

Name: Nishant S Khetal

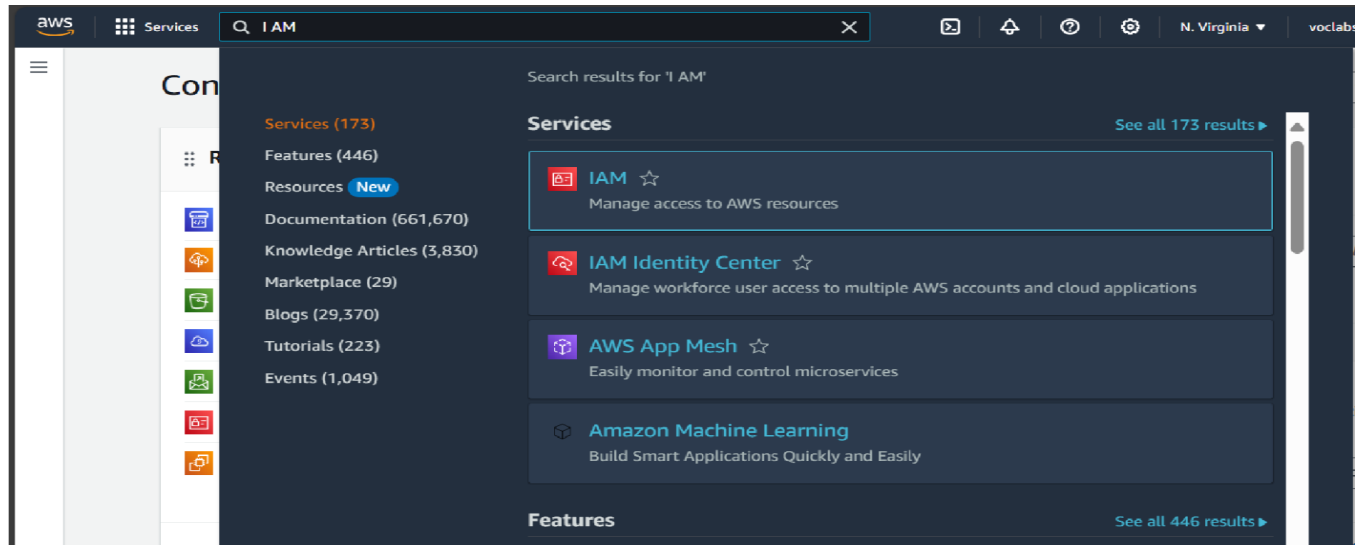
D15C

Roll No: 24

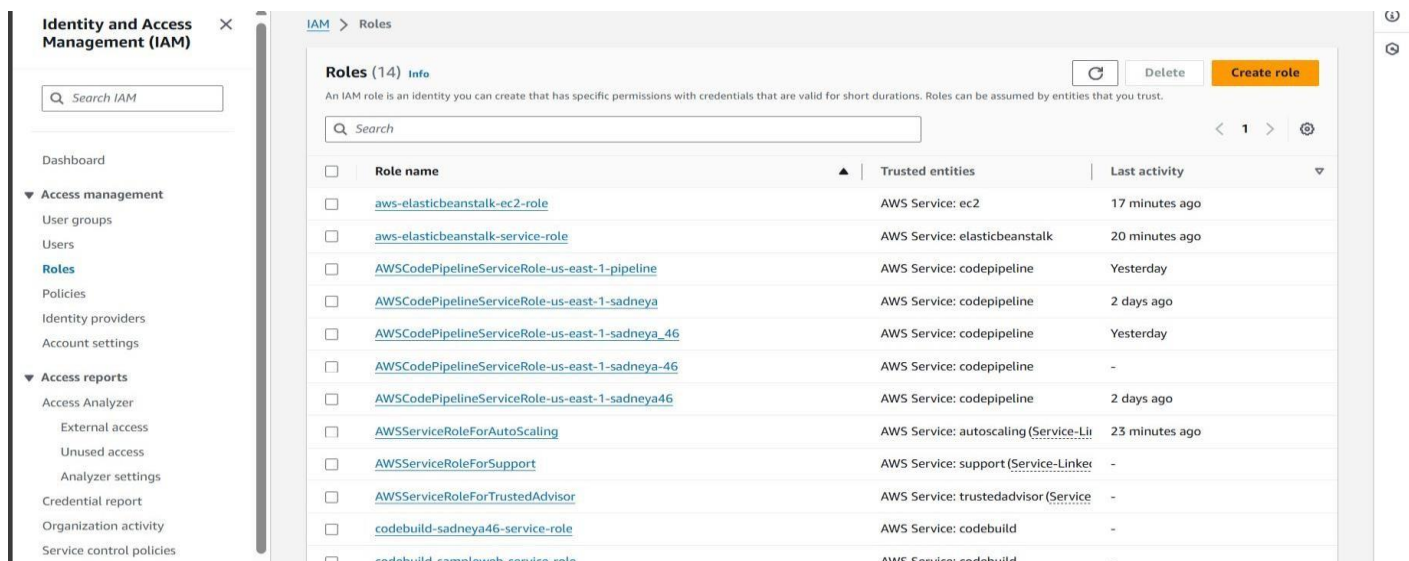
## Experiment No: 2

### Step1 :- Creation of role:

1. Login to your AWS account and search for IAM



2. Then go into the role section and click on create role.



3. Then select a trusted entity as AWS service.

The screenshot shows the 'Select trusted entity' step in the AWS IAM console. The left sidebar indicates the current step is 'Step 1: Select trusted entity'. The main content area is titled 'Select trusted entity' with an 'Info' link. Under the 'Trusted entity type' section, five options are listed: 'AWS service' (selected), 'AWS account', 'Web identity', 'SAML 2.0 federation', and 'Custom trust policy'. Each option includes a brief description of its function.

**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.
- ☐ **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.

4. Select use case as EC2.

The screenshot shows the 'Use case' step in the AWS IAM console. The left sidebar indicates the current step is 'Step 2: Add permissions'. The main content area is titled 'Use case' with a description: 'Allow an AWS service like EC2, Lambda, or others to perform actions in this account.' A dropdown menu for 'Service or use case' is set to 'EC2'. Below, a list of use cases is shown, with 'EC2' selected. Each use case includes a description of what it allows.

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

Service or use case: **EC2**

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.

5. Select permissions as AWS Elastic Beanstalk Web Tier and AWS elastic Beanstalk worker tier.

The screenshot shows the 'Permissions policy summary' and 'Add tags' steps in the AWS IAM console. The left sidebar indicates the current step is 'Step 3: Add tags'. The main content area is titled 'Permissions policy summary' and shows a table with two policies: 'AWSElasticBeanstalkWebTier' and 'AWSElasticBeanstalkWorkerTier', both of type 'AWS managed' and attached as 'Permissions policy'. Below the table, the 'Add tags' section is shown, indicating that no tags are currently associated with the resource and providing a button to 'Add new tag'.

**Permissions policy summary**

Policy name	Type	Attached as
<a href="#">AWSElasticBeanstalkWebTier</a>	AWS managed	Permissions policy
<a href="#">AWSElasticBeanstalkWorkerTier</a>	AWS managed	Permissions policy

**Step 3: Add tags**

**Add tags - optional** [Info](#)

Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

[Cancel](#) [Previous](#) [Create role](#)

6. Give a name to Role.

The screenshot shows the 'Create role' wizard in the AWS IAM console. The breadcrumb trail is 'IAM > Roles > Create role'. The left sidebar shows the progress: Step 1 'Select trusted entity', Step 2 'Add permissions', and Step 3 'Name, review, and create'. The main area is titled 'Name, review, and create' and contains a 'Role details' section. In the 'Role name' field, 'elasticbeanstalk-nishant' is entered. Below it, a description field contains 'Allows EC2 instances to call AWS services on your behalf.' At the bottom, there is a section for 'Step 1: Select trusted entities' with an 'Edit' button.

IAM > Roles > Create role

Step 1  
[Select trusted entity](#)

Step 2  
[Add permissions](#)

Step 3  
**Name, review, and create**

### Name, review, and create

**Role details**

**Role name**  
Enter a meaningful name to identify this role.  
  
Maximum 64 characters. Use alphanumeric and '+', '@', '\_' characters.

**Description**  
Add a short explanation for this role.  
  
Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: '+', '@', '/', '[', ']', '#', '%', '\*', ':', '-'

**Step 1: Select trusted entities** Edit

Trust policy

7. Then the role gets created

The screenshot shows the 'elasticbeanstalk-nishant' role page in the AWS IAM console. The breadcrumb trail is 'IAM > Roles > elasticbeanstalk-nishant'. The left sidebar shows the 'Identity and Access Management (IAM)' section with a search bar and a list of options: Dashboard, Access management (expanded), Users, Roles (selected), Policies, Identity providers, Account settings, Access reports, Access Analyzer, External access, and Unused access. The main area shows the role's summary, including its creation date (August 09, 2024, 09:33 UTC+05:30), ARN (arn:aws:iam::851725480355:role/aws-elasticbeanstalk-nishant), and Instance profile ARN (arn:aws:iam::851725480355:instance-profile/aws-elasticbeanstalk-nishant). Below the summary are tabs for Permissions, Trust relationships, Tags, Access Advisor, and Revoke sessions. The 'Permissions' tab is active, showing 'Permissions policies (2)' and buttons for 'Simulate', 'Remove', and 'Add permissions'.

Identity and Access Management (IAM)

Search IAM

Dashboard

▼ Access management

- User groups
- Users
- Roles**
- Policies
- Identity providers
- Account settings

▼ Access reports

- Access Analyzer
- External access
- Unused access

IAM > Roles > elasticbeanstalk-nishant

**elasticbeanstalk-nishant** Info Delete

Allows EC2 instances to call AWS services on your behalf.

**Summary** Edit

Creation date August 09, 2024, 09:33 (UTC+05:30)	ARN arn:aws:iam::851725480355:role/aws-elasticbeanstalk-nishant	Instance profile ARN arn:aws:iam::851725480355:instance-profile/aws-elasticbeanstalk-nishant
Last activity -	Maximum session duration 1 hour	

**Permissions** | Trust relationships | Tags | Access Advisor | Revoke sessions

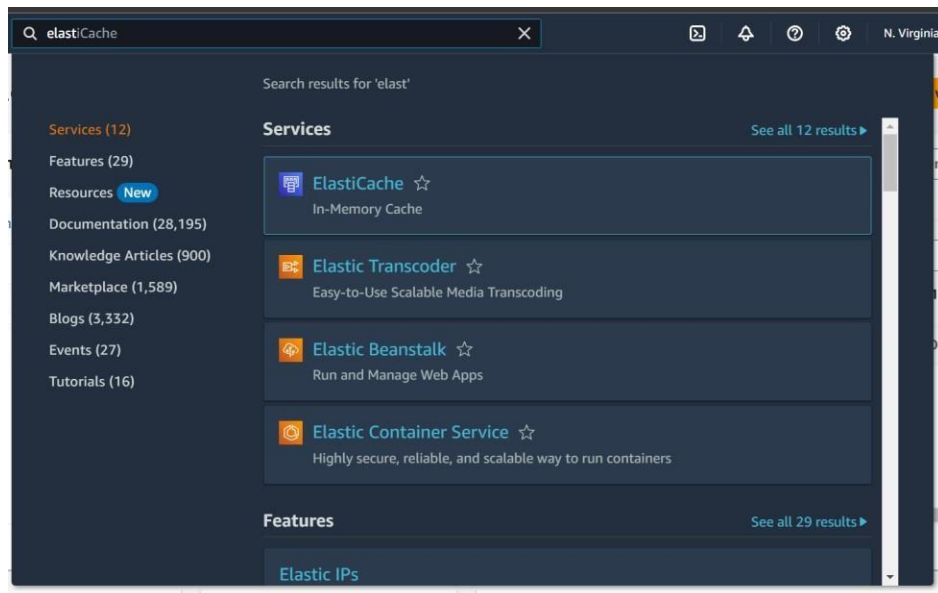
**Permissions policies (2)** Info ↺ Simulate Remove Add permissions ▼

You can attach up to 10 managed policies.

Trust policy

## Step 2 :- Creation Elastic Beanstalk Environment

1. search for Elastic Beanstalk in the search box.



2. Open up Elastic Beanstalk and name your web app.

**Application information** [Info](#)

**Application name**  
  
Maximum length of 100 characters.

**► Application tags (optional)**

**Environment information** [Info](#)  
Choose the name, subdomain and description for your environment. These cannot be changed later.

**Environment name**  
  
Must be from 4 to 40 characters in length. The name can contain only letters, numbers, and hyphens. It can't start or end with a hyphen. This name must be unique within a region in your account.

3. Select platform as PHP.

**Platform** [Info](#)

**Platform type**

- ☒ **Managed platform**  
Platforms published and maintained by Amazon Elastic Beanstalk. [Learn more](#)
- ☐ **Custom platform**  
Platforms created and owned by you. This option is unavailable if you have no platforms.

**Platform**

PHP

**Platform branch**

PHP 8.3 running on 64bit Amazon Linux 2023

**Platform version**

4.3.1 (Recommended)

4. After clicking on next you need to select the use existing role. Then you will see the existing role select it like here it is aws-elasticbeanstalk-service-role. Which we created in 1st part. Select role, then select key you have created then profile will be automatically selected according to role. then click on create application by keeping all the remaining settings as it is.

Step 3 - optional  
[Set up networking, database, and tags](#)

Step 4 - optional  
[Configure instance traffic and scaling](#)

Step 5 - optional  
[Configure updates, monitoring, and logging](#)

Step 6  
[Review](#)

**Service role**

☐ Create and use new service role

☒ **Use an existing service role**

**Existing service roles**

Choose an existing IAM role for Elastic Beanstalk to assume as a service role. The existing IAM role must have the required IAM managed policies.

aws-elasticbeanstalk-service-role

**EC2 key pair**

Select an EC2 key pair to securely log in to your EC2 instances. [Learn more](#)

key-linux

**EC2 instance profile**

Choose an IAM instance profile with managed policies that allow your EC2 instances to perform required operations.

aws-elasticbeanstalk-nishant

[View permission details](#)

[Cancel](#) [Skip to review](#) [Previous](#) [Next](#)

Keep Set up networking, database and tags, Configure instance traffic and scaling, Configure updates, monitoring and logging all these default.

5. Beanstalk creates a sample environment for you to deploy your application. By default, it creates an EC2 instance, a security group, an Auto Scaling group, an Amazon S3 Bucket, Amazon CloudWatch alarms and a domain name for your Application.

The screenshot displays the AWS Elastic Beanstalk console interface. At the top, a green banner indicates 'Environment successfully launched.' The left sidebar shows the 'Elastic Beanstalk' navigation menu with options like Applications, Environments, and Change history. The main content area is titled 'nishant-env' and includes an 'Info' link. Below the title, there are buttons for 'Actions' and 'Upload and deploy'. The 'Environment overview' section provides key details: Health is 'Ok', Environment ID is 'e-vw23gecggs', Domain is 'abhinav215-env.eba-6w7emmur.us-east-1.elasticbeanstalk.com', and Application name is 'Abhinav'. The 'Platform' section shows 'Node.js 20 running on 64bit Amazon Linux 2023/6.1.8' and a 'Supported' platform state. At the bottom, a horizontal menu allows switching between 'Events', 'Health', 'Logs', 'Monitoring', 'Alarms', 'Managed updates', and 'Tags'.

Environment overview	
Health Ok	Environment ID e-vw23gecggs
Domain abhinav215-env.eba-6w7emmur.us-east-1.elasticbeanstalk.com	Application name Abhinav

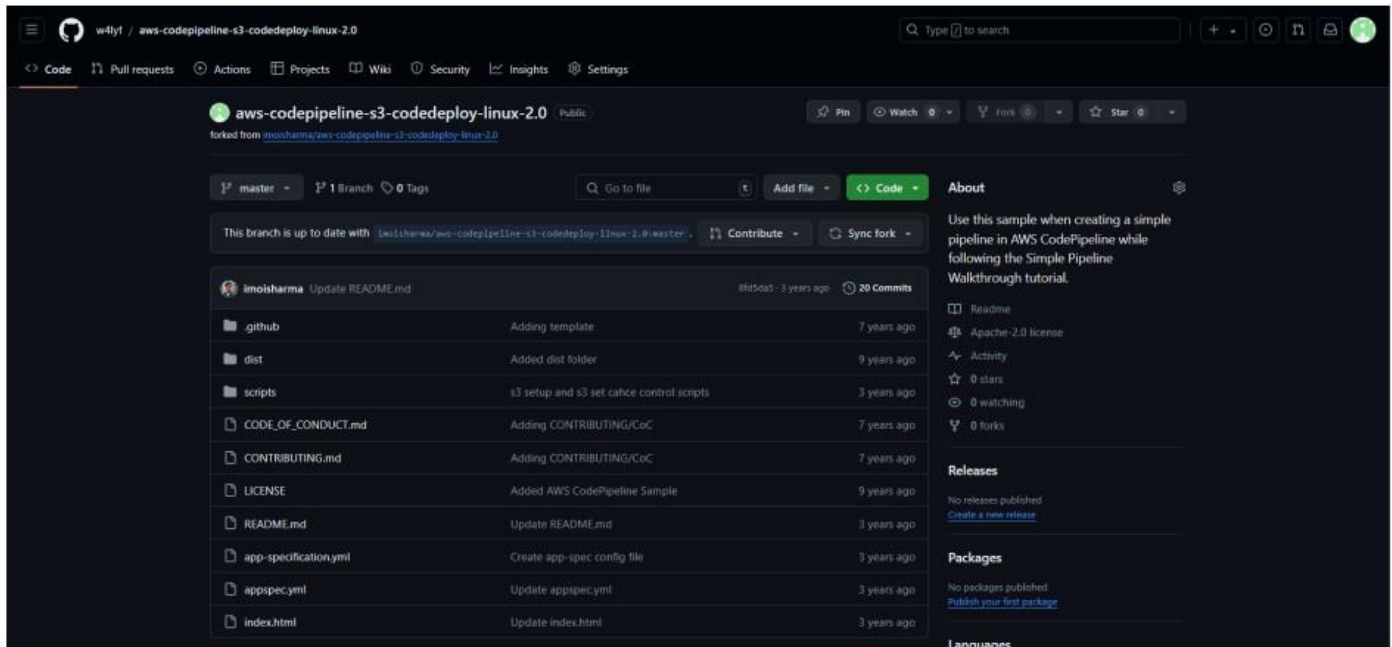
Platform
Platform Node.js 20 running on 64bit Amazon Linux 2023/6.1.8
Running version -
Platform state Supported

### Step 3: Get a copy of your sample code

In this step, we will get the sample code from [this](#) GitHub Repository to later host it. The pipeline takes code from the source and then performs actions on it.

For this experiment, as a source, we will use this forked GitHub repository. We can alternatively also use Amazon S3 and AWS CodeCommit.

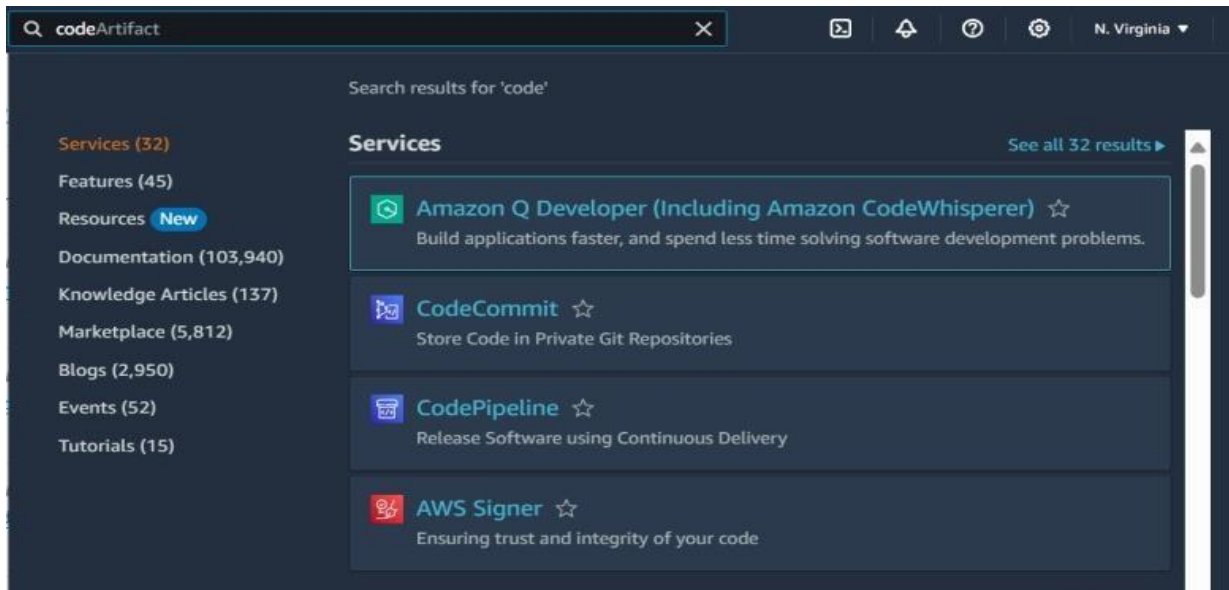
Go to the repository shared above and simply fork it.



## Step 4: Creating a CodePipeline

In this step, we'll create a simple pipeline that has its source and deployment information. In this case, however, we will skip the build stage where you get to plug in our preferred build provider.

1. Search CodePipeline in the search bar and click on create a new Pipeline.



2. Give a name to your pipeline.

**Pipeline settings**

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

**Pipeline type**  
*You can no longer create V1 pipelines through the console. We recommend you use the V2 pipeline type with improved release safety, pipeline triggers, parameterized pipelines, and a new billing model.*

**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.

**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**  
  
Type your service role name

☒ Allow AWS CodePipeline to create a service role so it can be used with this new pipeline



3. In the source stage, choose GitHub v2 as the provider, then connect your GitHub account to AWS by creating a connection. You'd need your GitHub credentials and then you'd need to authorize and install AWS on the forked GitHub Repository.

Developer Tools > Connections > Create connection

Beginning July 1, 2024, the console will create connections with codeconnections in the resource ARN. Resources with both service prefixes will continue to display in the console. [Learn more](#)

### Connect to GitHub

GitHub connection settings [Info](#)

Connection name

pipeline

GitHub Apps

GitHub Apps create a link for your connection with GitHub. Install a new app and save this connection.

53565526

or

Install a new app

Tags - optional

Connect

## Source

### Source provider

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

GitHub (Version 2)

#### New GitHub version 2 (app-based) action

To add a GitHub version 2 action in CodePipeline, you create a connection, which uses GitHub Apps to access your repository. Use the options below to choose an existing connection or create a new one. [Learn more](#)

### Connection

Choose an existing connection that you have already configured, or create a new one and then return to this task.

arn:aws:codeconnections:ap-south-1:860015268757:connection/47dc3241

 or 

Connect to GitHub

#### Ready to connect

Your GitHub connection is ready for use.

### Repository name

Choose a repository in your GitHub account.

w4lyf/aws-codepipeline-s3-codedeploy-linux-2.0

You can type or paste the group path to any project that the provided credentials can access. Use the format 'group/subgroup/project'.

### Default branch

Default branch will be used only when pipeline execution starts from a different source or manually started.

master

### Output artifact format

Choose the output artifact format.

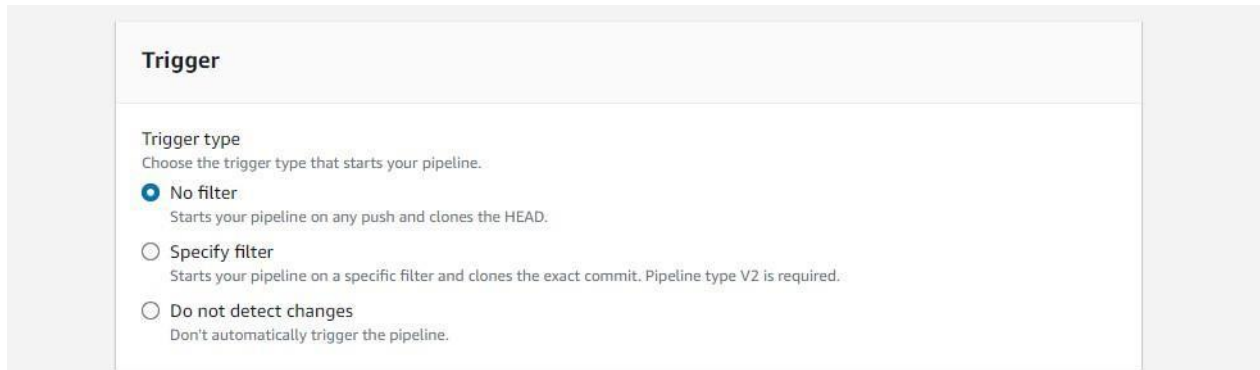
#### CodePipeline default

AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include Git metadata about the repository.

#### Full clone

AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full Git clone. Only supported for AWS CodeBuild actions.

4. Then select trigger type none.



The screenshot shows the 'Trigger' configuration section. It has a title 'Trigger' and a subtitle 'Trigger type'. Below the subtitle is the instruction 'Choose the trigger type that starts your pipeline.' There are three radio button options: 'No filter' (selected), 'Specify filter', and 'Do not detect changes'. Each option has a brief description below it.

**Trigger**

Trigger type  
Choose the trigger type that starts your pipeline.

☒ No filter  
Starts your pipeline on any push and clones the HEAD.

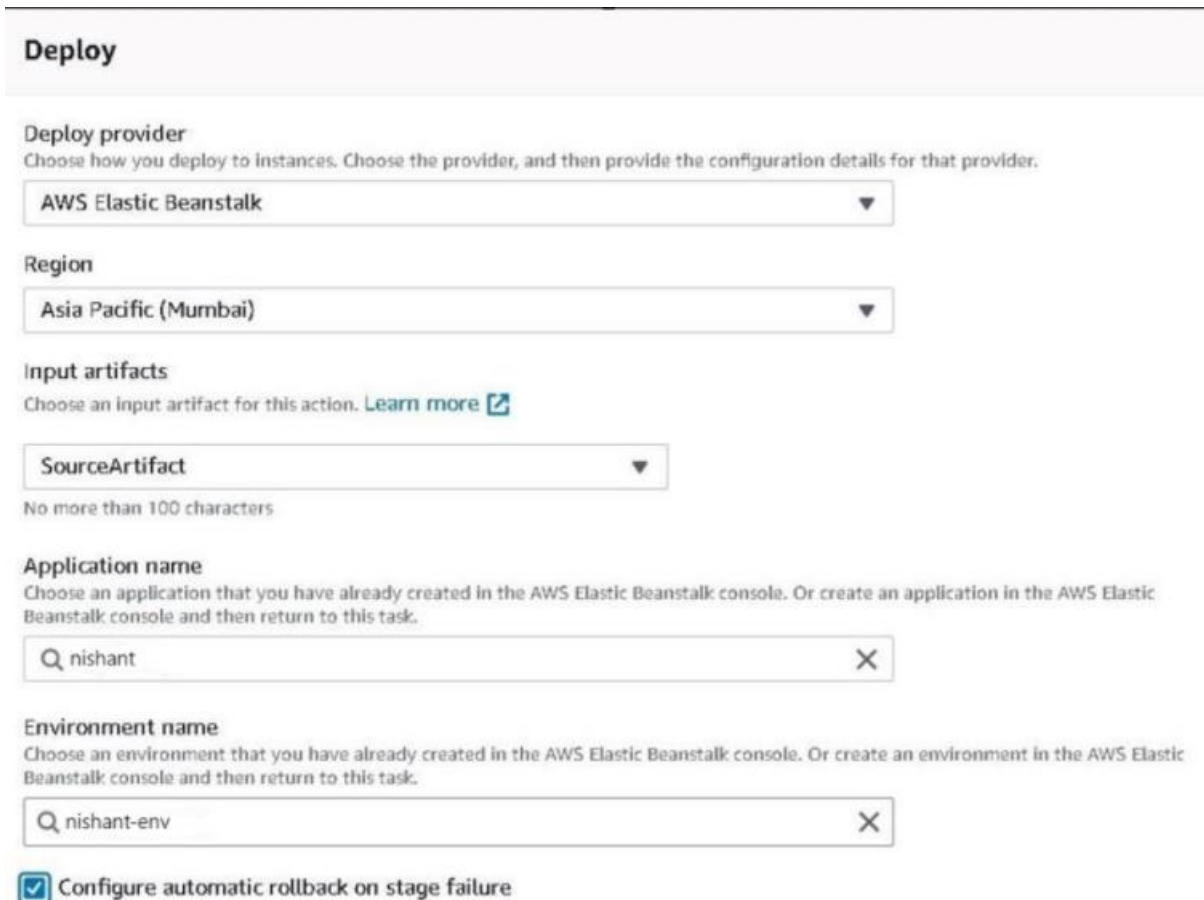
☐ Specify filter  
Starts your pipeline on a specific filter and clones the exact commit. Pipeline type V2 is required.

☐ Do not detect changes  
Don't automatically trigger the pipeline.

After that, click Continue and skip the build stage. Proceed to the Deployment stage.

### Step 5: Deployment

1. Choose Beanstalk as the Deploy Provider, same region as the Bucket and Beanstalk, name and environment name.



The screenshot shows the 'Deploy' configuration section. It has a title 'Deploy' and a subtitle 'Deploy provider'. Below the subtitle is the instruction 'Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.' There are three dropdown menus: 'AWS Elastic Beanstalk', 'Region' (set to 'Asia Pacific (Mumbai)'), and 'Input artifacts' (set to 'SourceArtifact'). Below the 'Input artifacts' dropdown is a note 'No more than 100 characters'. There are two text input fields: 'Application name' (set to 'nishant') and 'Environment name' (set to 'nishant-env'). At the bottom is a checkbox 'Configure automatic rollback on stage failure' which is checked.

**Deploy**

Deploy provider  
Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

AWS Elastic Beanstalk ▼

Region  
Asia Pacific (Mumbai) ▼

Input artifacts  
Choose an input artifact for this action. [Learn more](#)

SourceArtifact ▼

No more than 100 characters

Application name  
Choose an application that you have already created in the AWS Elastic Beanstalk console. Or create an application in the AWS Elastic Beanstalk console and then return to this task.

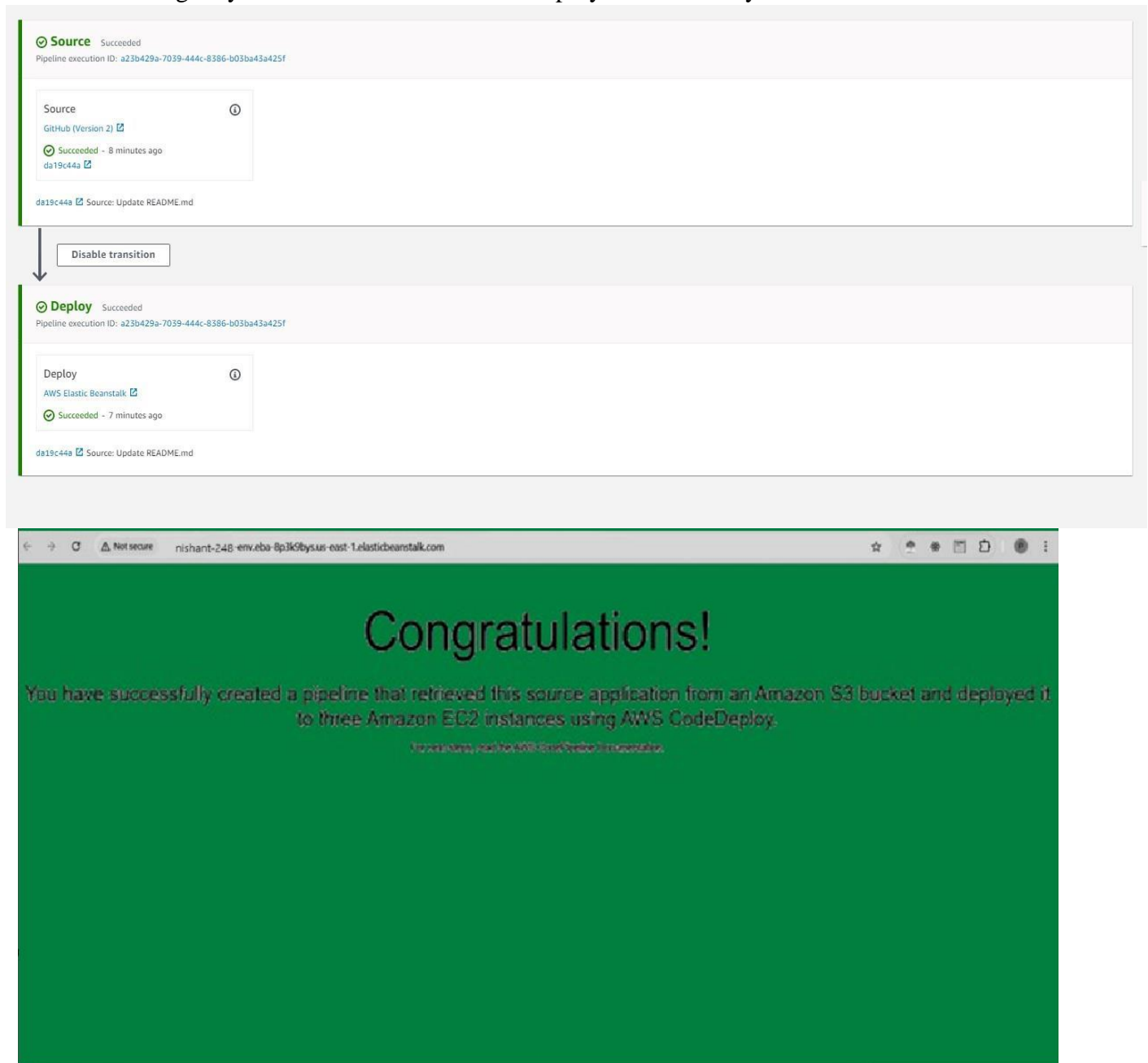
Q nishant X

Environment name  
Choose an environment that you have already created in the AWS Elastic Beanstalk console. Or create an environment in the AWS Elastic Beanstalk console and then return to this task.

Q nishant-env X

☒ Configure automatic rollback on stage failure

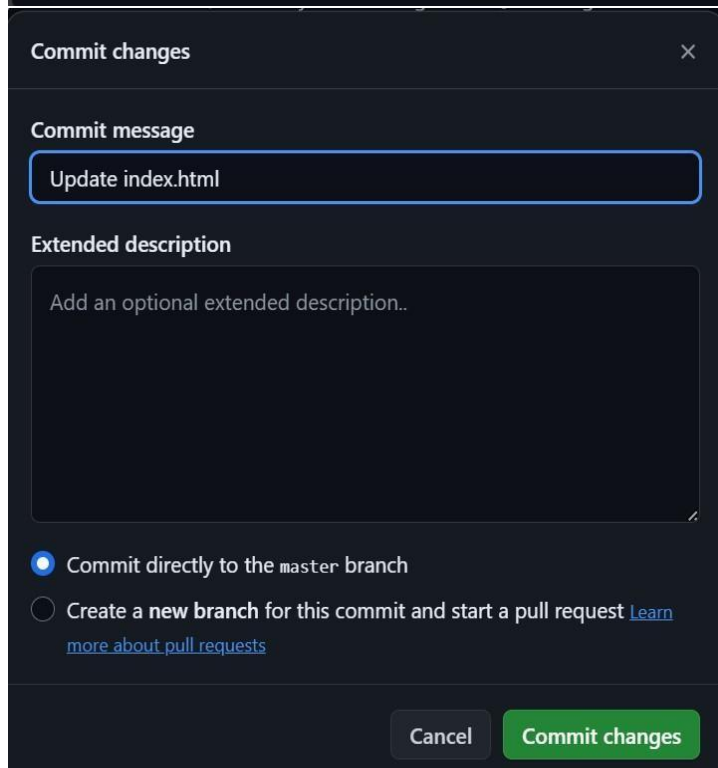
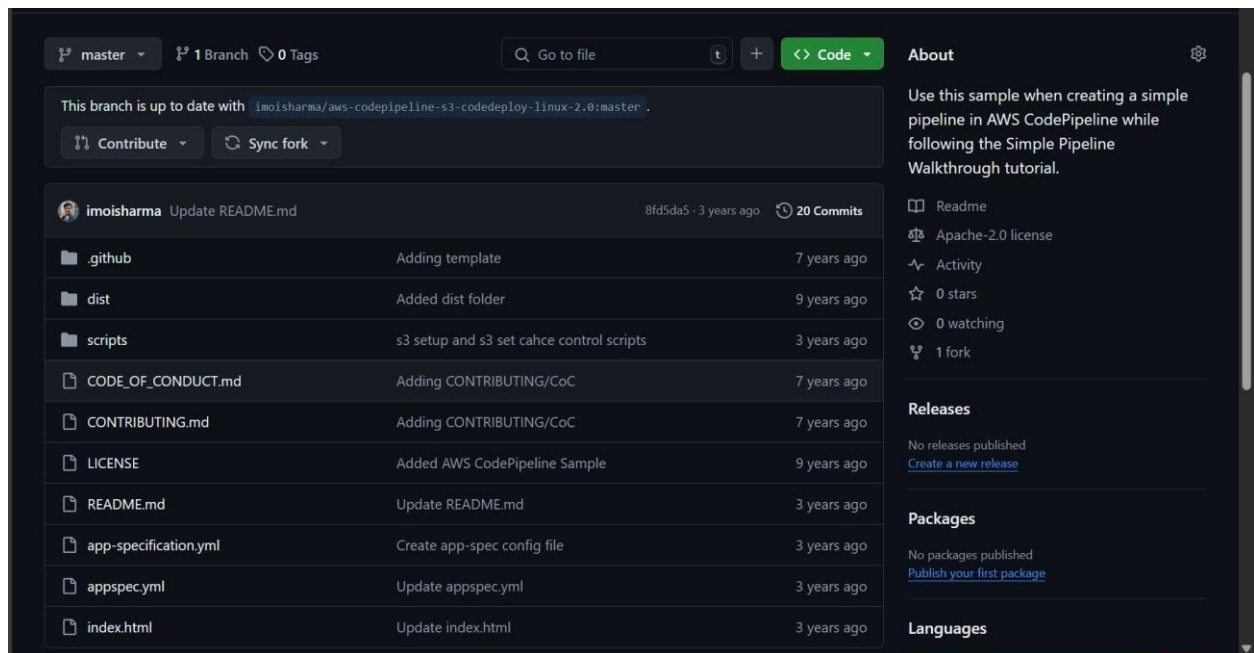
2. Then it will give you this result on screen. i.e. deployed successfully.



If you can see this, that means that you successfully created an automated software using CodePipeline.

### Step 6: Committing changes to update app

1. In this we make some changes in the file. Open github.com then open the forked repository. Then update the changes in the index.html file and finally commit those changes.



2. Then again start the deployment of the pipeline. And check the changes in the website

# Hiii Nishant Khetal

You have successfully created a pipeline that retrieved this source application from an Amazon S3 bucket and deployed it to three Amazon EC2 instances using AWS CodeDeploy.

For next steps, read the AWS CodePipeline Documentation.