

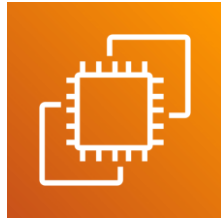


## CREATE A SIMPLE PIPELINE

**WE ARE USING THESE AWS SERVICES  
TO CREATE PIPELINE.**



**S3**



**EC2**



**CODE  
DEPLOY**



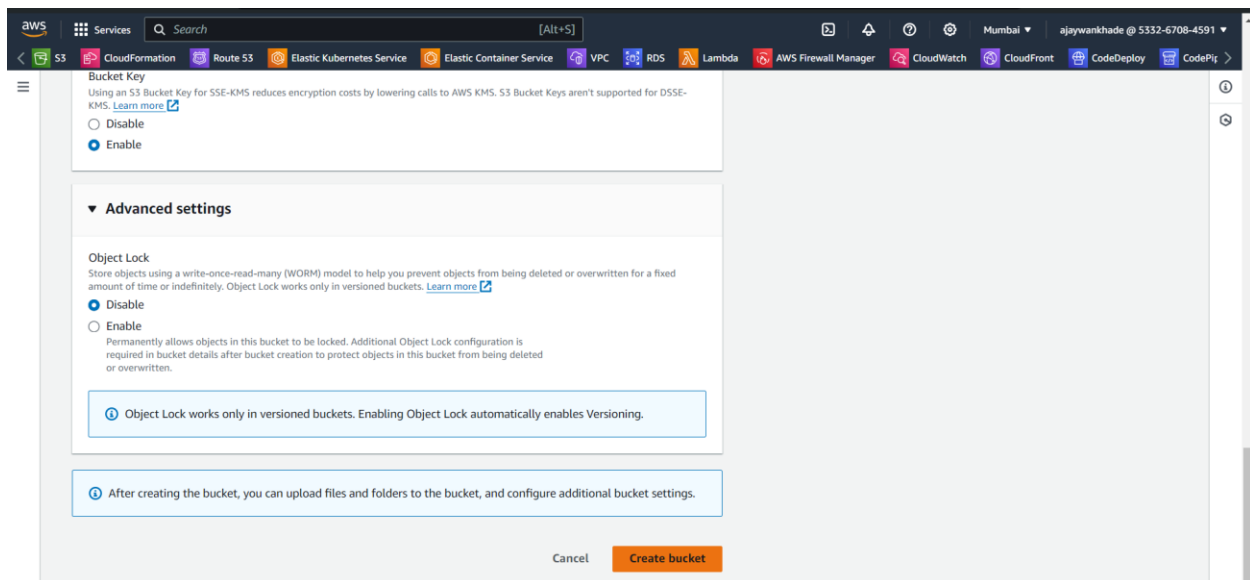
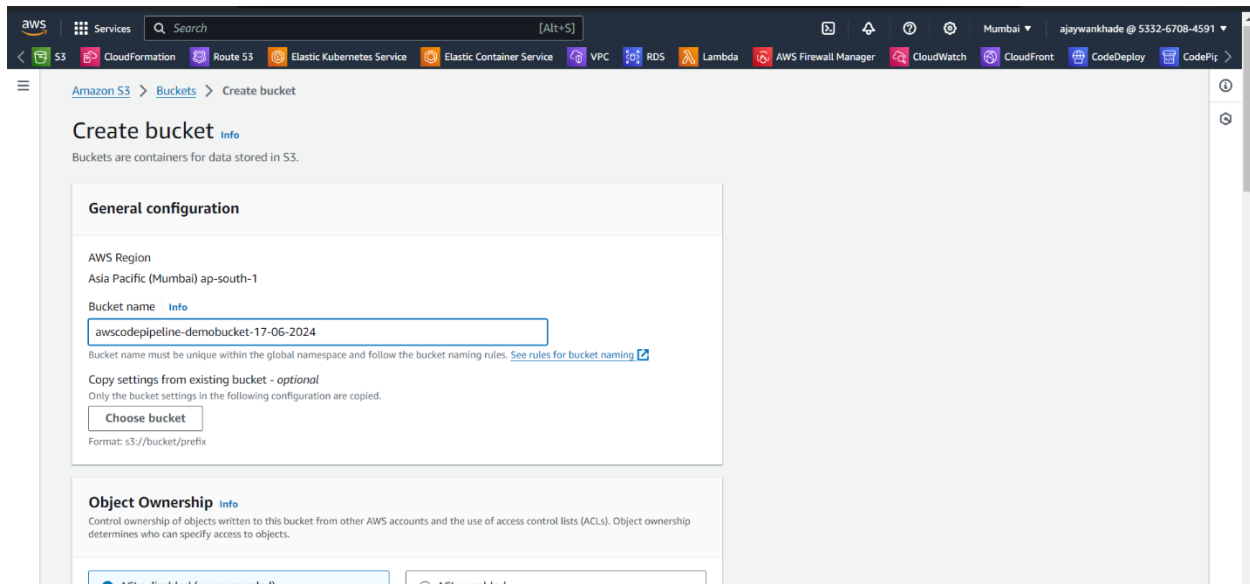
**CODE  
PIPELINE**

### **Step 1:** Create an S3 bucket for your application.

You can store your source files or applications in any versioned location. In this tutorial, you create an S3 bucket for the sample application files and enable versioning on that bucket. After you have enabled versioning, you copy the sample applications to that bucket.

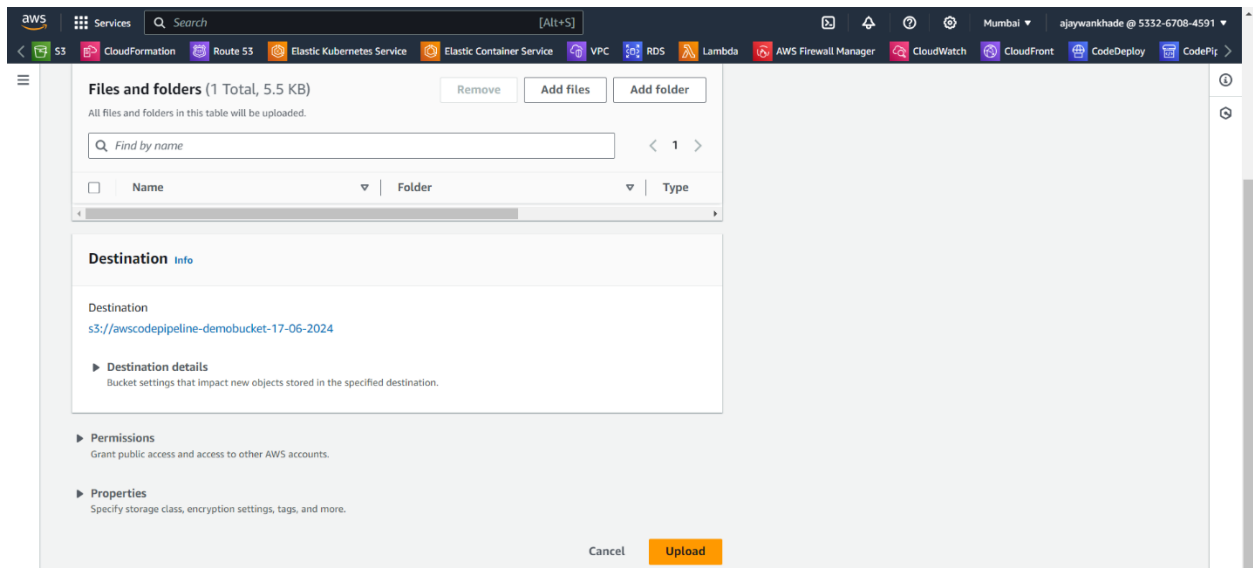
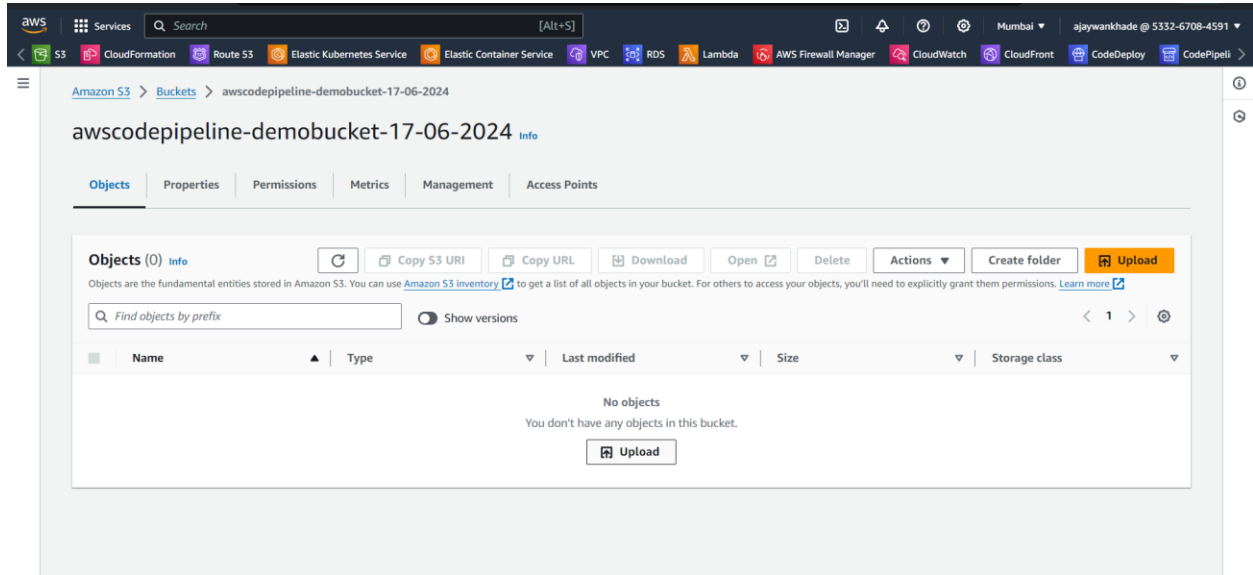
**To create an S3 bucket:**

1. Choose **Create bucket**.
2. In **Bucket name**, enter a name for your bucket (for example, **awscodepipeline-demobucket**.)



3. Next, download a sample and save it into a folder or directory on your local computer.
4. [https://docs.aws.amazon.com/codepipeline/latest/userguide/samples/SampleApp\\_Windows.zip](https://docs.aws.amazon.com/codepipeline/latest/userguide/samples/SampleApp_Windows.zip)

5. Download the compressed (zipped) file. Do not unzip the file.
6. In the Amazon S3 console, for your bucket, upload the file:
7. Choose Upload.
8. Drag and drop the file or choose Add files and browse for the file.
9. Choose Upload.



## Step 2: Create Amazon EC2 Windows instances and install the Code Deploy agent.

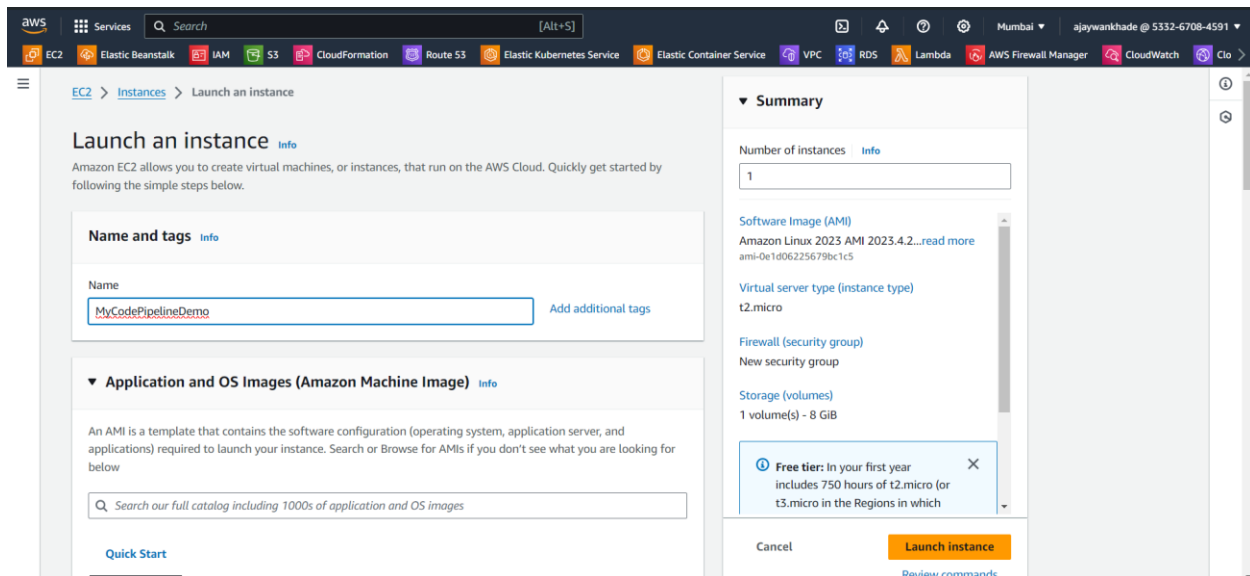
In this step, you create the Windows Server Amazon EC2 instances to which you will deploy a sample application. As part of this process, you create an instance role with policies that allow install and management of the Code Deploy agent on the instances. The Code Deploy agent is a software package that enables an instance to be used in Code Deploy deployments. You also attach policies that allow the instance to fetch files that the Code Deploy agent uses to deploy your application and to allow the instance to be managed by SSM.

### To create an instance role:

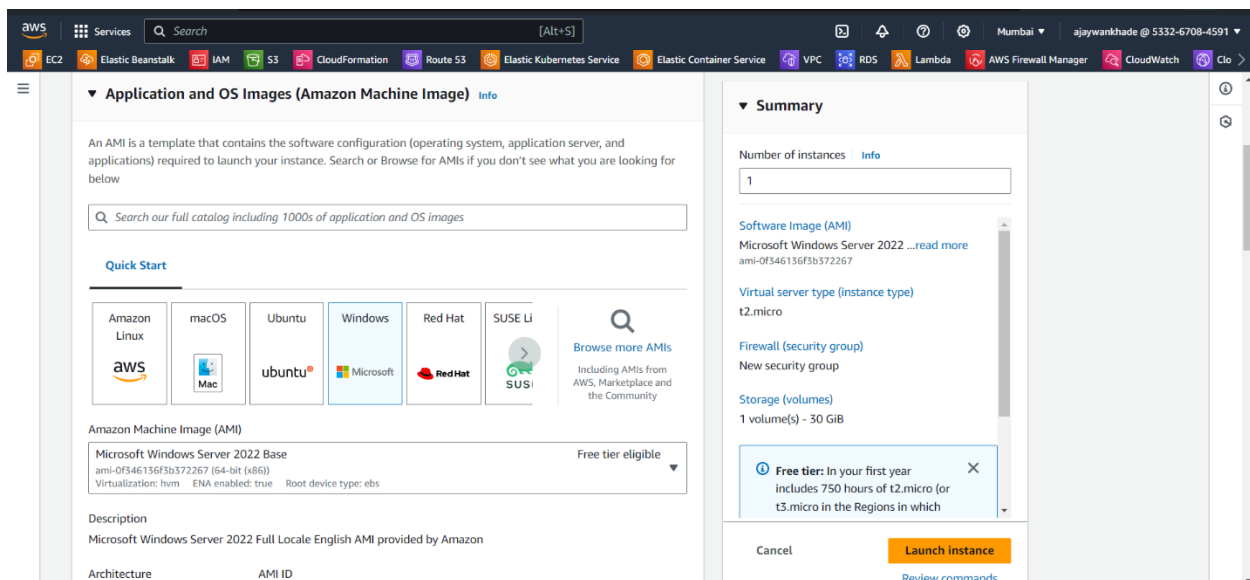
1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. From the console dashboard, choose **Roles**.
3. Choose **Create role**.
4. Under **Select type of trusted entity**, select **AWS service**. Under **Choose a use case**, select **EC2**, and then choose **Next: Permissions**.
5. Search for and select the policy named `AmazonEC2RoleforAWSCodeDeploy`.
6. Search for and select the policy named `AmazonSSMManagedInstanceCore`. Choose **Next: Tags**.
7. Choose **Next: Review**. Enter a name for the role (for example, `EC2InstanceRole`).
8. Choose **Create role**.

### To launch instances:

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. Under **Name and tags**, in **Name**, enter `MyCodePipelineDemo`. This assigns the instances a tag **Key** of `Name` and a tag **Value** of `MyCodePipelineDemo`. Later, you create a Code Deploy application that deploys the sample application to the instances. Code Deploy selects instances to deploy based on the tags



3. Under **Application and OS Images (Amazon Machine Image)**, choose the **Windows** option. (This AMI is described as the **Microsoft Windows Server 2019 Base** and is labeled "Free tier eligible" and can be found under **Quick Start**.)
4. Under **Instance type**, choose the free tier eligible t2. micro type as the hardware configuration for your instance.



- Under **Key pair (login)**, choose a key pair or create one.

You can also choose **Proceed without a key pair**.

**Key pair (login)** Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

Proceed without a key pair (Not recommended) Default value [Create new key pair](#)

For Windows instances, you use a key pair to decrypt the administrator password. You then use the decrypted password to connect to your instance.

**Network settings** Info [Edit](#)

Network Info

vpc-0884e37e9eb96e2d9

Subnet Info

No preference (Default subnet in any availability zone)

Auto-assign public IP Info

Enable

Additional charges apply when outside of free tier allowance

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.

**Summary**

Number of instances Info

1

Software Image (AMI)

Microsoft Windows Server 2022 ...[read more](#)  
ami-0f346136f3b372267

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 30 GiB

**Free tier:** In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which

Cancel [Launch instance](#) [Review commands](#)

- Under **Network settings**, do the following:

In **Auto-assign Public IP**, make sure the status is **Enable**.

- Next to **Assign a security group**, choose **Create a new security group**.
- In the row for **SSH**, under **Source type**, choose **My IP**.
- Choose **Add security group**, choose **HTTP**, and then under **Source type**, choose **My IP**.

**Network settings** Info [Edit](#)

Network Info

vpc-0884e37e9eb96e2d9

Subnet Info

No preference (Default subnet in any availability zone)

Auto-assign public IP Info

Enable

Additional charges apply when outside of free tier allowance

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-2' with the following rules:

☒ Allow RDP traffic from Anywhere (0.0.0.0/0)

☒ 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

**Summary**

Number of instances Info

1

Software Image (AMI)

Microsoft Windows Server 2022 ...[read more](#)  
ami-0f346136f3b372267

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 30 GiB

**Free tier:** In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which

Cancel [Launch instance](#) [Review commands](#)

- Expand **Advanced details**. In **IAM instance profile**, choose the IAM role you created in the previous procedure (for example, **EC2InstanceRole**).

The screenshot shows the AWS Management Console interface for creating an EC2 instance. The 'Advanced details' tab is selected, showing options for domain join directory, IAM instance profile (set to 'EC2InstanceRole'), hostname type (set to 'IP name'), DNS hostname (with checkboxes for IPv4 and IPv6), instance auto-recovery, and shutdown behavior (set to 'Stop'). The 'Summary' tab on the right displays the instance configuration: 1 instance, Microsoft Windows Server 2022 AMI, t2.micro instance type, new security group, and 1 30 GiB volume. A 'Free tier' notice is visible, indicating that the instance includes 750 hours of t2.micro (or t3.micro) in the first year. The 'Launch instance' button is highlighted in orange.

Choose **Launch instance**.

The screenshot shows the AWS Management Console interface for creating an EC2 instance. The 'Summary' tab is selected, displaying the instance configuration: 1 instance, Microsoft Windows Server 2022 AMI, t2.micro instance type, new security group, and 1 30 GiB volume. A 'Free tier' notice is visible, indicating that the instance includes 750 hours of t2.micro (or t3.micro) in the first year. The 'Launch instance' button is highlighted in orange.

## Step 3: Create an application in Code Deploy.

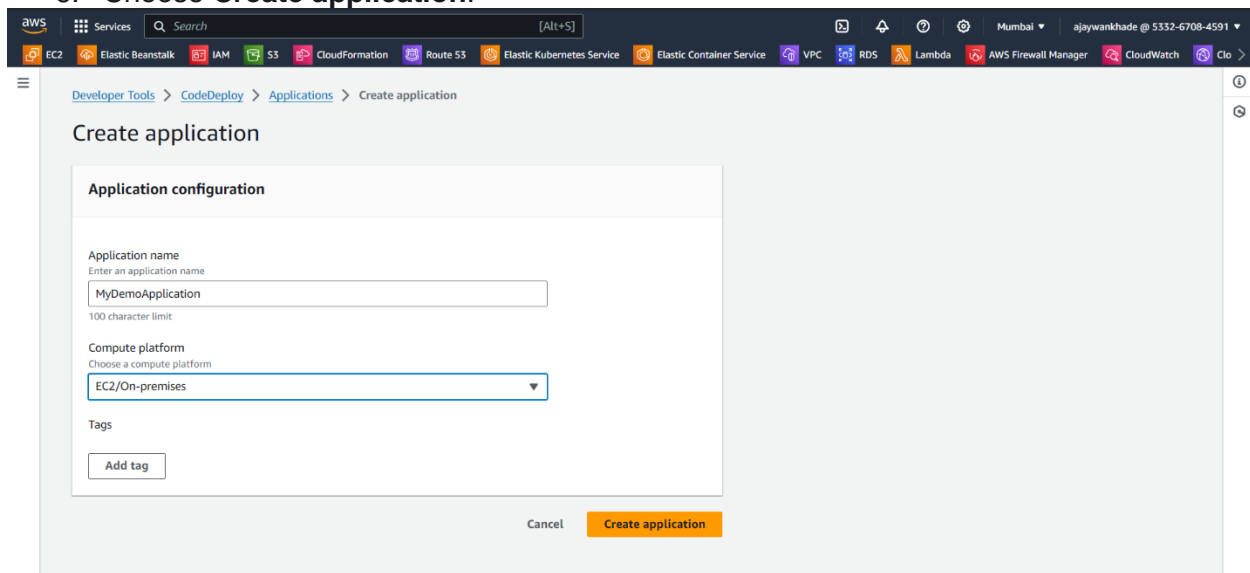
In Code Deploy, an application is an identifier, in the form of a name, for the code you want to deploy. Code Deploy uses this name to ensure the correct combination of revision, deployment configuration, and deployment group are referenced during a deployment. You select the name of the Code Deploy application you create in this step when you create your pipeline later in this tutorial.

### To create a Code, Deploy service role:

1. Choose **Create role**.
2. Under **Select trusted entity**, choose **AWS service**. Under **Use case**, choose **Code Deploy**. Choose **Code Deploy** from the options listed. Choose **Next**.  
The `AWSCodeDeployRole` managed policy is already attached to the role.
3. Choose **Next**.
4. Enter a name for the role (for example, `CodeDeployRole`), and then choose **Create role**.

### To create an application in Code Deploy

1. Open the Code Deploy console at <https://console.aws.amazon.com/codedeploy>.
2. If the **Applications** page does not appear, on the AWS Code Deploy menu, choose **Applications**.
3. Choose **Create application**.
4. In **Application name**, enter `MyDemoApplication`.
5. In **Compute Platform**, choose **EC2/On-premises**.
6. Choose **Create application**.



The screenshot shows the AWS CodeDeploy console interface. The top navigation bar includes the AWS logo, a search bar, and a list of services. The main content area is titled 'Create application' and contains a form for 'Application configuration'. The form has two main sections: 'Application name' and 'Compute platform'. The 'Application name' section has a text input field with the value 'MyDemoApplication' and a note '100 character limit'. The 'Compute platform' section has a dropdown menu with 'EC2/On-premises' selected. Below these sections is a 'Tags' section with an 'Add tag' button. At the bottom of the form are 'Cancel' and 'Create application' buttons.

Developer Tools > CodeDeploy > Applications > Create application

Create application

**Application configuration**

Application name  
Enter an application name  
MyDemoApplication  
100 character limit

Compute platform  
Choose a compute platform  
EC2/On-premises

Tags  
Add tag

Cancel Create application



## To create a deployment group in Code Deploy:

1. On the page that displays your application, choose **Create deployment group**.
2. In **Deployment group name**, enter **MyDemoDeploymentGroup**.

The screenshot shows the AWS CodeDeploy console. The breadcrumb navigation is: Developer Tools > CodeDeploy > Applications > MyDemoApplication > Create deployment group. The main heading is 'Create deployment group'. There are three sections: 'Application', 'Deployment group name', and 'Service role'. The 'Application' section shows 'MyDemoApplication' and 'EC2/On-premises'. The 'Deployment group name' section has a text input field containing 'MyDemoDeploymentGroup' with a '100 character limit' note. The 'Service role' section has a text input field that is currently empty.

3. In **Service role**, choose the service role you created earlier.
4. Under **Deployment type**, choose **In-place**.

The screenshot shows the same AWS CodeDeploy console, but now the 'Service role' and 'Deployment type' sections are visible. The 'Service role' section has a dropdown menu showing 'arn:aws:iam::533267084591:role/CodeDeployRoleNew'. The 'Deployment type' section has two options: 'In-place' (selected) and 'Blue/green'. The 'In-place' option is described as 'Updates the instances in the deployment group with the latest application revisions. During a deployment, each instance will be briefly taken offline for its update'. The 'Blue/green' option is described as 'Replaces the instances in the deployment group with new instances and deploys the latest application revision to them. After instances in the replacement environment are registered with a load balancer, instances from the original environment are deregistered and can be terminated'. Below these sections is the 'Environment configuration' section, which is currently empty.

5. Under **Environment configuration**, choose **Amazon EC2 Instances**. Choose **Name** in the **Key** field, and in the **Value** field, enter **MyCodePipelineDemo**.

The screenshot shows the AWS CodeDeploy console's 'Environment configuration' page. The page title is 'Environment configuration'. Below the title, there is a section 'Select any combination of Amazon EC2 Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add to this deployment'. There are two checkboxes: 'Amazon EC2 Auto Scaling groups' (unchecked) and 'Amazon EC2 instances' (checked). Below the 'Amazon EC2 instances' checkbox, it says '1 unique matched instance. [Click here for details](#)'. There is a note: 'You can add up to three groups of tags for EC2 instances to this deployment group. **One tag group:** Any instance identified by the tag group will be deployed to. **Multiple tag groups:** Only instances identified by all the tag groups will be deployed to.' Below this, there is a 'Tag group 1' section with a 'Key' field containing 'Name' and a 'Value - optional' field containing 'MyCodePipelineDemo'. There is a 'Remove tag' button. Below the tag group, there are buttons for 'Add tag' and '+ Add tag group'. There is also an 'On-premises instances' checkbox (unchecked). At the bottom, there is a 'Matching instances' section showing '1 unique matched instance. [Click here for details](#)'.

6. Under **Agent configuration with AWS Systems Manager**, choose **Now and schedule updates**. This installs the agent on the instance.

The screenshot shows the AWS CodeDeploy console's 'Agent configuration with AWS Systems Manager' page. The page title is 'Agent configuration with AWS Systems Manager'. Below the title, there is a blue box with an information icon and text: 'Complete the required prerequisites before AWS Systems Manager can install the CodeDeploy Agent. Make sure the AWS Systems Manager Agent is installed on all instances and attach the required IAM policies to them. [Learn more](#)'. Below this, there is a section 'Install AWS CodeDeploy Agent' with three radio buttons: 'Never' (unchecked), 'Only once' (unchecked), and 'Now and schedule updates' (checked). Below the radio buttons, there are two tabs: 'Basic scheduler' (selected) and 'Cron expression'. Below the 'Basic scheduler' tab, there is a text input field containing '14' and a dropdown menu set to 'Days'. Below this, there is a section 'Deployment settings' with a sub-section 'Deployment configuration'. It says 'Choose from a list of default and custom deployment configurations. A deployment configuration is a set of rules that determines how fast an application is deployed and the success or failure conditions for a deployment.' Below this, there is a dropdown menu set to 'CodeDeployDefault.AllAtOnce' and a button 'Create deployment configuration'.

7. Under **Deployment settings**, choose `CodeDeployDefault.OneAtATime`.
8. Under **Load Balancer**, make sure the **Enable load balancing** box is not selected.
9. In the **Advanced** section, leave the defaults.
10. Choose **Create deployment group**.

The screenshot shows the AWS CodeDeploy console interface for creating a deployment group. At the top, there's a navigation bar with the AWS logo, a search bar, and a list of services including EC2, Elastic Beanstalk, IAM, S3, CloudFormation, Route 53, Elastic Kubernetes Service, Elastic Container Service, VPC, RDS, Lambda, AWS Firewall Manager, CloudWatch, and CloudTrail. The main content area is divided into sections. The first section has a dropdown menu set to '14' and a 'Days' label. Below this is the 'Deployment settings' section, which includes a description of deployment configurations and a dropdown menu currently set to 'CodeDeployDefault.OneAtATime', with a 'Create deployment configuration' button next to it. The 'Load balancer' section follows, with a description and an unchecked checkbox labeled 'Enable load balancing'. At the bottom, there's a link to 'Advanced - optional' settings, and two buttons: 'Cancel' and 'Create deployment group'.

## Step 4: Create your first pipeline in Code Pipeline.

In this part of the tutorial, you create the pipeline. The sample runs automatically through the pipeline.

1. On the **Welcome** page, **Getting started** page, or the **Pipelines** page, choose **Create pipeline**.
2. In **Step 1: Choose pipeline settings**, in **Pipeline name**, enter **MyFirstPipeline**.

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1  
Choose pipeline settings

Step 2  
Add source stage

Step 3  
Add build stage

Step 4  
Add deploy stage

Step 5  
Review

### Choose pipeline settings

Step 1 of 5

#### Pipeline settings

**Pipeline name**  
Enter the pipeline name. You cannot edit the pipeline name after it is created.  
MyFirstPipeline  
No more than 100 characters

**Pipeline type**  
The pipeline type determines the pipeline structure and availability of parameters such as triggers. Pipeline type selection will impact features and pricing. [Which pipeline is right for me?](#)  
☐ V1 ☒ V2

**Execution mode**  
Choose the execution mode for your pipeline. This determines how the pipeline is run.  
☐ Superseded  
A more recent execution can overtake an older one. This is the default.  
☒ Queued (Pipeline type V2 required)  
Executions are processed one by one in the order that they are queued.  
☐ Parallel (Pipeline type V2 required)  
Executions don't wait for other runs to complete before starting or finishing.

3. In **Service role**, do one of the following:

- Choose **New service role** to allow Code Pipeline to create a new service role in IAM.
- Choose **Existing service role** to use a service role already created in IAM. In **Role name**, choose your service role from the list.

Service role

☒ New service role  
Create a service role in your account

☐ Existing service role  
Choose an existing service role from your account

**Role name**  
AWSCodePipelineServiceRole-ap-south-1-MyFirstPipeline  
Type your service role name  
☒ Allow AWS CodePipeline to create a service role so it can be used with this new pipeline

#### Variables

You can add variables at the pipeline level. You can choose to assign the value when you start the pipeline. Choosing this option requires pipeline type V2. [Learn more](#)

No variables defined at the pipeline level in this pipeline.

[Add variable](#)  
You can add up to 50 variables.

**The first pipeline execution will fail if variables have no default values.**

4. Leave the settings under **Advanced settings** at their defaults, and then choose **Next**.

No variables defined at the pipeline level in this pipeline.

**Add variable**

You can add up to 50 variables.

**Advanced settings**

**Artifact store**

☒ **Default location**  
Create a default S3 bucket in your account.

☐ **Custom location**  
Choose an existing S3 location from your account in the same region and account as your pipeline

**Encryption key**

☒ **Default AWS Managed Key**  
Use the AWS managed customer master key for CodePipeline in your account to encrypt the data in the artifact store.

☐ **Customer Managed Key**  
To encrypt the data in the artifact store under an AWS KMS customer-managed key, specify the key ID, key ARN, or alias ARN.

Cancel **Next**

5. In **Step 2: Add source stage**, in **Source provider**, choose **Amazon S3**. In **Bucket**, enter the name of the S3 bucket you created in [Step 1: Create an S3 bucket for your application](#). In **S3 object key**, enter the object key with or without a file path, and remember to include the file extension. For example, for `SampleApp_Windows.zip`, enter the sample file name as shown in this example:

**Add source stage** Info

Step 2 of 5

**Source**

**Source provider**  
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

**Amazon S3**

**Bucket**  
awscodepipeline-demobucket-17-06-2024

**S3 object key**  
SampleApp\_Windows (2).zip

Enter the object key. You can include a file path without the delimiter character (/) at the beginning. Include the file extension. Example: SampleApp.zip

**Change detection options**  
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.

☒ **Amazon CloudWatch Events (recommended)**  
Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs

☐ **AWS CodePipeline**  
Use AWS CodePipeline to check periodically for changes

Cancel Previous **Next**

- In **Step 3: Add build stage**, choose **Skip build stage**, and then accept the warning message by choosing **Skip** again. Choose **Next**.

- In **Step 4: Add deploy stage**, in **Deploy provider**, choose **Code Deploy**.

The **Region** field defaults to the same AWS Region as your pipeline. In **Application name**, enter MyDemoApplication, or choose the **Refresh** button, and then choose the application name from the list. In **Deployment group**, enter **MyDemoDeploymentGroup**, or choose it from the list, and then choose **Next**.

The screenshot shows the AWS CodeDeploy console in the 'Deploy' step configuration phase. The left sidebar indicates 'Step 4: Add deploy stage'. The main panel is titled 'Deploy' and contains the following fields:

- Deploy provider:** A dropdown menu with 'AWS CodeDeploy' selected.
- Region:** A dropdown menu with 'Asia Pacific (Mumbai)' selected.
- Application name:** A text input field containing 'MyDemoApplication'.
- Deployment group:** A text input field containing 'MyDemoDeploymentGroup'.
- Configure automatic rollback on stage failure:** An unchecked checkbox.

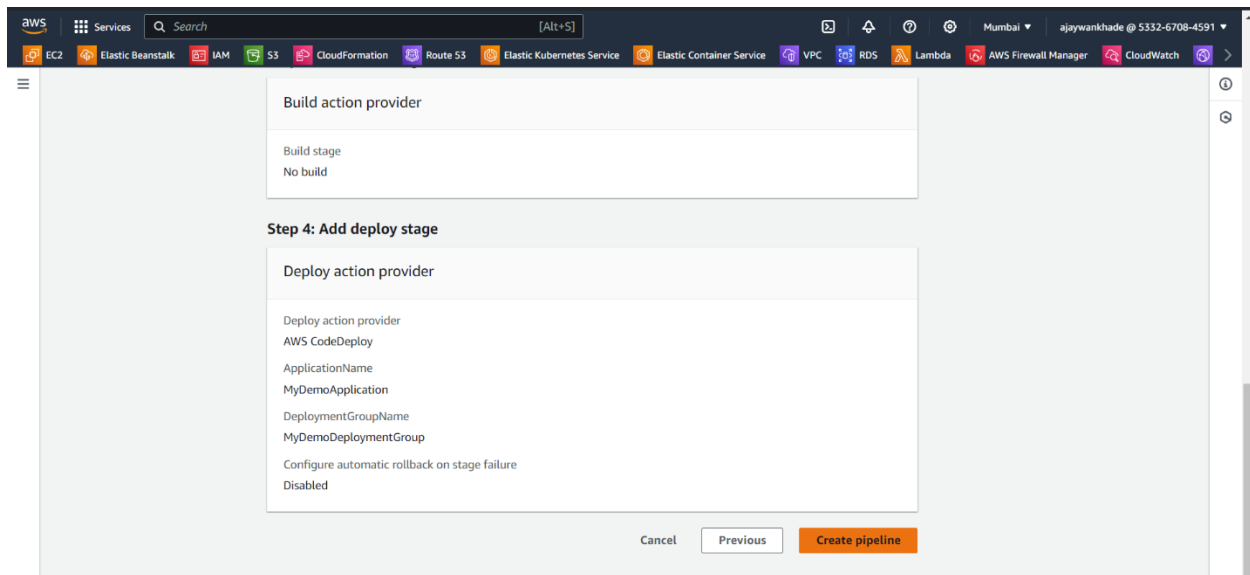
At the bottom right, there are three buttons: 'Cancel', 'Previous', and 'Next'.

- In **Step 5: Review**, review the information, and then choose **Create pipeline**.

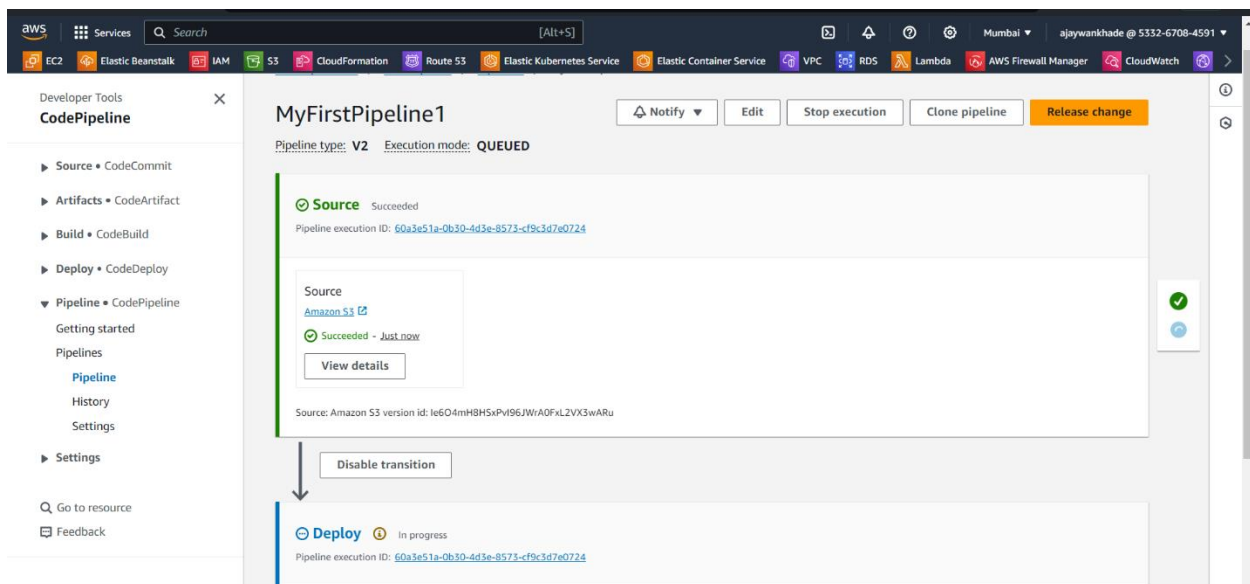
The screenshot shows the AWS CodePipeline console in the 'Review' step configuration phase. The left sidebar indicates 'Step 5: Review'. The main panel is titled 'Review' and contains the following information:

- Step 1: Choose pipeline settings**
- Pipeline settings:**
  - Pipeline name:** MyFirstPipeline
  - Pipeline type:** V2
  - Execution mode:** QUEUED
  - Artifact location:** A new Amazon S3 bucket will be created as the default artifact store for your pipeline
  - Service role name:** AWSCodePipelineServiceRole-ap-south-1-MyFirstPipeline1
- Variables:** (This section is partially visible at the bottom of the screenshot)

## 9. Choose **Create pipeline.**

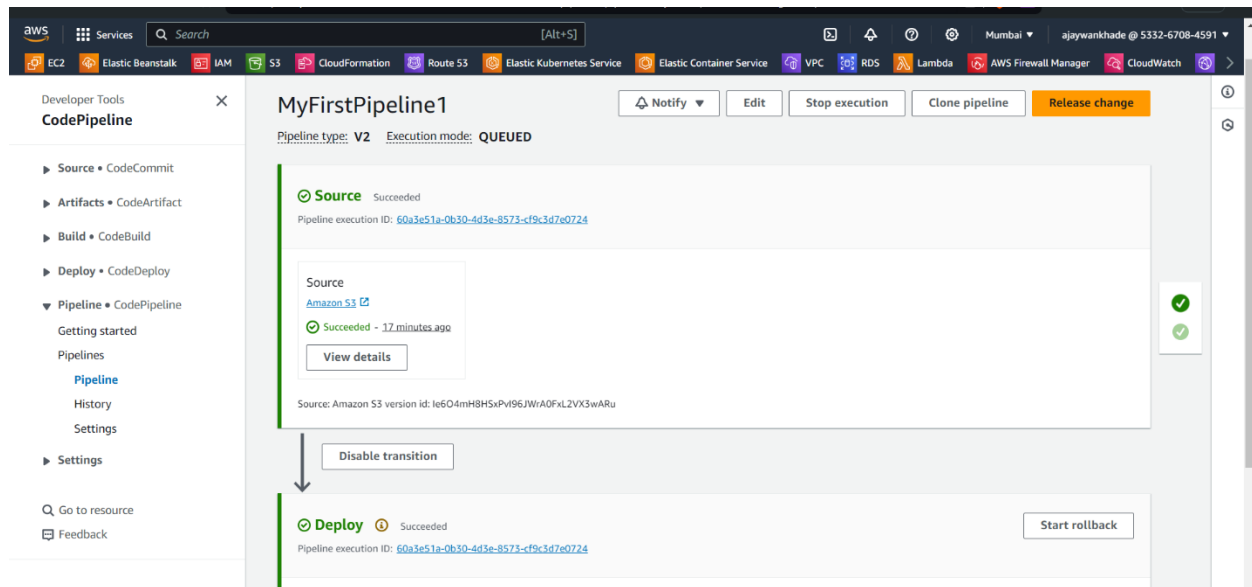


10. The pipeline starts to run. You can view progress and success and failure messages as the Code Pipeline sample deploys a webpage to each of the Amazon EC2 instances in the Code Deploy deployment.



11. View the initial progress of the pipeline. The status of each stage changes from **No executions yet** to **In Progress**, and then to either **Succeeded** or **Failed**. The pipeline should complete the first run within a few minutes.

12. After **Succeeded** is displayed for the action status, in the status area for the **Deploy** stage, choose **Details**. This opens the Code Deploy console.



### Step 6: To verify your pipeline ran successfully.

On the **Description** tab, in **Public DNS**, copy the address, and then paste it into the address bar of your web browser. View the index page for the sample application you uploaded to your S3 bucket.

