

Infrastructure as code (IaC) is the process of managing and provisioning IT resources through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools. This includes bare-metal servers as well as VMs and the associated configuration resources.

In any given enterprise which uses IaC for VM deployment and management, the process consists of the following 2 steps:

- 1) Create an OS image pre-configured with the required software and applications
- 2) Use the IaC system to create VMs based on the configured OS and with the required permissions

In this DIY exercise, we will be using custom Python code in an AWS Lambda function to perform the second step, i.e. using IaC to deploy VMs.

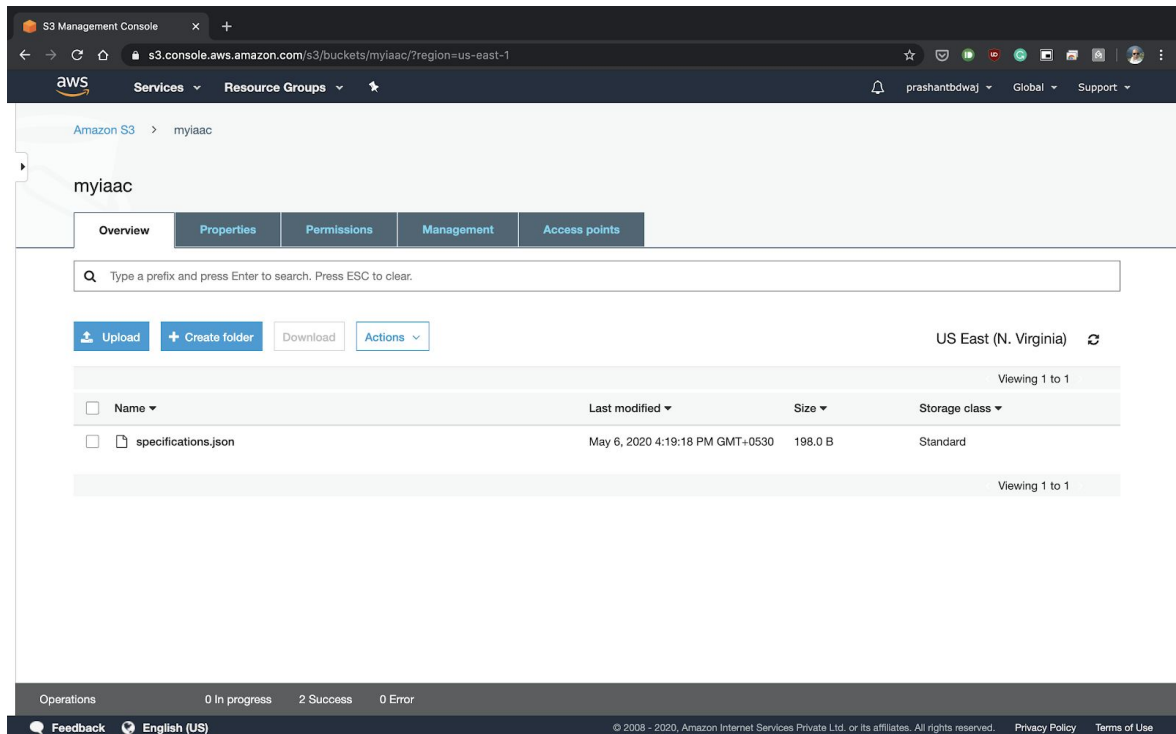
#### Requirements:

- 1) AWS account: This exercise should not be performed on an Educate account as some features may not be available
- 2) Configuration files:
  - a) Lambda program: *myIaaC.py*
  - b) JSON file: *specification.json*
  - c) Test event JSON file: *test.json*

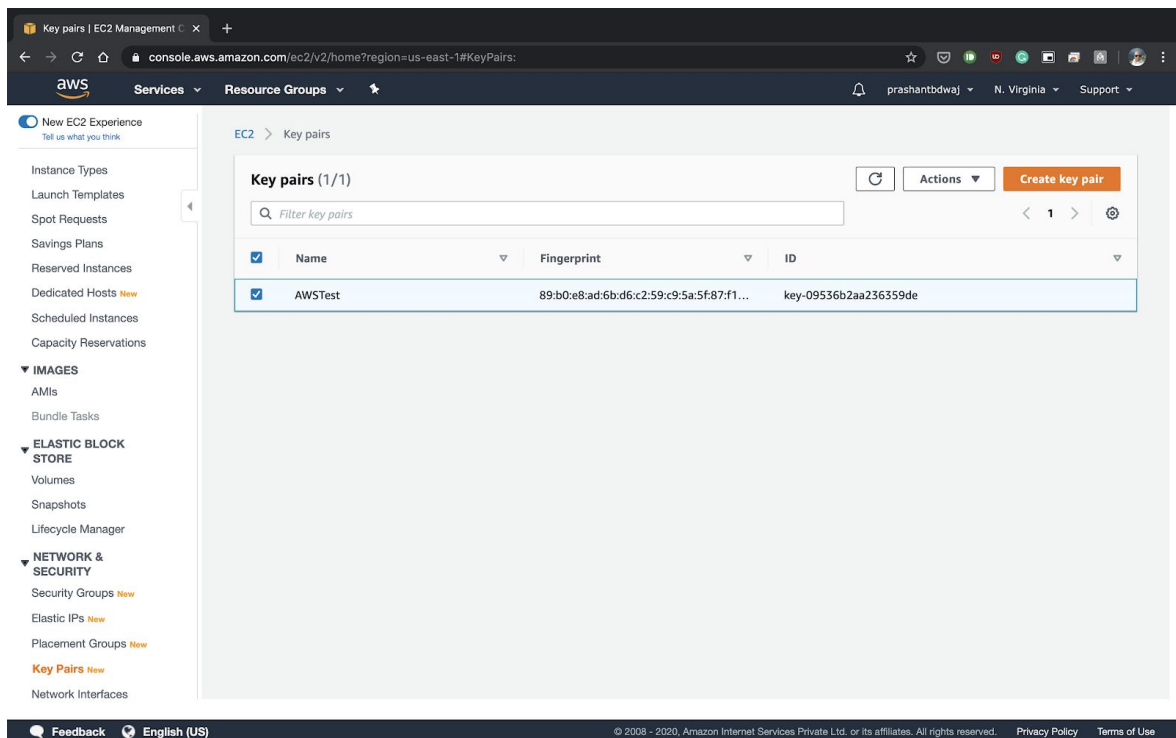
#### Steps to be taken:

- 1) Ensure that the selected region is N.Virginia (us-east-1)
- 2) Navigate to S3 and create a bucket called *myIaaC-[SOMENUMBERS]*
  - a) Eg myIaaC-55898412 to make the bucket name unique

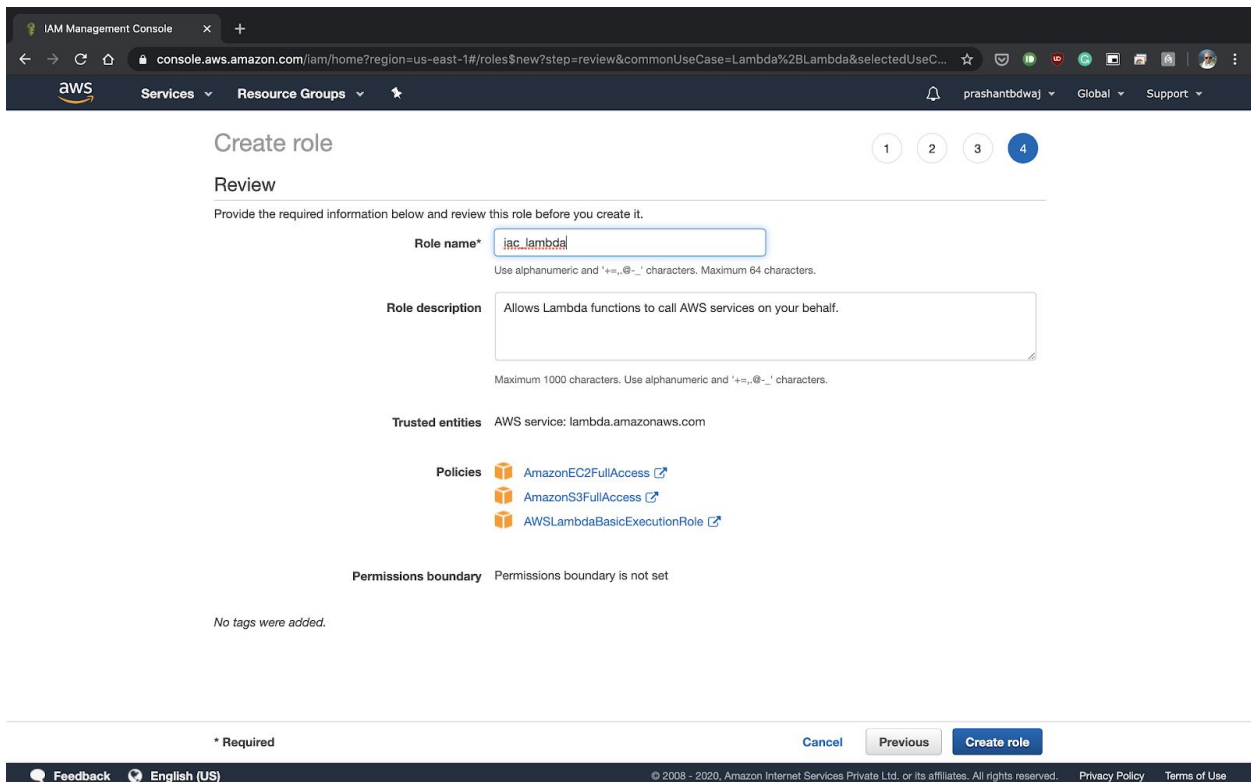
3) Upload the provided specification.json file to the bucket



4) Navigate to EC2, then go to Key Pairs and create a new keypair called **AWSTest**.



- 5) Verify that there is no security group called `serversg` by navigating to Security groups under EC2. If there is, then delete it.
- 6) Navigate to IAM->Roles
- 7) Create a new role called `iac_lambda` with AWS Lambda as the use case and attach the following policies to it
  - a) AWSLambdaBasicExecutionRole
  - b) AmazonS3FullAccess
  - c) AmazonEC2FullAccess



The screenshot shows the AWS IAM Management Console 'Create role' page, specifically the 'Review' step. The page title is 'Create role' with a progress indicator showing steps 1, 2, 3, and 4, where 4 is the current step. The 'Review' section prompts the user to 'Provide the required information below and review this role before you create it.' The form fields are as follows:

- Role name\***: `iac_lambda`. A note below states: 'Use alphanumeric and '+=, @, \_' characters. Maximum 64 characters.'
- Role description**: 'Allows Lambda functions to call AWS services on your behalf.' A note below states: 'Maximum 1000 characters. Use alphanumeric and '+=, @, \_' characters.'
- Trusted entities**: 'AWS service: lambda.amazonaws.com'
- Policies**: Three policies are listed with checkboxes: `AmazonEC2FullAccess`, `AmazonS3FullAccess`, and `AWSLambdaBasicExecutionRole`.
- Permissions boundary**: 'Permissions boundary is not set'

At the bottom, it says 'No tags were added.' The footer includes a '\* Required' label, 'Cancel', 'Previous', and 'Create role' buttons. The bottom-most bar contains 'Feedback', 'English (US)', and copyright information: '© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use'.

- 8) Navigate to AWS Lambda
- 9) Create a Lambda function from scratch with the name `myiac` and runtime set to Python 3.8
- 10) Select the above-created role `iac_lambda` as an execution role by selecting "Use an existing role" and click on Create

**Basic information**

**Function name**  
Enter a name that describes the purpose of your function.  
myiac  
Use only letters, numbers, hyphens, or underscores with no spaces.

**Runtime** [Info](#)  
Choose the language to use to write your function.  
Python 3.8

**Permissions** [Info](#)  
Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

▼ **Choose or create an execution role**

**Execution role**  
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☐ Create a new role with basic Lambda permissions

☒ Use an existing role

☐ Create a new role from AWS policy templates

**Existing role**  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.  
iaac\_lambda  
[View the iaac\\_lambda role](#) on the IAM console.

Cancel **Create function**

11) Under Configuration, select Add Trigger and enter the following details

- a) Trigger: S3
- b) Bucket name: myiaac
- c) Event type : PUT
- d) Uncheck "Enable now"

Rest of the fields need not be filled

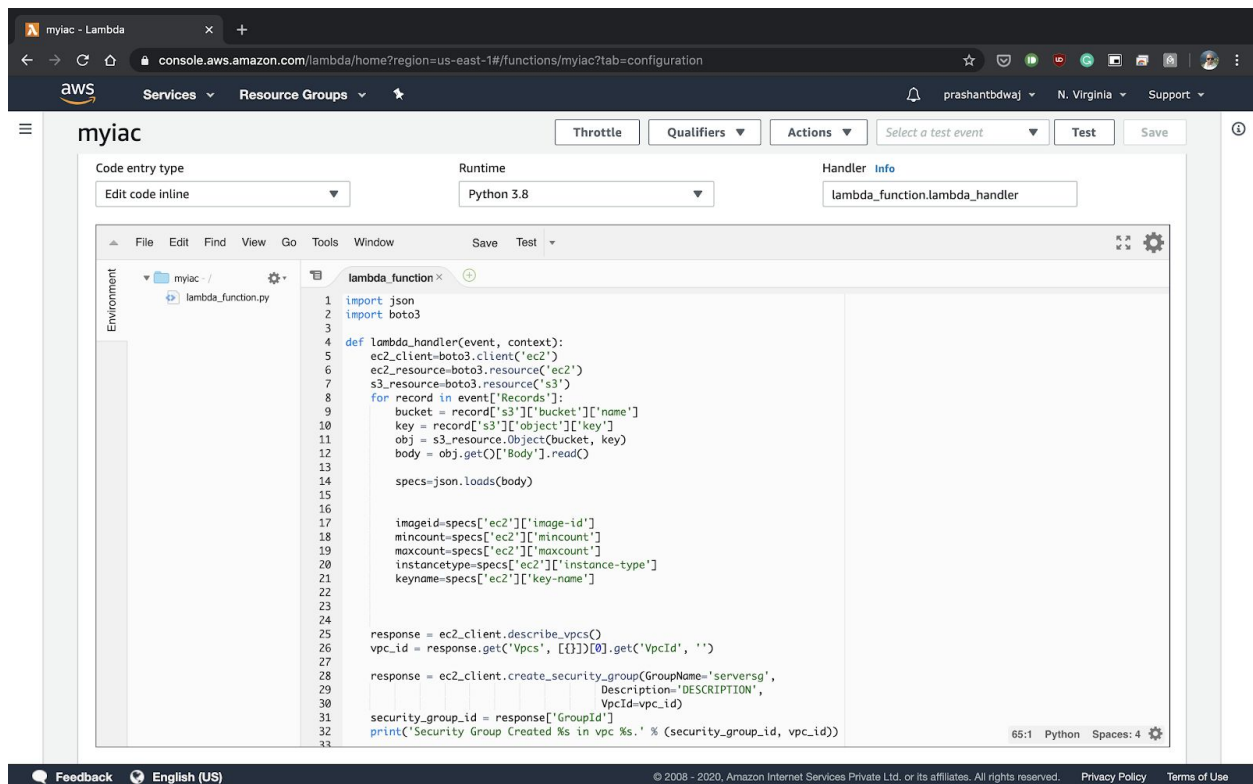
The screenshot shows the AWS Lambda console's 'Trigger configuration' page. At the top, there's a navigation bar with the AWS logo, 'Services', 'Resource Groups', and a user profile 'prashantbdwaj' in 'N. Virginia'. The main content area is titled 'Trigger configuration' and contains the following sections:

- S3**: A dropdown menu showing 'S3' with 'aws storage' below it.
- Bucket**: A text input field containing 'myiaac' and a refresh icon.
- Event type**: A dropdown menu showing 'PUT'.
- Prefix - optional**: A text input field with the placeholder 'e.g. images/'.
- Suffix - optional**: A text input field with the placeholder 'e.g. .jpg'.
- Permissions**: A note stating 'Lambda will add the necessary permissions for Amazon S3 to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.'
- Enable trigger**: A checkbox that is currently unchecked, with the text 'Enable the trigger now, or create it in a disabled state for testing (recommended).'

At the bottom of the configuration panel, there are 'Cancel' and 'Add' buttons. The footer of the console shows 'Feedback', 'English (US)', and copyright information for 2008-2020.

12) Under Configuration, scroll down to Basic Settings and set the timeout to 2 minutes

13) Under Function Code, ensure that the runtime is set to Python 3.8 and replace the text with the code given in the file mylaaC.py and press Save on the top right corner



- 14) On the top right corner, select “Select a test event” and select then “Configure test events”.
- 15) Under event template, select s3-put, and enter the event name as “laCtestevent”
- 16) Replace the test event template with the code in the file test.json and click on Create

The screenshot displays the AWS Lambda console for a function named 'myiac'. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and a user profile for 'prashantbdwaj' in 'N. Virginia' region. The function configuration shows it is a 'Python 3.8' function with the handler 'lambda\_function.lambda\_handler'. The code editor displays the following Python code:

```

55     KeyName = keyname)
56
57     print ('Instance created %s' % instance)
58
59
60
61     return {
62         'statusCode': 200,
63         'body': json.dumps("Execution Successful")
64     }
65

```

Below the code editor, the 'Execution Result' tab is selected, showing the following details:

- Status:** Succeeded
- Max Memory Used:** 88 MB
- Time:** 4401.48 ms

The **Response:** section shows:

```

{
  "statusCode": 200,
  "body": "\"Execution Successful\""
}

```

The **Request ID:** is '3caefce1-c5b9-440c-8c9c-552f79ffb19'.

The **Function Logs:** section shows the following log entries:

```

START RequestID: 3caefce1-c5b9-440c-8c9c-552f79ffb19 Version: $LATEST
Security Group Created sg-0ba27d6e093fa3cfd in vpc vpc-0983951e69015b521.
Ingress Successfully Set { 'ResponseMetadata': { 'RequestId': '8fde43a5-57f7-46f7-8e08-71a178e8433', 'HTTPStatusCode': 200, 'HTTPHeaders': {
Instance created [ec2.Instance(id='i-807fa24691b906e3'), ec2.Instance(id='i-0badae67d27d92bca'), ec2.Instance(id='i-0c20de5c35f92f8d0')]

```

The bottom of the console shows a 'Feedback' button and the language set to 'English (US)'. The footer contains copyright information for 2020 and links to 'Privacy Policy' and 'Terms of Use'.

18) Navigate to EC2 and verify that 3 new instances have been created and they all use the security group *serversg* which has ports open for SSH(22) and HTTP(80).

**Bonus Objectives:**

- 1) Open the file specifications.json and try the following tasks
  - a) Modify the AMI ID to launch an Amazon Linux 2 AMI instead of Ubuntu
  - b) Modify the value maxcount to launch 2 instead of 3 instances (Note: If you attempt to launch a large number of instances, say 10, the Lambda function may timeout. If this happens, simply increase the timeout value)
  - c) Try using a different key-pair. The keypair should be already created before running the Lambda functions
- 2) Trigger the Lambda function in realtime instead of through a test event