

Table of Contents

Table of Contents.....	1
Introduction.....	2
Prerequisites.....	2
Overview of the Setup Process.....	2
Step 1: Parameter Store Configuration.....	3
Step Recap.....	6
Step 2a: IAM Role Setup.....	7
Step 2b: IAM Role to retrieve secrets.....	13
Step Recap.....	19
Step 3: EC2 Instance Launch Configuration.....	20
Step Recap.....	25
Step 4: EC2 Instance installation.....	26
Step Recap.....	29
Step 5: Sample App Deployment.....	30
Step 5a: Transferring the Sample App files.....	30
3a. Transferring files from your laptop.....	30
3b. Downloading from github.....	32
Step 5b: Starting the Sample App.....	33
Step Recap.....	34
Additional Notes.....	35
To stop the container that is running the app, run the command below.....	35
Associate an elastic ip address to your EC2.....	35
Disassociate the elastic ip address to your EC2.....	35
To stop the EC2 instance.....	35
Delete parameters from parameter store.....	36

Introduction

This document provides a step-by-step guide to set up the Sample App on your AWS account. It assumes no prior configuration and is designed to help you get started from scratch. By following the steps, you will be able to have the Sample App on your AWS account and a link for people to access it.

Prerequisites

Before proceeding with the setup, ensure you have the following:

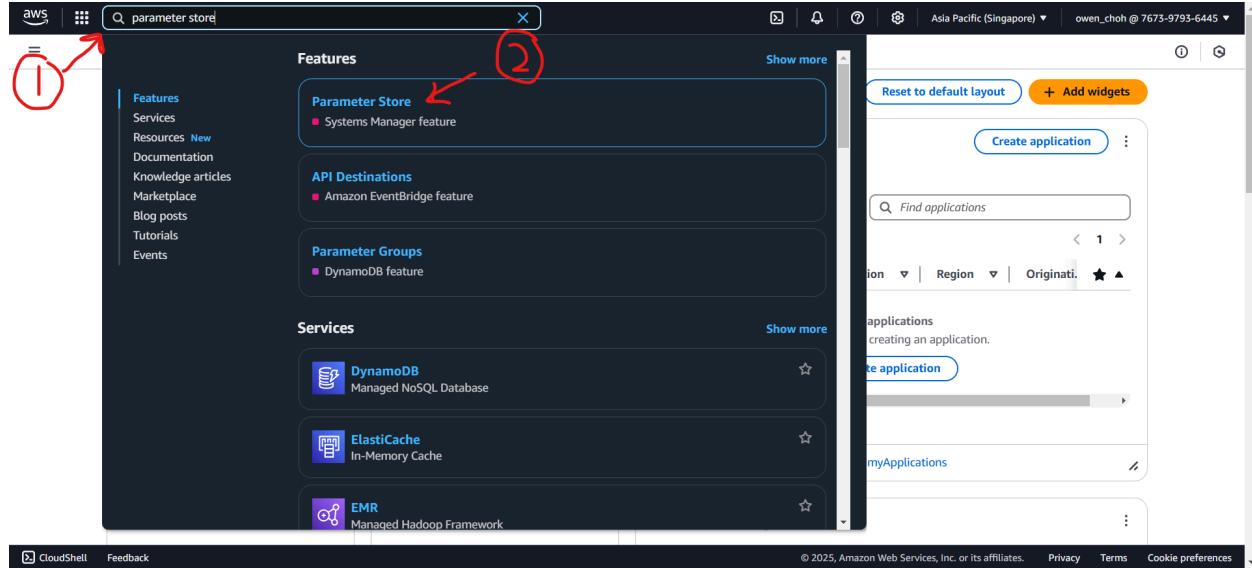
1. **AWS Account:**
An active AWS account with the necessary permissions to create and manage the following resources:
 - EC2 instances
 - Parameter Store parameters (via AWS Systems Manager)
 - IAM roles
2. **Sample App Source Code:**
The source code of the Sample App, including its dependencies, available either locally or in a repository.
 - In this guide, we will use the SSG GitHub repository (<https://github.com/ssg-wsg/Sample-Codes>) as an example.
3. **Sample App Secrets:**
The secrets used by the Sample App to make the API calls on behalf of our users. There are a total of 3 items:
 - Encryption key
 - Certificate
 - Private Key

Overview of the Setup Process

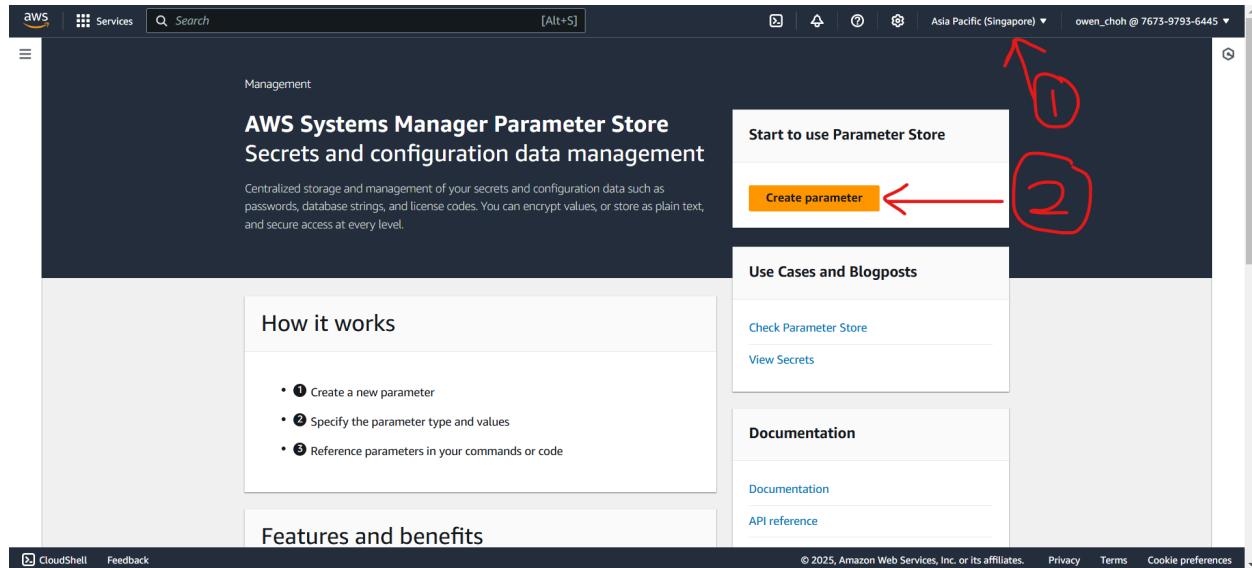
1. **Parameter Store Configuration:** Store the necessary secrets and configuration settings.
2. **IAM Role Setup:** Create an IAM role and attach policies for accessing the parameter store.
3. **EC2 Instance Launch Configuration:** Launch and configure an EC2 instance to host the Sample App.
4. **EC2 Instance installation:** Install dependencies on the EC2 instance.
5. **Sample App Deployment:** Deploy the app on the EC2 instance.
6. **Additional Notes:** Additional information after setting up the instance.

Step 1: Parameter Store Configuration

- From the AWS homepage, navigate to the AWS Parameter Store.



- On the **parameter store** page, make sure you are in the correct **region** and click **Create parameter**.



- For each secret you will need to put a Name and select “**Standard**” tier.

AWS Systems Manager > Parameter Store > Create parameter

Create parameter

Parameter details

Name When naming a parameter, you can use forward slashes (/) to organize it into a hierarchy. [Learn more about hierarchies](#)

Description — Optional

Tier

Parameter Store offers standard and advanced parameters.

Standard
Store up to 10,000 standard parameters. Store parameter values up to 4 KB. Parameter policies and sharing with other AWS accounts are not available. No additional charge.

Advanced
Store up to 100,000 advanced parameters. Store parameter values up to 8 KB. Add parameter policies. Share with other AWS accounts. Charges apply.

[Standard parameters cannot be shared with other AWS accounts. Learn more](#)

Type CloudShell Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- The names for each of the secret in this example will be as follows:
- Encryption key - name: **/SampleApp/test/secrets/encrypt**
- Certificate - name: **/SampleApp/test/secrets/cert**
- Private Key - name: **/SampleApp/test/secrets/key**

4. Scroll down and select the “**SecureString**” type for each of the secrets. The KMS key source will be the default option.

Standard parameters cannot be shared with other AWS accounts. [Learn more](#)

Type

String
Any string value.

StringList
Separate strings using commas.

SecureString
Encrypt sensitive data using KMS keys from your account or another account.

KMS Key source - [Learn more](#)

My current account
Use the default KMS key for this account or specify a customer-managed key for this account.

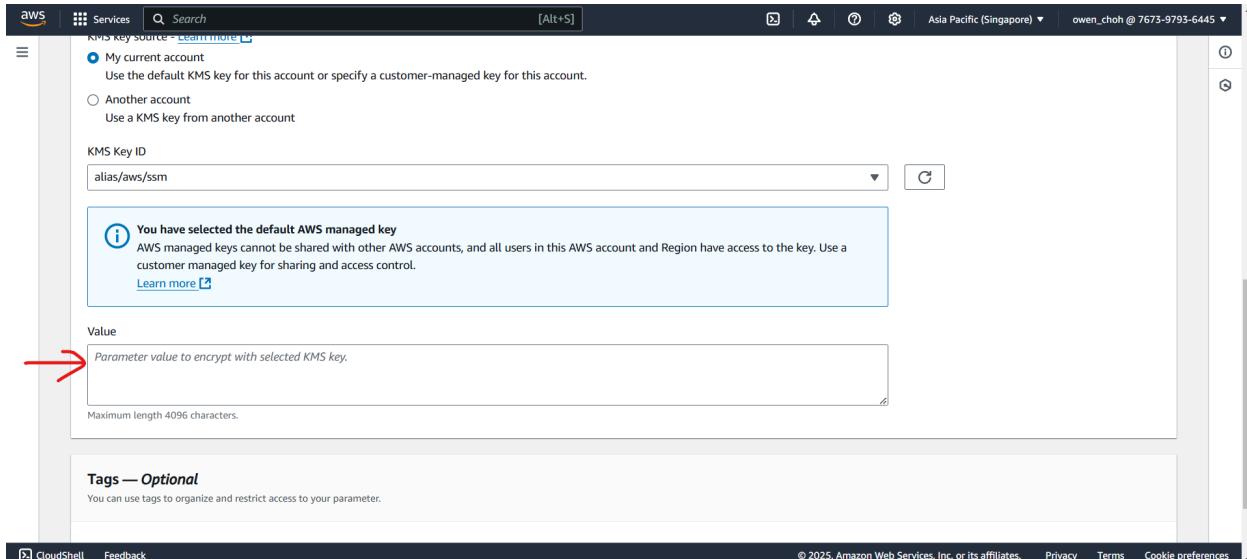
Another account
Use a KMS key from another account

KMS Key ID

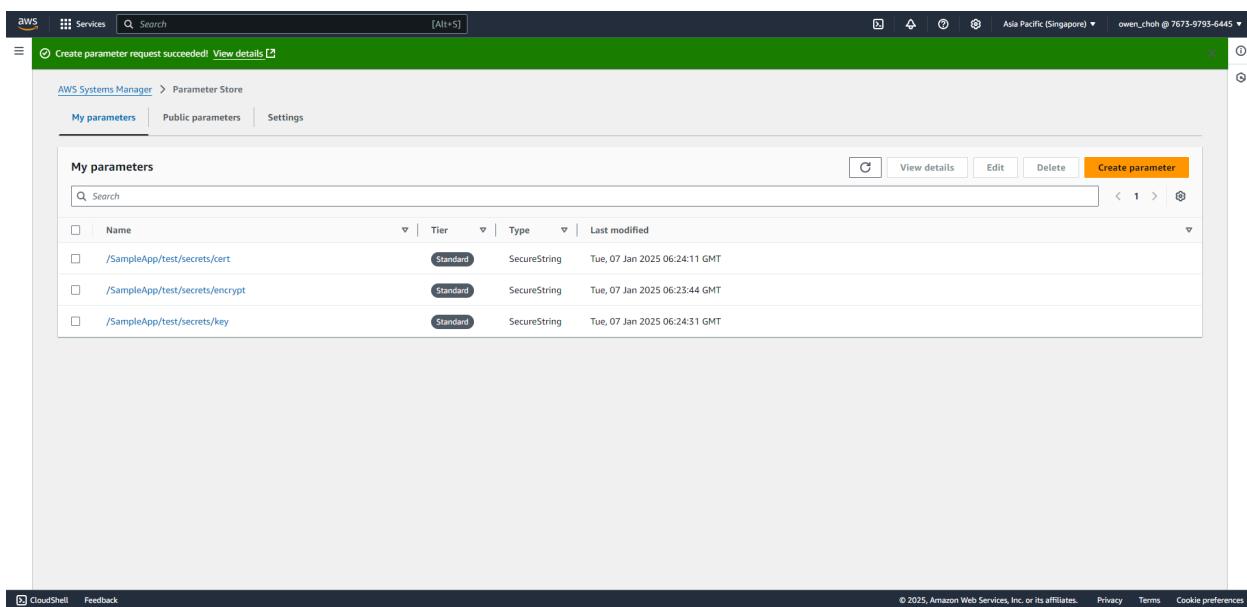
You have selected the default AWS managed key
AWS managed keys cannot be shared with other AWS accounts, and all users in this AWS account and Region have access to the key. Use a customer managed key for sharing and access control.
[Learn more](#)

CloudShell Feedback

5. Scroll down and paste the secret into the “**Value**” field.



6. Perform the previous steps to create parameters for all 3 secrets. This is the screen you should see after you created them.



7. Copy the arn of one of the secrets you created for reference in the later steps.
 - Example used in the steps will be
 - arn:aws:ssm:ap-southeast-1:767397936445:parameter/**SampleApp/test/secrets/encrypt**

The screenshot shows the AWS Systems Manager Parameter Store interface. The parameter details page for the path `/SampleApp/test/secrets/encrypt` is displayed. Key information includes:

- Name:** `/SampleApp/test/secrets/encrypt`
- ARN:** `arn:aws:ssm:ap-southeast-1:767397936445:parameter/SampleApp/test/secrets/encrypt`
- Tier:** Standard
- Type:** SecureString
- Value:** (Show decrypted value) `*****`
- Description:** for documentation
- Data type:** text
- Last modified user:** arn:aws:iam:767397936445:user/owen_choh
- Last modified date:** Tue, 07 Jan 2025 06:23:44 GMT
- Version:** 1

Step Recap

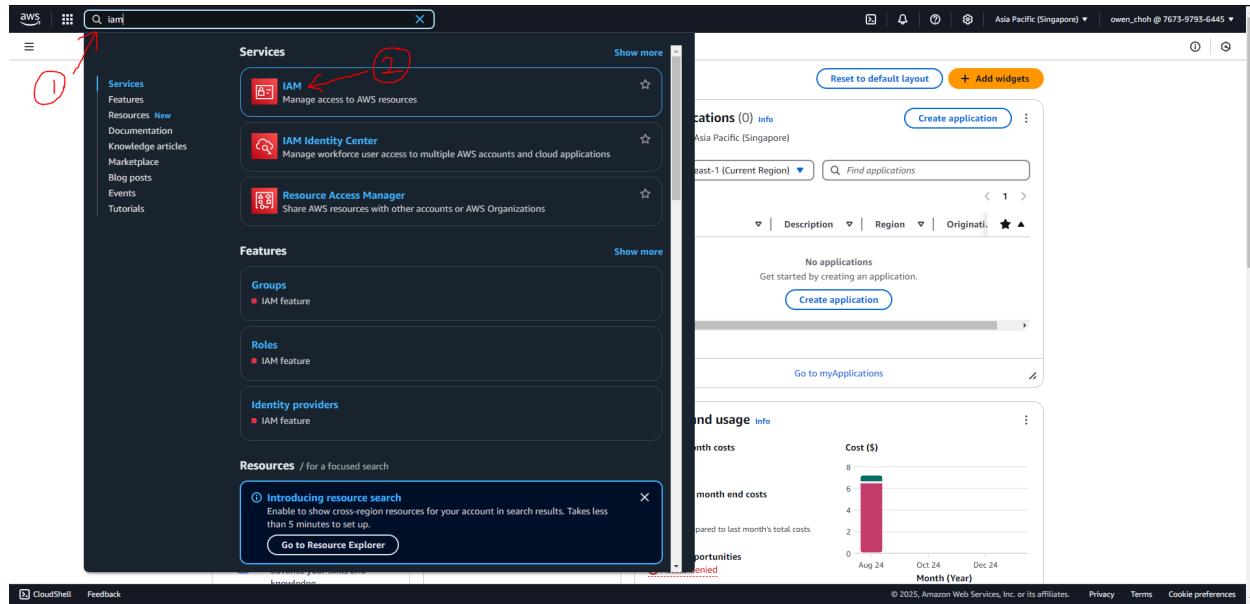
After completing this step, you should have:

1. Created 3 parameters containing the secrets.
 - o Encryption key - name: **/SampleApp/test/secrets/encrypt**
 - o Certificate - name: **/SampleApp/test/secrets/cert**
 - o Private Key - name: **/SampleApp/test/secrets/key**
2. Recorded one of the arn of the parameters for reference later.
 - o For example,
“`arn:aws:ssm:ap-southeast-1:767397936445:parameter/SampleApp/test/secrets/encrypt`”.

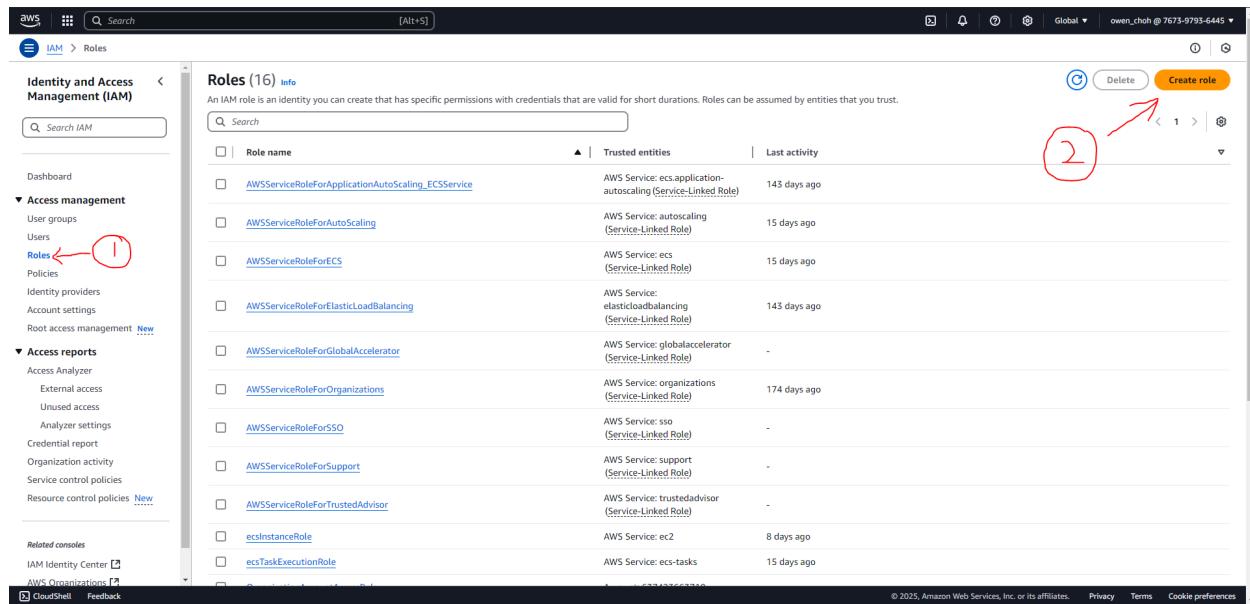
Step 2a: IAM Role Setup

The Sample App will need two roles to function, one for the EC2 instance and one to retrieve the secret from the parameter store to align with best practices.

1. Navigate to the IAM console.



2. Click on “Roles” under Access management and click “Create role”



3. This IAM role is for the EC2 instance later. Select “**AWS service**” under Trusted entity type and select “**EC2**” under Use case. Then click on “**Next**” at the bottom of the screen to proceed.

Step 1
 Select trusted entity [Info](#)
 Step 2
 Step 3
 Name, review, and create

Trusted entity type

- AWS service Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

Use case
 Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Choose a service or use case

[Cancel](#) [Next](#)

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

aws Search [Alt+S] Global owner_choh @ 7673-9793-6445

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Step 2
 Add permissions
 Create role

Use case
 Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Choose a service or use case

SAML 2.0 federation Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

Custom trust policy Create a custom trust policy to enable others to perform actions in this account.

Use case
 Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Choose a use case for the specified service.

Use case

- EC2 Allows EC2 instances to call AWS services on your behalf.
- EC2 Role for AWS Systems Manager Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.
- EC2 Spot Fleet Role Allows EC2 Spot Fleet to request and terminate Spot Instances on your behalf.
- EC2 - Spot Fleet Auto Scaling Allows Auto Scaling to access and update EC2 spot fleets on your behalf.
- EC2 - Spot Fleet Tagging Allows EC2 to launch spot instances and attach tags to the launched instances on your behalf.
- EC2 - Spot Instances Allows EC2 Spot Instances to launch and manage spot instances on your behalf.
- EC2 - Spot Fleet Allows EC2 Spot Fleet to launch and manage spot fleet instances on your behalf.
- EC2 - Scheduled Instances Allows EC2 Scheduled Instances to manage instances on your behalf.

[Cancel](#) [Next](#)

4. On this screen, scroll to the bottom and click next

The screenshot shows the 'Add permissions' step in the IAM wizard. The left sidebar has three steps: Step 1 (Select trusted entity), Step 2 (Add permissions, which is selected), and Step 3 (Name, review, and create). The main area is titled 'Add permissions' with a 'Permissions policies (1022)' link. It says 'Choose one or more policies to attach to your new role.' A search bar and a 'Filter by Type' dropdown are at the top. Below is a table of policies:

Policy name	Type	Description
AdministratorAccess	AWS managed - job function	Provides full access to AWS services an...
AdministratorAccess-Amplify	AWS managed	Grants account administrative permis...
AdministratorAccess-AWSElasticBeanstalk	AWS managed	Grants account administrative permis...
AIOpsAssistantPolicy	AWS managed	Provides ReadOnly permissions requir...
AIOpsConsoleAdminPolicy	AWS managed	Grants full access to Amazon AI Opera...
AIOpsOperatorAccess	AWS managed	Grants access to the Amazon AI Opera...
AIOpsReadOnlyAccess	AWS managed	Grants ReadOnly permissions to the A...
AlexaForBusinessDeviceSetup	AWS managed	Provide device setup access to AlexaFo...
AlexaForBusinessFullAccess	AWS managed	Grants full access to AlexaForBusiness ...
AlexaForBusinessGatewayExecution	AWS managed	Provide gateway execution access to A...
AlexaForBusinessLifesizeDelegatedAccessPolicy	AWS managed	Provide access to Lifesize AVS devices
AlexaForBusinessPolyDelegatedAccessPolicy	AWS managed	Provide access to Poly AVS devices
AlexaForBusinessReadonlyAccess	AWS managed	Provide read only access to AlexaForB...
AmazonAPIGatewayAdministrator	AWS managed	Provides full access to create/edit/dele...

- Fill in the role name and scroll to the bottom and click "Create role". It will be named "SampleAppEC2Role" in this example.

The screenshot shows the 'Name, review, and create' step in the IAM wizard. The left sidebar has three steps: Step 1 (Select trusted entity), Step 2 (Add permissions), and Step 3 (Name, review, and create, which is selected). The main area is titled 'Name, review, and create' with a 'Role details' section. It shows the 'Role name' field with 'SampleAppEC2Role' and a red arrow pointing to it. Below is a 'Description' field with the placeholder 'Allows EC2 instances to call AWS services on your behalf.' At the bottom is a 'Step 1: Select trusted entities' section with a 'Trust policy' code editor:

```

1- [{ Version: "2012-10-17",
2-   Statement: [
3-     {
4-       Effect: "allow",
5-       Action: [
6-         "sts:AssumeRole"
7-       ],
8-       Principal: [
9-         "ec2.amazonaws.com"
10-      ]
11-    }
12-  ]
13- }
14- ]
15- ]
16- ]

```

- Look for the role you just created in the list of roles and click on it.

The screenshot shows the AWS IAM Roles page. The left sidebar includes sections for Identity and Access Management (IAM), Access management, Access reports, and Related consoles. The main content area displays a table titled 'Roles (17)'. The table has columns for Role name, Trusted entities, and Last activity. One row, 'SampleAppEC2Role', is highlighted with a red arrow pointing to it.

7. On the role permissions screen, click on “Add permissions” and “Create inline policy”.

The screenshot shows the 'SampleAppEC2Role' permissions page. The left sidebar is identical to the previous screenshot. The main content area shows the 'Permissions' tab selected. At the top right, there is a 'Delete' button, an 'Edit' button, and a 'Actions' menu. The 'Actions' menu is open, with two items circled in red: 'Add permissions' (number 1) and 'Create inline policy' (number 2). Below the menu, there is a section for 'Permissions policies' and another for 'Permissions boundary'.

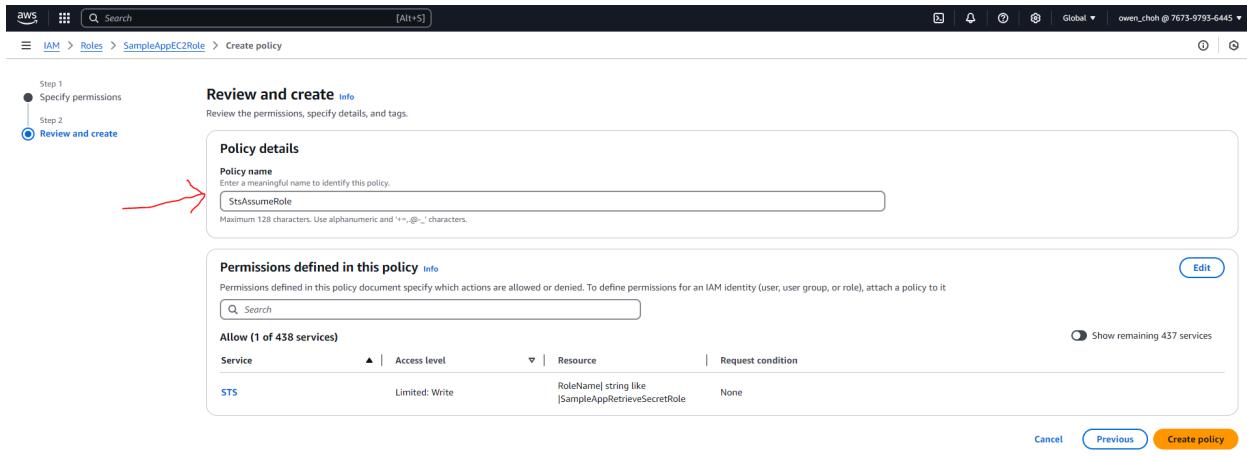
8. On the create policy screen, click on “JSON” and overwrite the lines in the “Policy editor” box. The lines are included below for your convenience. Click “Next” at the bottom of the screen once you are done.

The following are the lines to include in the policy editor. Do take note:

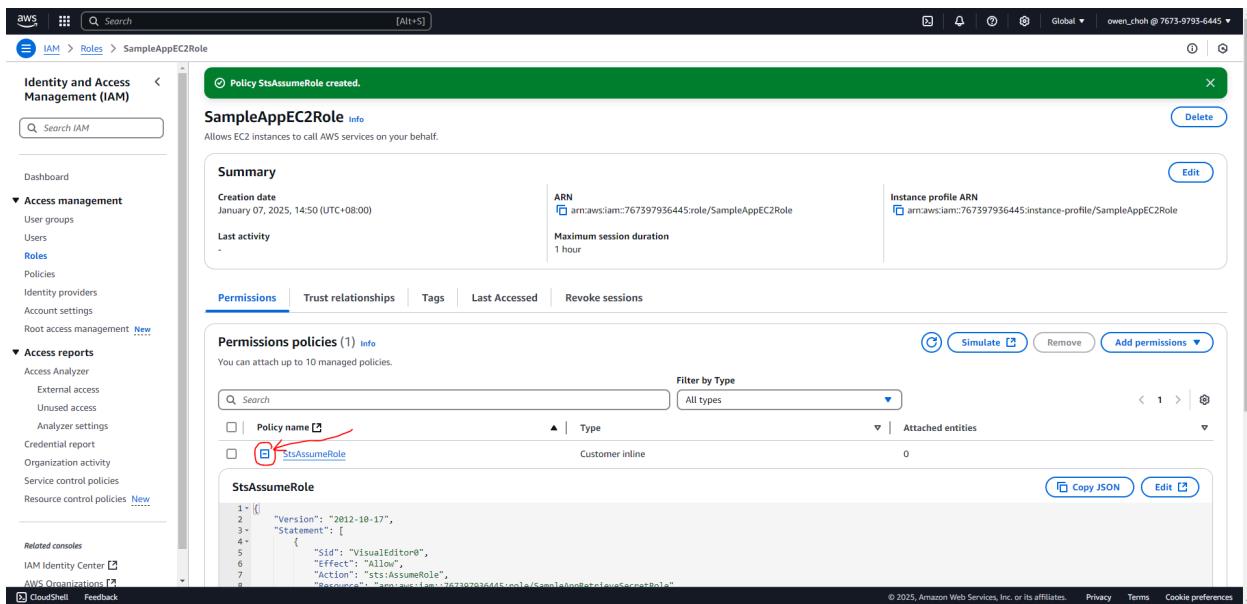
- The numbers in **green** should be your account number (found beside your username at the top right corner of the screen)
- The words in **blue** will be the name of the role (to be created in the later steps) to retrieve secrets from the parameter store.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::767397936445:role/SampleAppRetrieveSecretRole"
    }
  ]
}
```

9. Input a name for this policy. It will be named “**StsAssumeRole**” in this example. Click on “**Create policy**” once you are done.



10. This is the screen you will see once it is created. You can click on the button beside the policy name to check if the lines you pasted earlier are correct.



11. Copy the arn of the role for the EC2. It will be
"arn:aws:iam::767397936445:role/SampleAppEC2Role" in this example.

Identity and Access Management (IAM)

SampleAppEC2Role

Summary

- Creation date:** January 07, 2025, 14:50 (UTC+08:00)
- Last activity:** -
- ARN:** arn:aws:iam::767397936445:role/SampleAppEC2Role
- Maximum session duration:** 1 hour

Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Permissions policies (1) Info

You can attach up to 10 managed policies.

Policy name	Type	Attached entities
StsAssumeRole	Customer inline	0

StsAssumeRole

```

1- [
2-   "Version": "2012-10-17",
3-   "Statement": [
4-     {
5-       "Sid": "VisualEditor0",
6-       "Effect": "Allow",
7-       "Action": "sts:AssumeRole",
8-       "Resource": "arn:aws:iam::767397936445:role/SampleAppRetrieveSecretRole"
9-     }
10-   ]
11- ]
  
```

Instance profile ARN: arn:aws:iam::767397936445:instance-profile/SampleAppEC2Role

Step 2b: IAM Role to retrieve secrets

- Now we need to create a new IAM role to retrieve secrets with the same role name as the one provided when creating an inline policy (It should be **SampleAppRetrieveSecretRole** if you followed the example given). Please proceed to click on create a new role from the role screen.

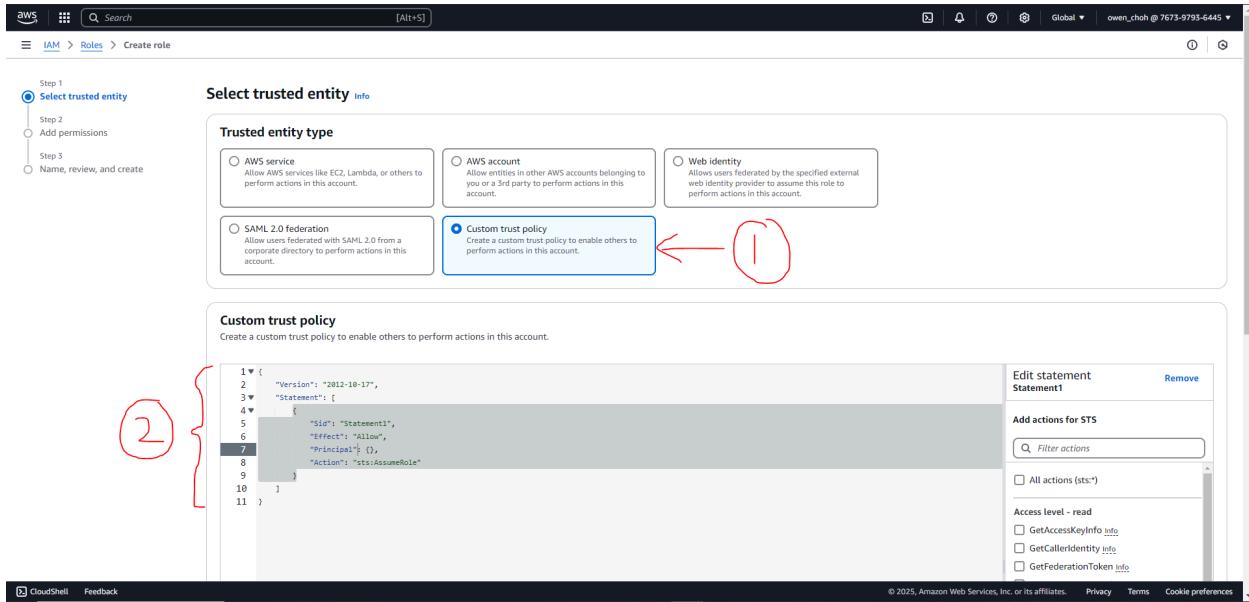
Identity and Access Management (IAM)

Roles (16) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
AWSServiceRoleForApplicationAutoScaling_ECSService	AWS Service: ecs.application-autoscaling (Service-Linked Role)	143 days ago
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Linked Role)	15 days ago
AWSServiceRoleForECS	AWS Service: ecs (Service-Linked Role)	15 days ago
AWSServiceRoleForElasticLoadBalancing	AWS Service: elasticloadbalancing (Service-Linked Role)	143 days ago
AWSServiceRoleForGlobalAccelerator	AWS Service: globalaccelerator (Service-Linked Role)	-
AWSServiceRoleForOrganizations	AWS Service: organizations (Service-Linked Role)	174 days ago
AWSServiceRoleForSSO	AWS Service: sso (Service-Linked Role)	-
AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked Role)	-
ec2InstanceRole	AWS Service: ec2	8 days ago
ecsTaskExecutionRole	AWS Service: ecs-tasks	15 days ago

- On the create role screen, click on “Custom trust policy”. You should see the box labeled as (2) below.



14. Put in the arn of the EC2 role created earlier to allow it to access this role that we are creating. Below are the lines you should see after editing the policy. Take note that the numbers in **green** should be your account number instead. Click “**Next**” when you are done and click “**Next**” again when you are on the “Add permissions” screen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::767397936445:role/SampleAppEC2Role"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

15. Input the same role name as the one provided when creating an inline policy for the EC2 role (It should be **SampleAppRetrieveSecretRole** if you followed the example given). Scroll down and click on “Create role” once you are done.

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.

Description
Add a short explanation for this role.

Step 1: Select trusted entities

Trust policy

```

1 | [ {
2 |   "Version": "2012-10-17",
3 |   "Statement": [
4 |     {
5 |       "Effect": "Allow",
6 |       "Principal": "*",
7 |       "AWS": [
8 |         "arn:aws:iam::767397936445:role/SampleAppEC2Role"
9 |       ],
10 |     },
11 |     {
12 |       "Action": "sts:AssumeRole"
13 |     }
14 |   ]
}

```

Step 2: Add permissions

16. Look for the role you just created in the list of roles and click on it.

Identity and Access Management (IAM)

Roles (18) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
AWSServiceRoleForGlobalAccelerator	AWS Service: globalaccelerator (Service-Linked Role)	-
AWSServiceRoleForSSO	AWS Service: sso (Service-Linked Role)	-
AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked Role)	-
OrganizationAccountAccessRole	Account: 637423663710	-
SampleAppEC2Role	AWS Service: ec2	-
SampleAppRetrieveSecretRole	Account: 767397936445	-
tf_secrets_role	AWS Service: ec2	-
AWSServiceRoleForOrganizations	AWS Service: organizations (Service-Linked Role)	174 days ago
AWSServiceRoleForApplicationAutoScaling_ECSService	AWS Service: ecs.application-autoscaling (Service-Linked Role)	144 days ago
AWSServiceRoleForElasticLoadBalancing	AWS Service: elasticloadbalancing (Service-Linked Role)	143 days ago

17. On the role permissions screen, click on “**Add permissions**” and “**Create inline policy**”.

Role SampleAppRetrieveSecretRole created.

SampleAppRetrieveSecretRole [Info](#)

Summary

Creation date: January 07, 2025, 15:29 (UTC+08:00)

ARN: arn:aws:iam::767397936445:role/SampleAppRetrieveSecretRole

Last activity: -

Maximum session duration: 1 hour

Permissions [Trust relationships](#) [Tags](#) [Last Accessed](#) [Revoke sessions](#)

Permissions policies (0) [Info](#)

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name: Type: No resources to display

Attached entities:

[Simulate](#) [Remove](#) [Add permissions](#)

[Attach policies](#) [Create inline policy](#)

Permissions boundary (not set)

Access denied to access-analyzer>ListPolicyGenerations

You don't have permission to access-analyzer>ListPolicyGenerations. To request access, copy the following text and send it to your AWS administrator. [Learn more about troubleshooting access denied errors.](#)

18. On the create policy screen, click on “**JSON**” and overwrite the lines in the “**Policy editor**” box. The lines are included below for your convenience. Click “**Next**” at the bottom of the screen once you are done.

Step 1 **Specify permissions** Step 2 [Review and create](#)

Specify permissions [Info](#)

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor

```

1 ▼ {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "VisualEditor0",
6       "Effect": "Allow",
7       "Action": "kms:Decrypt",
8       "Resource": "*"
9     }
10   ]
11 }
12

```

[Visual](#) **JSON** [Actions](#)

[Edit statement](#)

Select a statement

Select an existing statement in the policy or add a new statement.

[+ Add new statement](#)

10123 of 10260 characters remaining

The following are the lines to include in the policy editor to decrypt secrets when retrieving from the parameter store.

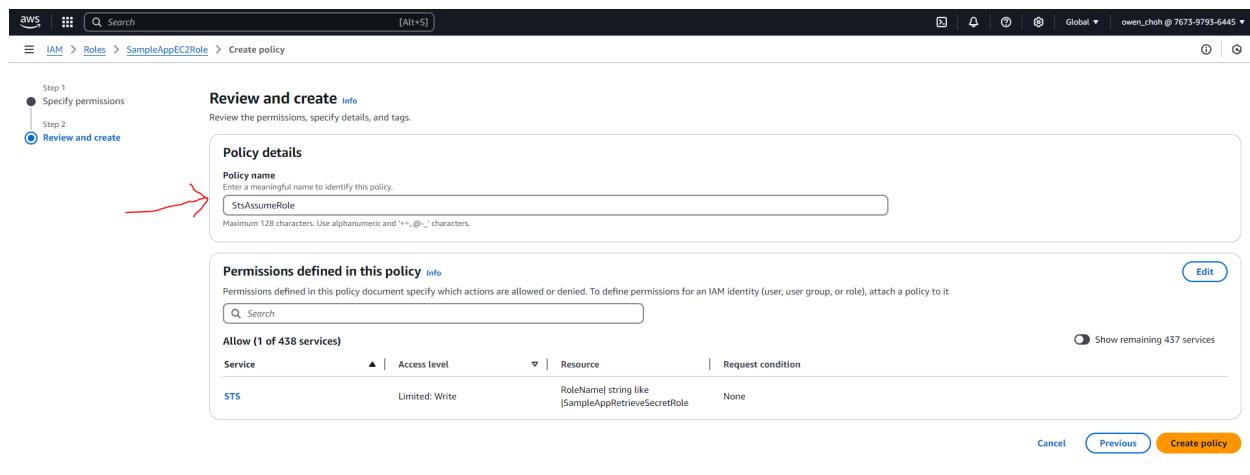
```
{
```

```

"Version": "2012-10-17",
"Statement": [
    {
        "Sid": "VisualEditor0",
        "Effect": "Allow",
        "Action": "kms:Decrypt",
        "Resource": "*"
    }
]
}

```

19. Input a name for this policy. It will be named “**KmsDecrypt**” in this example. Click on “**Create policy**” once you are done.



20. Please repeat the same steps to add the **inline policy** provided below to retrieve secrets from the parameter store. It will be named as “**SsmRetrieveSecrets**” in this example.
- Do take note that the “resource” value is dependent on the arn of the secret in the parameter store. In our example at the end of “Step 1: Parameter Store Configuration”, the arn is “arn:aws:ssm:ap-southeast-1:767397936445:parameter/**SampleApp/test/secret s/encrypt**”. Thus the resource value here should be “.../SampleApp/test/secrets/*”.
 - Please also remember to change the numbers in **green** to your account number.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "ssm:GetParametersByPath",
      "Resource": "arn:aws:ssm:ap-southeast-1:767397936445:parameter/SampleApp/test/secrets/*"
    }
  ]
}
```

21. This is the screen you will see once both policies are created. You can click on the button beside the policy name to check if the lines you pasted earlier are correct.

The screenshot shows the AWS IAM Role details page for 'SampleAppRetrieveSecretRole'. The top navigation bar includes 'Search' and 'Global'. The left sidebar shows 'Identity and Access Management (IAM)' with sections for 'Dashboard', 'Access management' (selected), 'Access reports', and 'Related consoles'. The main content area displays the role's summary, including its ARN (arn:aws:iam::767397936445:role/SampleAppRetrieveSecretRole) and creation date (January 07, 2025, 15:29 (UTC+08:00)). The 'Permissions' tab is selected, showing two attached policies: 'KmsDecrypt' and 'SsmRetrieveSecrets', both highlighted with red circles. A green banner at the top states 'Policy SsmRetrieveSecrets created.' Below the policies, there are tabs for 'Trust relationships', 'Tags', 'Last Accessed', and 'Revoke sessions'. At the bottom, a red box highlights an error message: 'Access denied to access-analyzer>ListPolicyGenerations'.

The screenshot shows the AWS IAM Policies page. At the top, there are two policy entries:

- KmsDecrypt**: Customer inline policy with 1 statement. The JSON code grants permission to "VisualEditor0" to "KmsDecrypt" on all resources.
- SsmRetrieveSecrets**: Customer inline policy with 1 statement. The JSON code grants permission to "VisualEditor0" to "ssm:GetParametersByPath" on the specific resource "arn:aws:ssm:ap-southeast-1:767397936445:parameter/SampleApp/test/secrets/*".

22. Copy the arn of the role to retrieve secrets as you will need it later.

The screenshot shows the AWS IAM Roles page. A single role named "SampleAppRetrieveSecretRole" is listed. The ARN of the role is highlighted with a red box: `arn:aws:iam::767397936445:role/SampleAppRetrieveSecretRole`.

Step Recap

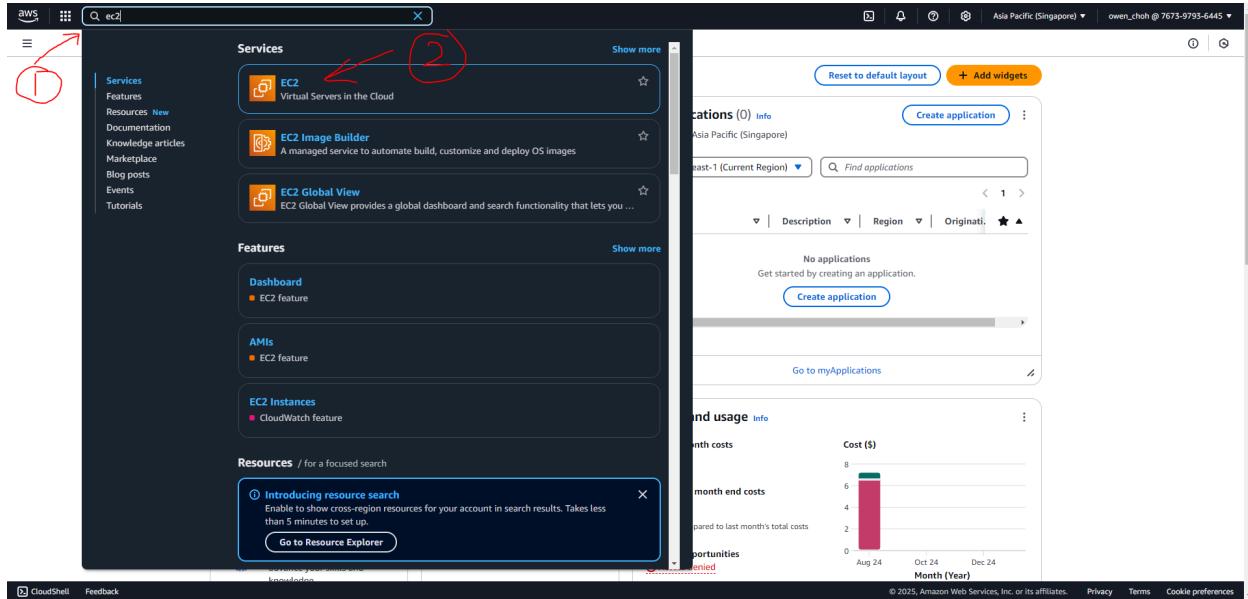
After completing step 2A and 2B, you should have:

1. Created two roles:
 - a. "**SampleAppEC2Role**" for the EC2 instance
 - b. "**SampleAppRetrieveSecretRole**" for the EC2 to retrieve secrets
2. Copied the arn of the "**SampleAppRetrieveSecretRole**" e.g.
`arn:aws:iam::767397936445:role/SampleAppRetrieveSecretRole`

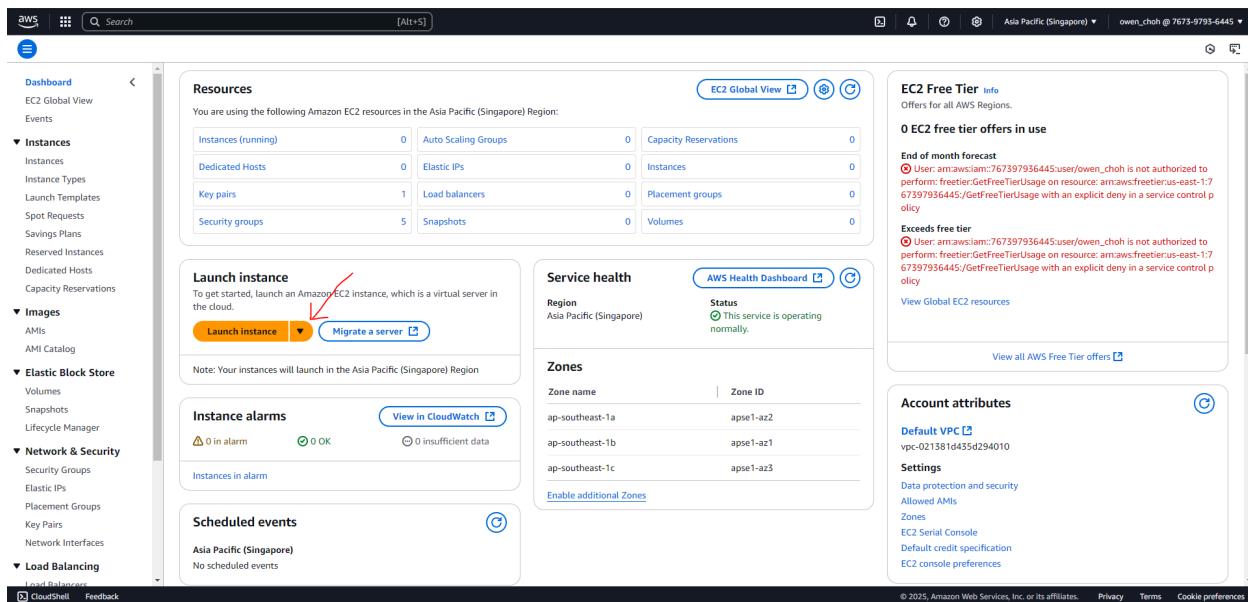
Step 3: EC2 Instance Launch Configuration

This step is to start an EC2 instance on AWS which will host the Sample App for people to use.

1. Navigate to the EC2 dashboard



2. Launch an EC2 instance



3. Input a name for the instance such as “SampleAppInstance”

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below.

Search our full catalog including 1000s of application and OS images

Recents Quick Start

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Linux Debian

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI
ami-0995922d49dc9a17d (64-bit (x86), uefi-preferred) / ami-0f3785bf40feb970e (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Amazon Linux 2023 AMI 2023.6.20241212.0-x86_64 HVM kernel-6.1

Summary

Number of instances Info

Software Image (AMI)
Amazon Linux 2023 AMI 2023.6.2... [read more](#)
ami-0995922d49dc9a17d

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million IOPS, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Cancel Launch instance Preview code

- In this example, we will use the default options for most of the options but you can choose other options if needed. Screenshots are provided in case you like to verify them.

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below.

Search our full catalog including 1000s of application and OS images

Recents Quick Start

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Linux Debian

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI
ami-0995922d49dc9a17d (64-bit (x86), uefi-preferred) / ami-0f3785bf40feb970e (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Amazon Linux 2023 AMI 2023.6.20241212.0-x86_64 HVM kernel-6.1

Architecture 64-bit (x86) **Boot mode** uefi-preferred **AMI ID** ami-0995922d49dc9a17d **Username** ec2-user **Verified provider**

Free tier eligible

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI
ami-0995922d49dc9a17d (64-bit (x86), uefi-preferred) / ami-0f3785bf40feb970e (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Amazon Linux 2023 AMI 2023.6.20241212.0-x86_64 HVM kernel-6.1

Architecture 64-bit (x86) **Boot mode** uefi-preferred **AMI ID** ami-0995922d49dc9a17d **Username** ec2-user **Verified provider**

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true
 On-Demand Ubuntu Pro base pricing: 0.0164 USD per Hour On-Demand Linux base pricing: 0.0146 USD per Hour
 On-Demand Windows base pricing: 0.0192 USD per Hour On-Demand RHEL base pricing: 0.029 USD per Hour
 On-Demand SUSE base pricing: 0.0146 USD per Hour

[All generations](#) [Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

▼ Configure storage [Info](#)

Root volume 3000 IOPS (Not encrypted)

1x 8 GiB gp3

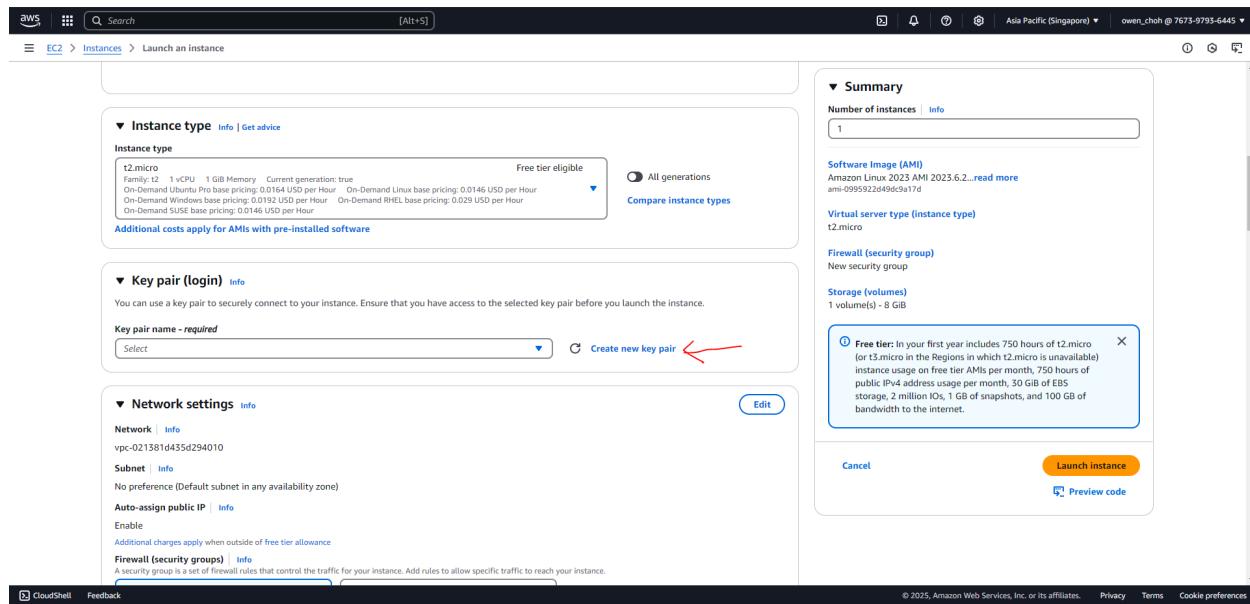
(i) Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

[Add new volume](#)

(i) Click refresh to view backup information
 The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.

0 x File systems [Edit](#)

5. Click on create a new key pair.



The screenshot shows the AWS EC2 'Launch an instance' wizard. The current step is 'Key pair (login)'. It asks for a key pair name (required) and provides a 'Create new key pair' button. Below this, there are 'Network settings' and a summary section. The summary section includes details about the instance type (t2.micro), storage (1 volume(s) - 8 GiB), and free tier usage. At the bottom right are 'Cancel', 'Launch instance', and 'Preview code' buttons.

6. Input a name for the key pair such as "SampleAppInstanceKey" and click "Create key pair". Your browser should prompt you to download a file (named SampleAppInstanceKey.pem in this example) automatically, please save this file as it is

required to access the instance once it is running and there is no way to obtain this file again should you lose it.

Create key pair

Key pair name
Key pairs allow you to connect to your instance securely.

SampleAppInstanceKey

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

RSA
RSA encrypted private and public key pair

ED25519
ED25519 encrypted private and public key pair

Private key file format

.pem
For use with OpenSSH

.ppk
For use with PuTTY

⚠️ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#) ↗

Cancel **Create key pair**

7. Check these two boxes under the “Network settings”

Key pair name - required
SampleAppInstanceKey [Create new key pair](#)

Network settings [Info](#)

Network [Info](#)
vpc-021381d435d294010

Subnet [Info](#)
No preference (Default subnet in any availability zone)

Auto-assign public IP [Info](#)
Enable

Firewall (security groups) [Info](#)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group Select existing security group

We'll create a new security group called 'launch-wizard-5' with the following rules:

- Allow SSH traffic from Anywhere (0.0.0.0/0) Helps you connect to your instance
- Allow HTTPS traffic from the internet To set up an endpoint, for example when creating a web server
- Allow HTTP traffic from the internet To set up an endpoint, for example when creating a web server

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Configure storage [Info](#)

Advanced

Summary
Number of instances [Info](#)
1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.6.2... [read more](#)
ami-0995922d49de9a17d

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

[Cancel](#) [Launch instance](#) [Preview code](#)

8. Assign the IAM role created in Step 2 to the instance under the “Advanced details” tab. Click on “Launch instance” on the right once you are done.

Add new volume

Click refresh to view backup information
The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.

O File systems [Edit](#)

Advanced details [Info](#)

Domain join directory [Info](#)
Select [Create new directory](#)

IAM instance profile [Info](#)
 SampleAppEC2Role [aws:iam::767397935445:instance-profile/SampleAppEC2Role](#) [Create new IAM profile](#)

Hostname type [Info](#)
IP name

DNS Hostname [Info](#)
 Enable IP name IPv4 (A record) DNS requests
 Enable resource-based IPv4 (A record) DNS requests
 Enable resource-based IPv6 (AAAA record) DNS requests

Instance auto-recovery [Info](#)
Select

Shutdown behavior [Info](#)
Stop

Stop - Hibernate behavior [Info](#)

Summary
Number of instances [Info](#)
1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.6.2... [read more](#)
ami-0995922d49de9a17d

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

[Cancel](#) [Launch instance](#) [Preview code](#)

9. Click on the instance you just created.

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with links like Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces, Load Balancing, and CloudWatch Metrics. The main area shows a table titled 'Instances (1/1) Info' with one item: 'SampleApplInstance' (Instance ID: i-07571b68e218eac93). The instance is listed as 'Running' (Status check: Initializing). It has a Public IPv4 DNS of 'ec2-54-255-135-233.ap-southeast-1.compute.amazonaws.com' and a Public IPv4 of '54.255.135.233'. A red arrow points to the instance row in the table.

10. Copy the “Public IPv4 DNS”, this is the link that users will use to access the sample app once it is running. Please take note that you need to append “http://” to the front of the link to use it (e.g. <http://ec2-52-77-81-200.ap-southeast-1.compute.amazonaws.com>).

The screenshot shows the AWS EC2 Instance details page for the instance 'i-07571b68e218eac93'. The left sidebar is identical to the previous screenshot. The main area shows the 'Instance summary for i-07571b68e218eac93 (SampleApplInstance)' section. It displays the instance's public IPv4 address as '54.255.135.233' and its public DNS name as 'ip-172-31-23-13.ap-southeast-1.compute.internal'. A red arrow points to the 'Public IPv4 address' field. The 'Details' tab is selected, showing other instance details like IAM Role ('SampleApplEC2Role'), VPC ID ('vpc-021381d435d294010'), and Subnet ID ('subnet-0f245fc9e866ed24').

Step Recap

After completing this step, you should have:

1. An EC2 instance running with the correct IAM role
2. Obtained a key to access the instance (e.g. SampleApplInstanceKey.pem)

- A “Public IPv4 DNS” for users to use to access the sample app once it is running (e.g. <http://ec2-52-77-81-200.ap-southeast-1.compute.amazonaws.com>)

Step 4: EC2 Instance installation

This step is to install the dependencies in order to run the Sample App.

- Connect to the instance via one of two methods. Both are equivalent.
 - Using the aws console

Instance summary for i-07571b68e218eac93 (SampleAppInstance) Info

Updated less than a minute ago

Instance ID: i-07571b68e218eac93

IPv4 address: -

Hostname type: IP name: ip-172-31-23-13.ap-southeast-1.compute.internal

Answer private resource DNS name: IPv4 (A)

Auto-assigned IP address: 54.255.135.233 [Public IP]

IAM Role: SampleAppEC2Role

IMDSv2: Required

Operator: -

Private IP DNS name (IPv4 only): ip-172-31-23-13.ap-southeast-1.compute.internal

Instance type: t2.micro

VPC ID: vpc-021381d435d294010

Subnet ID: subnet-0f245fc9e866ed24

Instance ARN: arn:aws:ec2:ap-southeast-1:767397936445:instance/i-07571b68e218eac93

Public IPv4 address: 54.255.135.233 [open address]

Public IPv4 DNS: ec2-54-255-135-233.ap-southeast-1.compute.amazonaws.com [open address]

Elastic IP addresses: -

AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations. | Learn more

Auto Scaling Group name: -

Managed: false

Connect to instance Info

Connect to your instance i-07571b68e218eac93 (SampleAppInstance) using any of these options

EC2 Instance Connect | **Session Manager** | **SSH client** | **EC2 serial console**

Instance ID: i-07571b68e218eac93 (SampleAppInstance)

Connection Type:

- Public IPv4 address**: 54.255.135.233
- IPv6 address**: -

Username: Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.

ec2-user X

Note: In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Cancel **Connect**

This is the screen once connected via the aws console:

i-07571b68e218eac93 (SampleAppInstance)
PublicIP: 54.255.135.233 PrivateIP: 172.31.23.13

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- Using the ssh command “ssh -i <relative path to the instance key> ec2-user@<dns of the instance>”
 - Type “yes” if the ssh command produces a warning about a new fingerprint
 - Refer to this page (<https://superuser.com/questions/1296024/windows-ssh-permissions-for-private-key-are-too-open>) if there is a warning about the key being unprotected

```
@@@@@@@WARNING: UNPROTECTED PRIVATE KEY FILE!@@@@@@@  
Permissions for 'private-key.ppk' are too open.  
It is required that your private key files are NOT accessible by others.  
This private key will be ignored.
```

- Load key "private-key.ppk": bad permissions

Example ssh command:

```
ssh -i SampleAppInstanceKey.pem  
ec2-user@ec2-52-77-81-200.ap-southeast-1.compute.amazonaws.com
```

- Once connected, type in the following commands
 - To update the instance package manager
 - sudo yum update -y
 - To install docker
 - sudo yum install -y docker

```
Installed:
  containerd-1.7.23-1.amzn2023.0.1.x86_64
  libcuprof-3.0.1.amzn2023.0.1.x86_64
  pigrx-2.5.1.amzn2023.0.3.x86_64

Available Updates:
  docker-25.0.6-1.amzn2023.0.2.x86_64
  libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
  runc-1.1.14-1.amzn2023.0.1.x86_64

Complete!
```

(you will be able to type once its done)

- Once installed, start docker
 - sudo service docker start
 - Once started, give yourself permissions
 - sudo usermod -a -G docker ec2-user
 - newgrp docker

```
[ec2-user@ip-172-31-23-13 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-31-23-13 ~]$ sudo usermod -a -G docker ec2-user
[ec2-user@ip-172-31-23-13 ~]$ newgrp docker
```

- Check that you have permissions and docker is running
 - `docker ps`

```
[ec2-user@ip-172-31-23-13 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND       CREATED          STATUS          PORTS          NAMES
```

Step Recap

After this step, you should have installed docker on the EC2 instance where the Sample App will run.

Step 5: Sample App Deployment

This step is to get the source code to the instance and run it using the docker service installed in the previous step.

Step 5a: Transferring the Sample App files

This sub step is to describe how to transfer the files to the EC2 instance.

1. Connect to the instance (Please refer to the previous “Step 4: EC2 Instance installation” if you need help with this)
2. (Optional) Make a new folder called “downloads” in the instance so that it is more tidy by typing the commands below.
 - o Make a new directory
 - mkdir downloads
 - o Navigate to the directory
 - cd downloads
 - o The commands above do not produce any outputs but you can check that you are in the right directory by typing “pwd”.
3. Copy the Sample App code to the EC2 instance via one of two methods.
 - a. Using scp command to transfer files from your laptop to the instance
 - b. Downloading directly from github

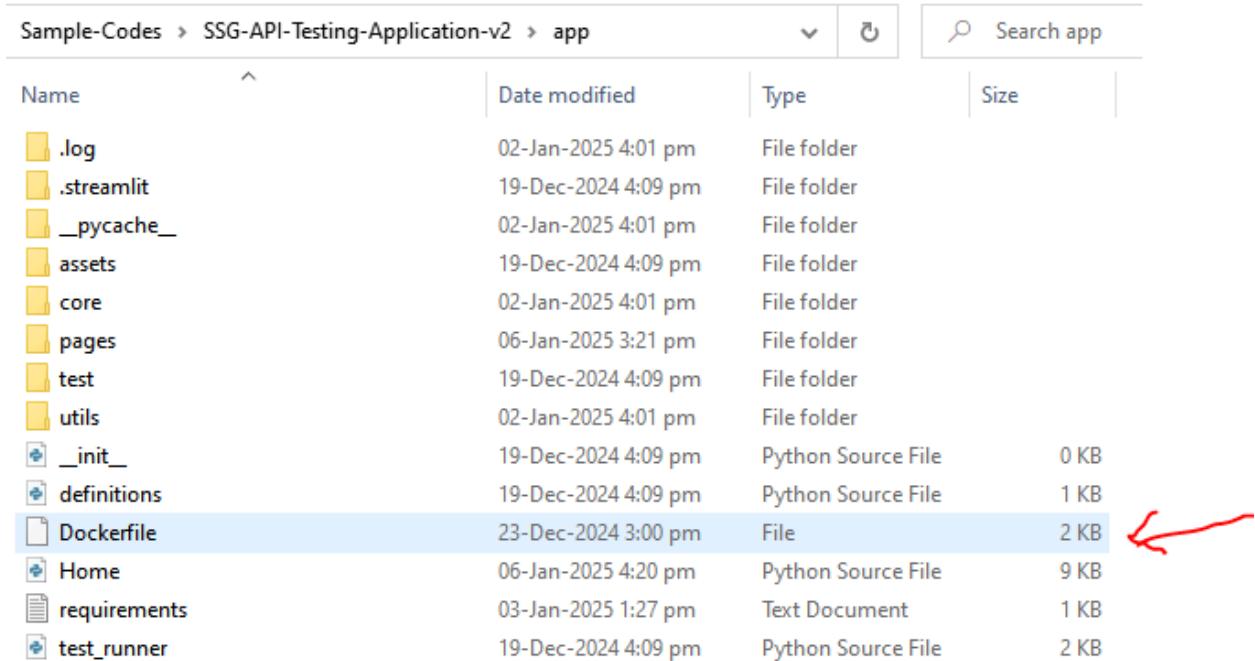
```
[ec2-user@ip-172-31-17-157 downloads]$ pwd  
/home/ec2-user/downloads
```

3a. Transferring files from your laptop

This is the preferred method as there will be less unnecessary files and dependencies in the instance.

1. Download the files from github if you do not have it on your laptop
2. Navigate to the directory with the source code on your laptop (look for the location with the dockerfile as shown below)

Sample-Codes > SSG-API-Testing-Application-v2 > app



Name	Date modified	Type	Size
.log	02-Jan-2025 4:01 pm	File folder	
.streamlit	19-Dec-2024 4:09 pm	File folder	
__pycache__	02-Jan-2025 4:01 pm	File folder	
assets	19-Dec-2024 4:09 pm	File folder	
core	02-Jan-2025 4:01 pm	File folder	
pages	06-Jan-2025 3:21 pm	File folder	
test	19-Dec-2024 4:09 pm	File folder	
utils	02-Jan-2025 4:01 pm	File folder	
__init__	19-Dec-2024 4:09 pm	Python Source File	0 KB
definitions	19-Dec-2024 4:09 pm	Python Source File	1 KB
Dockerfile	23-Dec-2024 3:00 pm	File	2 KB
Home	06-Jan-2025 4:20 pm	Python Source File	9 KB
requirements	03-Jan-2025 1:27 pm	Text Document	1 KB
test_runner	19-Dec-2024 4:09 pm	Python Source File	2 KB

3. Open a terminal / command prompt and send the files to the instance using the command below
 - scp -i <relative location of the instance key> -r <relative location of the source code> ec2-user@<dns of the instance>:<where to put your files on the instance>
 - It should look like the command below if the terminal is in this directory

Name	Date modified	Type	Size
Sample-Codes	07-Jan-2025 6:01 pm	File folder	
SampleApplianceKey.pem	07-Jan-2025 4:10 pm	PEM File	2 KB

 -
 - scp -i mykey.pem -r ./Sample-Codes/SSG-API-Testing-Application-v2/app/ ec2-user@ec2-52-77-81-200.ap-southeast-1.compute.amazonaws.com:/home/ec2-user/downloads/

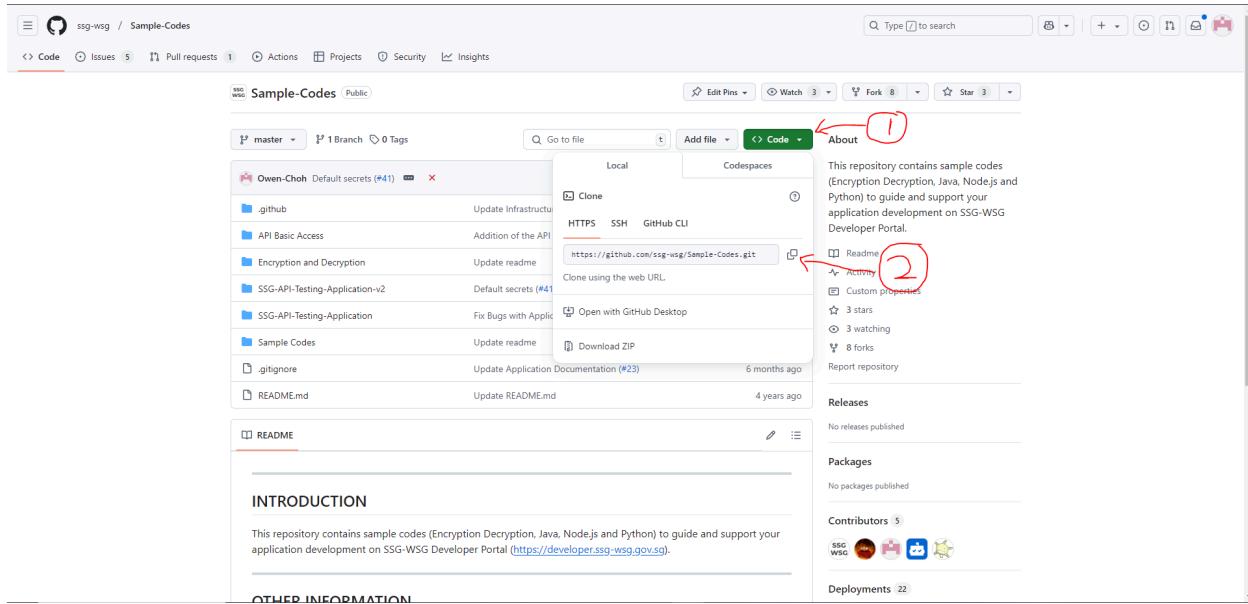
4. Once it stops running, you can connect to the instance and navigate to the directory specified above to check that it is there. (screenshot below assumes you are connected and navigated to the “downloads” directory)

```
[ec2-user@ip-172-31-17-157 downloads]$ ls
app
[ec2-user@ip-172-31-17-157 downloads]$ cd app
[ec2-user@ip-172-31-17-157 app]$ ls
Dockerfile __init__.py assets definitions.py requirements.txt test_runner.py
Home.py __pycache__ core pages test utils
[ec2-user@ip-172-31-17-157 app]$
```

5. Proceed to “Step 5b: Starting the Sample App” once this is done

3b. Downloading from github

1. Copy the link (2) in the image from the SSG repository on github (<https://github.com/ssg-wsg/Sample-Codes>)



2. Connect to the instance (Please refer to the previous “Step 4: EC2 Instance installation” if you need help with this step)
3. Install git using the command
 - sudo yum install git -y
4. Download the files from the link you copied earlier using the command (it may take a while since it will need to download the entire repository)
 - git clone https://github.com/ssg-wsg/Sample-Codes.git
5. Type in the following commands to check that the files are there and to navigate to the correct directory

```
[ec2-user@ip-172-31-17-157 test]$ git clone https://github.com/ssg-wsg/Sample-Codes.git
Cloning into 'Sample-Codes'...
remote: Enumerating objects: 1100, done.
remote: Counting objects: 100% (441/441), done.
remote: Compressing objects: 100% (338/338), done.
remote: Total 1100 (delta 259), reused 121 (delta 103), pack-reused 659 (from 2)
Receiving objects: 100% (1100/1100), 708.64 MiB | 17.73 MiB/s, done.
Resolving deltas: 100% (391/391), done.
Updating files: 100% (429/429), done.
[ec2-user@ip-172-31-17-157 test]$ ls
Sample-Codes
[ec2-user@ip-172-31-17-157 test]$ cd Sample-Codes/SSG-API-Testing-Application-v2/app
[ec2-user@ip-172-31-17-157 app]$ ls
Dockerfile Home.py __init__.py assets core definitions.py pages requirements.txt test test_runner.py utils
[ec2-user@ip-172-31-17-157 app]$ |
```

6. Proceed to “Step 5b: Starting the Sample App” once this is done

Step 5b: Starting the Sample App

This sub step assumes that you have:

1. Placed the secrets in the parameter store.
2. Created the required IAM roles.
3. Created an EC2 instance.
4. Transferred the Sample App files to the instance and you are in the correct directory.

Please refer to the previous steps in this document if there are missing items.

This sub step is to describe how to start the Sample App once the files are in the instance.

1. Build the container image that the Sample App will run in using the command below
 - o Please note that the strings in bold are following the examples provided in the previous steps. Do make sure to amend them if your values are different else the app will not be able to retrieve the secrets from the parameter store.
 - o docker build --build-arg SECRET_PATH="**/SampleApp/test/secrets/**" --build-arg SECRET_ENCRYPTION_KEY_PATH="**/SampleApp/test/secrets/encrypt**" --build-arg SECRET_CERT_PATH="**/SampleApp/test/secrets/cert**" --build-arg SECRET_KEY_PATH="**/SampleApp/test/secrets/key**" --build-arg ROLE_ARN="arn:aws:iam::**767397936445**:role/**SampleAppRetrieveSecretRole**" --build-arg REGION_NAME="ap-southeast-1" -t **ssg/sample-app**.
 - i. SECRET_PATH="**/SampleApp/test/secrets/**" is the path where all 3 secrets are stored.

```
[ec2-user@ip-172-31-17-157 app]$ docker build --build-arg SECRET_PATH="/SampleApp/test/secrets/" --build-arg SECRET_ENCRYPTION_KEY_PATH="/SampleApp/test/secrets/encrypt" --build-arg SECRET_CERT_PATH="/SampleApp/test/secrets/cert" --build-arg SECRET_KEY_PATH="/SampleApp/test/secrets/key" --build-arg ROLE_ARN="arn:aws:iam::637423663710:role/TestSecretRole" --build-arg REGION_NAME="ap-southeast-1" -t ssg/sample-app .
[+] Building 1.9s (10/10) FINISHED
=> [internal] load build definition from Dockerfile                               docker:default
=> => transferring dockerfile: 1.35kB                                         0.0s
=> [internal] load metadata for docker.io/library/python:3.12-slim           1.8s
=> [internal] load .dockerignore                                              0.0s
=> => transferring context: 2B                                              0.0s
=> [1/5] FROM docker.io/library/python:3.12-slim@sha256:10f3aaab98db50cba827d3b33a91f39dc9ec2d02ca9b85cbc5008220 0.0s
=> [internal] load build context                                             0.0s
=> => transferring context: 26.62kB                                         0.0s
=> CACHED [2/5] WORKDIR /app                                                 0.0s
=> CACHED [3/5] COPY requirements.txt requirements.txt                      0.0s
=> CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt          0.0s
=> CACHED [5/5] COPY . .                                                 0.0s
=> exporting to image                                                       0.0s
=> => exporting layers                                                       0.0s
=> => writing image sha256:7e63b558b859d63c97b74a1293621b4c90cfcb5b474b2c94b15e9dbbe35cc0347 0.0s
=> => naming to docker.io/ssg/sample-app                                     0.0s
[ec2-user@ip-172-31-17-157 app]$ |
```

The screenshot is for reference only. It takes about a minute to download items and complete building if this is your first time running the command.

2. Once your image is built, run the command below to start the container
 - o docker run -d --rm -p 80:80 --name **sampleapp** **ssg/sample-app**
 - o The value in **magenta** must be the same as the build command above.

- The value in orange is the name of the container that the Sample App is running in.
- The output of the command does not matter
- Please refer to <https://docs.docker.com/reference/cli/docker/container/run/> for detailed explanation of the command.

```
[ec2-user@ip-172-31-17-157 app]$ docker run -d --rm -p 80:80 --name sampleapp ssg/sample-app  
b1f17c4ba9e90fd092205373ad134a420bd77eb094c0579429e76a5a51febb6f  
[ec2-user@ip-172-31-17-157 app]$
```

Step Recap

After this step, you should have:

- Transferred the Sample App files to the instance
- Started the Sample App on the instance
- Able to access the Sample App via the “**Public IPv4 DNS**” of the instance (e.g. <http://ec2-52-77-81-200.ap-southeast-1.compute.amazonaws.com>)
 - You can think of a “**Public IPv4 DNS**” as a website link.

Additional Notes

To stop the container that is running the app, run the command below

- Please note that you can **skip this step** if you want to stop the instance as this will only stop the Sample App.
 - docker stop sampleapp
- ```
[ec2-user@ip-172-31-17-157 app]$ docker stop sampleapp
sampleapp
```
- 

## Associate an elastic ip address to your EC2

- Please note that this step is **not necessary** for the Sample App to function and only helps to stop the “**Public IPv4 DNS**” of the instance from changing.
- If you want your EC2 instance to use an IP address you have in your AWS account, please refer to the steps in the link below.
- <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/working-with-eips.html#using-instance-addressing-eips-associating>

## Disassociate the elastic ip address to your EC2

- Please note that you can **skip this step** if you want to stop the instance as this will only “unlink” the IP address from the EC2 and change the “**Public IPv4 DNS**” of the instance.
- If you associate an IP address to the EC2 instance and you want to disassociate it, please refer to the steps in the link below.
- <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/working-with-eips.html#using-instance-addressing-eips-associating-different>
- If you terminate an instance, AWS will automatically disassociate the IP address from the instance and **will not** release your IP address.

## To stop the EC2 instance

- Choose the state that you require from this dropdown on the EC2 dashboard
- “**Stop instance**” will allow you to “**Start instance**” later but will still incur charges from:
  - The instance storage - the “hard disk” of the instance
  - You may still incur charges for the ip address if you reserved one under the “**Elastic IPs**” tab on the left. Otherwise, the “**Public IPv4 DNS**” of the instance will change after you “**Start instance**” later.

- “Terminate (delete) instance” will delete the instance and storage forever (It may remain on the list for a while after deletion). You will not be able to start this instance again and will need to install and deploy the app again. You may still incur charges for the ip address if you reserved one under the “Elastic IPs” tab on the left.

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with links like Dashboard, EC2 Global View, Events, Instances (which is expanded), Images, Elastic Block Store, Network & Security, and Load Balancing. The main area displays a table of instances. One instance is selected, highlighted with a blue border. The Actions menu for this instance is open, showing options like Stop instance, Start instance, Reboot instance, Hibernate instance, and Terminate (delete) instance. A red circle labeled '1' points to the 'Actions' button in the top right of the table header. A red circle labeled '2' points to the 'Terminate (delete) instance' button in the bottom right of the open Actions menu.

## Delete parameters from parameter store

- It is currently under the “always free” tier to store “standard” tier secrets in the parameter store but you can choose to delete them if they are not needed.

The screenshot shows the AWS Systems Manager Parameter Store. The top navigation bar includes links for Services, Search, and RAM sharing. Below it, a banner introduces cross-account parameter sharing. The main area is titled "Parameter Store" and has tabs for "My parameters", "Public parameters", and "Settings". Under "My parameters", there's a search bar and a table of parameters. The first parameter listed is "/SampleApp/test/secrets/cert", which is a SecureString type created on Tue, 07 Jan 2025. A red circle labeled '1' points to the search bar. A red circle labeled '2' points to the "Delete" button in the top right of the parameter table.