

Review the key policy and click on “Finish”

```
{
    "Version": "2012-10-17",
    "Id": "key-consolepolicy-3",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::667671192261:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allow access for Key Administrators",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::667671192261:user/AWS-ADMIN-WSU",
                    "arn:aws:iam::667671192261:role/WSU-RDS-ROLE"
                ]
            },
            "Action": [
                "kms:Create*",
                "kms:Describe*",
                "kms:Enable*",
                "kms:List*",
                "kms:Put*",
                "kms:Update*",
                "kms:Revoke*",
                "kms:Disable*",
                "kms:Get*",
                "kms:Delete*",
                "kms:TagResource",
                "kms:UntagResource",
                "kms:ScheduleKeyDeletion",
                "kms:CancelKeyDeletion"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::667671192261:role/WSU-RDS-ROLE"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::667671192261:role/WSU-RDS-ROLE"
            },
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        }
    ]
}
```

Now copy the Key ID into the clipboard

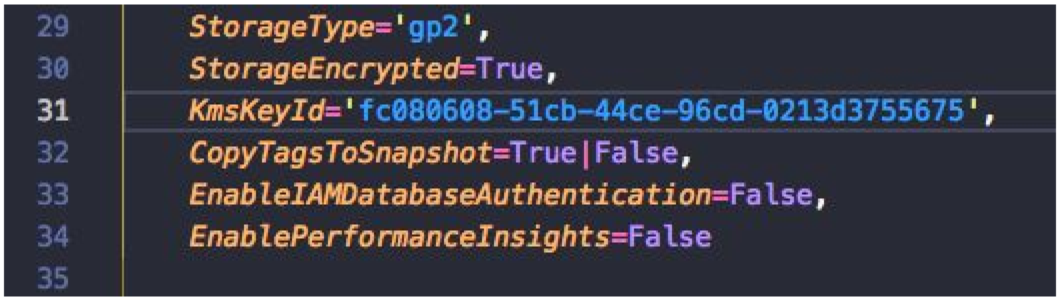


Customer managed keys

Alias	Key ID	Status	Creation date
rds-kms-key	fc080608-51cb-44ce-96cd-0213d3755675	Enabled	2018-12-05 12:55 EST

In my case, it is `fc080608-51cb-44ce-96cd-0213d3755675`

Now open the file in deployment_package called “rds_function.py” and replace the KmsKeyId:



```
    StorageType='gp2',
    StorageEncrypted=True,
    KmsKeyId='fc080608-51cb-44ce-96cd-0213d3755675',
    CopyTagsToSnapshot=True|False,
    EnableIAMDatabaseAuthentication=False,
    EnablePerformanceInsights=False
```

Now, compress/zip the same file “rds_function.py”. After compress/zip, name the file “aws_rds.zip”

Create new Lambda function

```
$ aws lambda create-function --function-name AWSRDS --runtime python3.6
--timeout 200 --role arn:aws:iam::667671192261:role/WSU-RDS-ROLE --handler
rds_function.lambda_handler --code
S3Bucket=raspberry-pi-wsu-senior,S3Key=aws_rds.zip

 Response: {
    "FunctionName": "AWSRDS",
    "FunctionArn": "arn:aws:lambda:us-east-1:667671192261:function:AWSRDS",
    "Runtime": "python3.6",
    "Role": "arn:aws:iam::667671192261:role/WSU-RDS-ROLE",
    "Handler": "rds_function.lambda_handler",
    "CodeSize": 700,
    "Description": "",
    "Timeout": 200,
    "MemorySize": 128,
    "LastModified": "2018-12-05T17:18:04.766+0000",
    "CodeSha256": "kblpiu2FFl3eCYdSaDquQb05MfcVW3D4IiDsf+0ywys=",
    "Version": "$LATEST",
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "RevisionId": "731a6cad-99f6-45fa-bc95-8786ec3309d7"
}
```