

## **Problem Statement -**

Setup a CI/CD pipeline using the tools your choice(or preferably the mentioned tools).

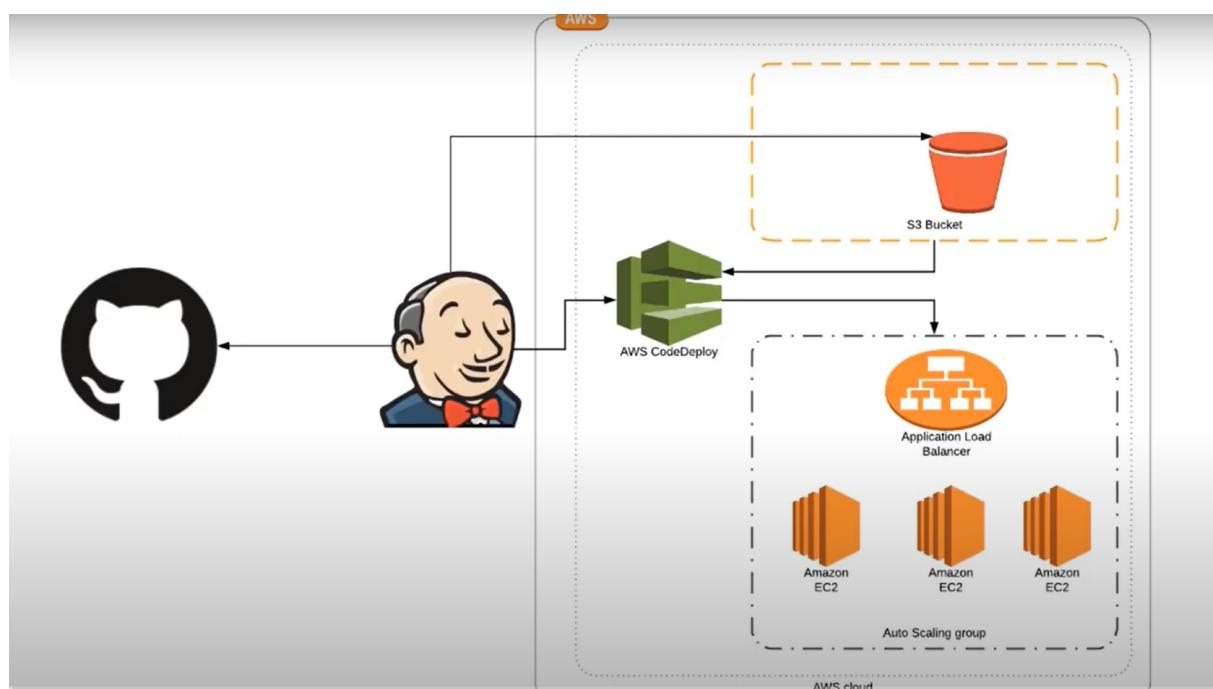
1. It should deploy a simple web application to a server on a code push to a repository.
2. The deployed web application should be reachable on any web browser.
3. Make it scalable such that when load increases the number of servers scale up and down making sure the new servers have the updated code.

## **Additional requirements**

1. Setup to be done using AWS, Jenkins, CodeDeploy
2. Jenkins should not be on the same server as the application being deployed to

**Tools** 1. Jenkins 2. Git/Bitbucket 3. AWS EC2 4. AWS CodeDeploy

## **Solution -**



# Step 1 - Create Two IAM roles as -

1. In first provide `AmazonEC2RoleforAWSCodeDeploy` and `AmazonS3FullAccess`
2. In second provide `codedeployrole` role.

This roles will be used further.

1.

The screenshot shows the AWS IAM console with the 'ec2codedeploy' role selected. The left sidebar shows the 'Roles' section. The main panel displays the role's ARN, description, instance profile ARNs, path, creation time, last activity, and maximum session duration. The 'Permissions' tab is active, showing two attached policies: 'AmazonEC2RoleforAWSCodeDeploy' and 'AmazonS3FullAccess', both of which are AWS managed policies. Other tabs include 'Trust relationships', 'Tags (1)', 'Access Advisor', and 'Revoke sessions'. The status bar at the bottom indicates the user is shashank Jain, it's 17:33, and the date is 09-01-2022.

2.

The screenshot shows the AWS IAM console with the 'codedeplyrole' role selected. The left sidebar shows the 'Roles' section. The main panel displays the role's ARN, description, instance profile ARNs, path, creation time, last activity, and maximum session duration. The 'Permissions' tab is active, showing one attached policy: 'AWSCodeDeployRole', which is an AWS managed policy. Other tabs include 'Trust relationships', 'Tags (1)', 'Access Advisor', and 'Revoke sessions'. The status bar at the bottom indicates the user is shashank Jain, it's 17:32, and the date is 09-01-2022.

## Step 2 - Create 4 ec2 instances and put user data in it -

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace, or you can select one of your own AMIs.

Search for an AMI by entering a search term e.g. "Windows"

Cancel and Exit

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Feedback English (US) ▾

© 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences

1640586101195.pdf

Type here to search

15°C Haze 17:36 ENG 09-01-2022

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance families Current generation Show/Hide Columns

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, ~ 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	<b>t2.micro</b> Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
<input type="checkbox"/>	t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
<input type="checkbox"/>	t3	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes
<input type="checkbox"/>	t3	t3.micro	2	1	EBS only	Yes	Up to 5 Gigabit	Yes

Cancel Previous Review and Launch Next: Configure Instance Details

Feedback English (US) ▾

© 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences

1640586101195.pdf

Type here to search

15°C Haze 17:37 ENG 09-01-2022

Screenshot of the AWS Lambda Launch Instance Wizard - Step 3: Configure Instance Details. The page shows configuration options for a new EC2 instance, including tenancy (Shared), credit specification (Unlimited), and advanced details like user data (a shell script to install dependencies). Buttons at the bottom include 'Cancel', 'Previous', 'Review and Launch' (highlighted in blue), and 'Next: Add Storage'.

Screenshot of the AWS Lambda Launch Instance Wizard - Step 4: Review and Launch. The page displays the summary of the instance configuration, including the instance type (t2.micro), availability zone (ap-south-1a), and public DNS (ec2-35-154-3-124.ap-south-1.compute.amazonaws.com). A 'Launch Instance' button is prominently displayed at the bottom.

**Step 6: Configure Security Group**

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group:  Create a new security group  Select an existing security group

Security Group ID	Name	Description	Actions
sg-0385a21edc490ea4c	code_deploy_sg	launch-wizard-9 created 2022-01-07T23:40:46.978+05:30	<a href="#">Copy to new</a>
sg-00dea362	default	default VPC security group	<a href="#">Copy to new</a>
sg-01c95b8f1907c391b	launch-wizard-1	launch-wizard-1 created 2021-09-16T09:33:34.450+05:30	<a href="#">Copy to new</a>
sg-092c38380f799b6ab4	launch-wizard-10	launch-wizard-10 created 2022-01-09T14:23:00.524+05:30	<a href="#">Copy to new</a>
sg-054ffebab6005c6bc	launch-wizard-2	launch-wizard-2 created 2021-09-17T00:51:06.924+05:30	<a href="#">Copy to new</a>
sg-0118a09881e75d17	launch-wizard-3	launch-wizard-3 created 2021-09-27T15:40:09.874+05:30	<a href="#">Copy to new</a>
sg-0543f29d470caeaa39	launch-wizard-4	launch-wizard-4 created 2021-09-27T15:41:52.090+05:30	<a href="#">Copy to new</a>
sg-0385a21edc490ea4c	launch-wizard-5	launch-wizard-5 created 2021-01-07T23:40:46.978+05:30	<a href="#">Copy to new</a>

Inbound rules for sg-0385a21edc490ea4c (Selected security groups: sg-0385a21edc490ea4c)

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	0.0.0.0/0	
HTTP	TCP	80	0.0.0.0/0	
SSH	TCP	22	0.0.0.0/0	

[Cancel](#) [Previous](#) [Review and Launch](#)

Feedback English (US) ▾ © 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences

1640586101195.pdf Show all X

Type here to search 15°C Haze 17:37 09-01-2022 ENG

## User data i have used -

```
#!/bin/bash
sudo yum -y update
sudo yum -y install ruby
sudo yum -y install wget
cd /home/ec2-user
Wget
https://aws-codedeploy-ap-south-1.s3.ap-south-1.amazonaws.com/latest/install
sudo chmod +x ./install
sudo ./install auto
sudo yum install -y python-pip
sudo pip install awscli
```

- This user data includes installation of some software like aws codedeploy , aws cli etc.
- add first IAM role to the instances .
- Create Security Group and provide http 80 and ssh 22 rules.
- Than create instances.

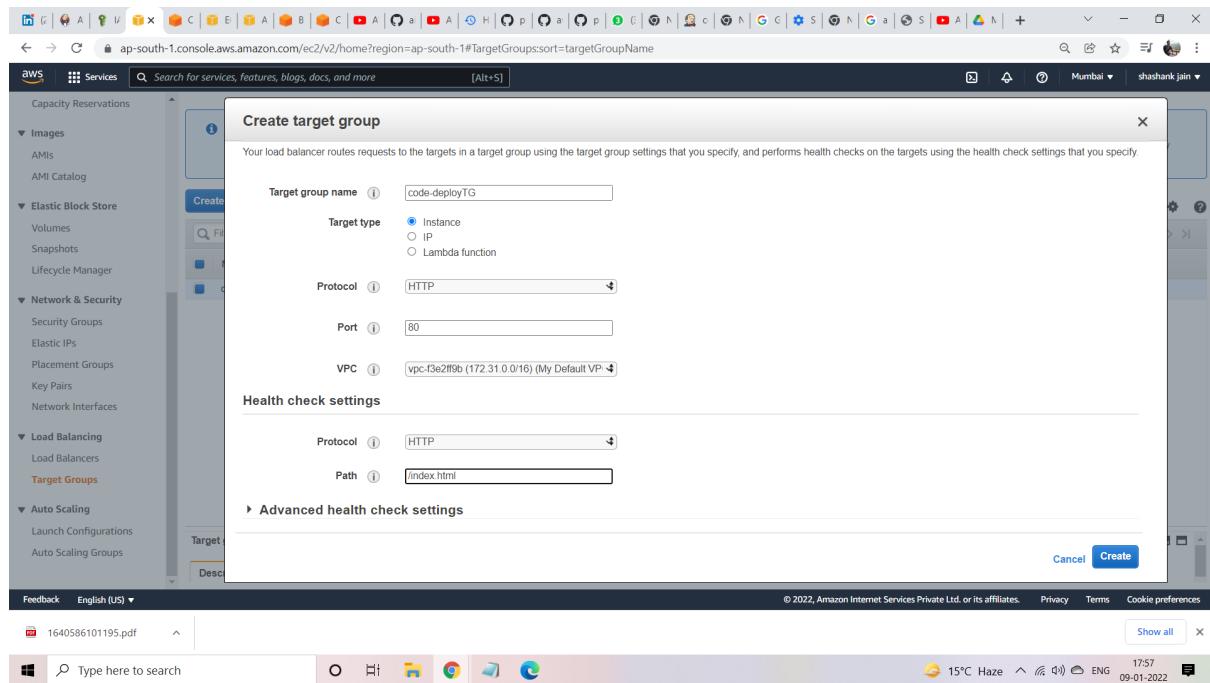
# Step 3 - Create an of any of the instance which we will use in AutoScalingGroup.

The screenshot shows the AWS EC2 Instances page. A context menu is open over an instance named 'ec2-code\_depl'. The 'Create Image' option is highlighted in the menu. The main table lists several instances, including 'ec2-code\_depl' (running), 'i-0760aa95f159' (running), 'i-096560ae9db' (pending), 'i-0af9f9dca96' (pending), and 'Jenkins\_server' (running). The instance details for 'ec2-code\_depl' are shown in a modal, including its Public DNS and private IP addresses.

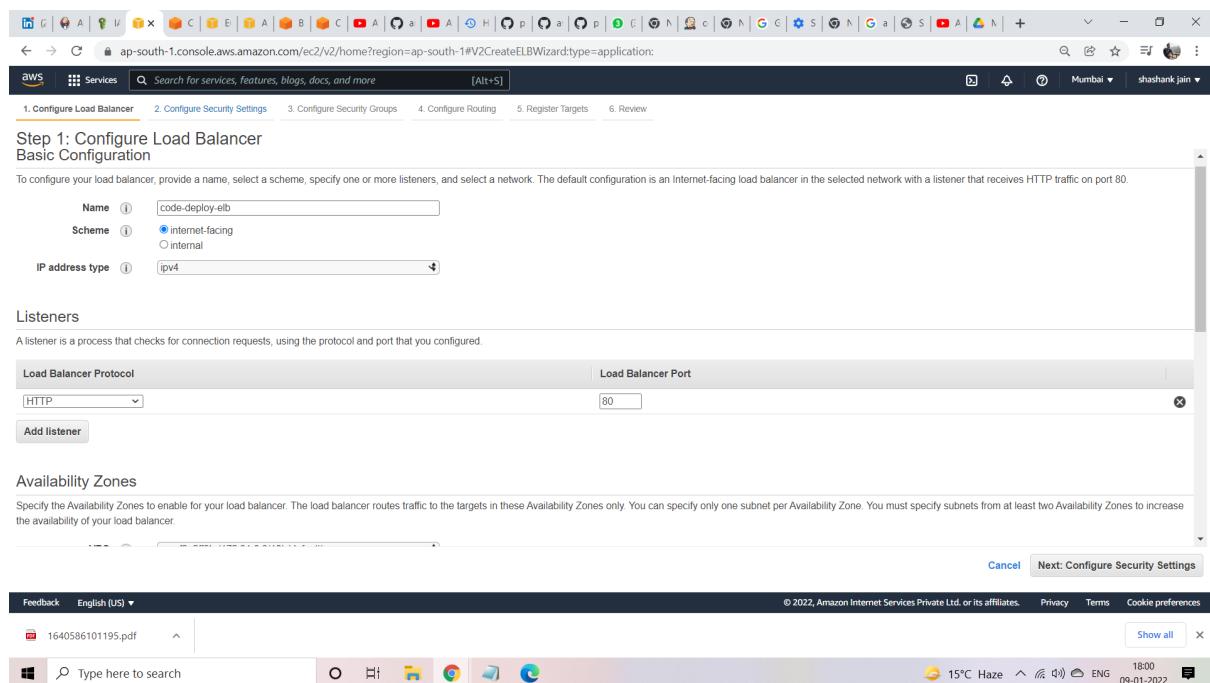
The screenshot shows the 'Create Image' dialog box. The 'Image name' field is set to 'code\_deploy\_image'. The 'Instance Volumes' section shows a single volume configuration for the 'Root' volume, which is a snapshot of '/dev/xvda' with a size of 8 GiB and a volume type of 'General Purpose SSD (gp2)'. The 'Create Image' button is visible at the bottom right of the dialog.

## Step 4 - Create Application Load Balancer.

1. Create target group than load balancer , in target group give path to /index.html , which is the app file containing code over git.



## Now Create Application Load Balancer -



**Step 3: Configure Security Groups**

A security group is a set of firewall rules that control the traffic to your load balancer. On this page, you can add rules to allow specific traffic to reach your load balancer. First, decide whether to create a new security group or select an existing one.

**Assign a security group**

Create a new security group  
 Select an existing security group

**Security group name**: code-deploy-sg  
**Description**: load-balancer-wizard-1 created on 2022-01-09T18:03:02+05:30

Type	Protocol	Port Range	Source
HTTP	TCP	80	Custom 0.0.0.0/:/0

**Add Rule**

**Cancel** **Previous** **Next: Configure Routing**

**Feedback** English (US) ▾ **© 2022, Amazon Internet Services Private Ltd. or its affiliates.** **Privacy** **Terms** **Cookie preferences**

1640586101195.pdf

Type here to search

15°C Haze 18:04 ENG 09-01-2022

**Step 4: Configure Routing**

on this load balancer. You can edit or add listeners after the load balancer is created.

**Target group**

**Target group**: New target group  
**Name**: code-deploy-TG  
**Target type**: Instance  
**Protocol**: HTTP  
**Port**: 80  
**Protocol version**:  
 **HTTP1**: Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.  
 **HTTP2**: Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.  
 **gRPC**: Send requests to targets using gRPC. Supported when the request protocol is gRPC.

**Health checks**

**Cancel** **Previous** **Next: Register Targets**

**Feedback** English (US) ▾ **© 2022, Amazon Internet Services Private Ltd. or its affiliates.** **Privacy** **Terms** **Cookie preferences**

1640586101195.pdf

Type here to search

15°C Haze 18:05 ENG 09-01-2022

**Step 5: Register Targets**  
Register targets with your target group. If you register a target in an enabled Availability Zone, the load balancer starts routing requests to the targets as soon as the registration process completes and the target passes the initial health checks.

**Registered targets**

To deregister instances, select one or more registered instances and then click Remove.

Instance	Name	Port	State	Security groups	Zone
No instances available.					

**Instances**

To register additional instances, select one or more running instances, specify a port, and then click Add. The default port is the port specified for the target group. If the instance is already registered on the specified port, you must specify a different port.

Add to registered on port 80

Instance	Name	State	Security groups	Zone	Subnet ID	Subnet CIDR
i-0ad9f6dcb28ddca5	ec2-code_deploy	running	launch-wizard-10	ap-south-1a	subnet-59f0ca31	172.31.32.0/20
i-098560ae9dbc1117	ec2-code_deploy	running	launch-wizard-10	ap-south-1a	subnet-59f0ca31	172.31.32.0/20
i-0760aa5f156002bd	ec2-code_deploy	running	launch-wizard-10	ap-south-1a	subnet-59f0ca31	172.31.32.0/20
i-0059c5e7f8315249	ec2-code_deploy	running	launch-wizard-10	ap-south-1a	subnet-59f0ca31	172.31.32.0/20
i-0fe8dc7ded70c8812	Code_deploy_ASG	running	AutoScaling-Security-Gr...	ap-south-1a	subnet-59f0ca31	172.31.32.0/20

Cancel Previous Next: Review

Feedback English (US) ▾ Show all 15°C Haze 18:06 ENG 09-01-2022

We have added the targets to already created instances.

**Step 6: Review**  
Please review the load balancer details before continuing

**Load balancer**

Name codedeploylb1  
Scheme internet-facing  
Listeners Port:80 - Protocol:HTTP  
IP address type ipv4  
VPC vpc-f3e2ff9b  
Subnets subnet-59f0ca31, subnet-e76e05ab, subnet-53f94428  
Tags

**Security groups**

Security groups code-deploy-sg

**Routing**

Target group New target group code-deploy-TG  
Target group name code-deploy-TG  
Port 80  
Target type instance  
Protocol HTTP  
Protocol version HTTP1  
Health check protocol HTTP  
Path /index.html  
Health check port traffic port  
Healthy threshold 5

Cancel Previous Create

Feedback English (US) ▾ Show all 15°C Haze 18:06 ENG 09-01-2022

# Step 5 - Create Launch Configurations and than create Auto scaling group.

The screenshot shows the 'Create launch configuration' page in the AWS EC2 console. The 'Launch configuration name' field contains 'code\_deploy\_LC'. The 'Amazon machine image (AMI)' dropdown is set to 'code\_deploy\_image'. The 'Instance type' dropdown is set to 't2.micro'. The status bar at the bottom indicates 'Feedback English (US) ▾' and '1640586101195.pdf'.

The screenshot continues the 'Create launch configuration' process. It shows the 'Advanced details' section with a note about editing existing launch configurations. Below it, the 'Storage (volumes)' section lists an 'EBS volume' for the 'Root' device, which is a 8 GiB General purpose SSD volume. The status bar at the bottom indicates 'Feedback English (US) ▾' and '1640586101195.pdf'.

Assign a security group

Create a new security group

Select an existing security group

Security group name

AutoScaling-Security-Group\_SG

Description

AutoScaling-Security-Group-1 (2022-01-09T11:19:43.658Z)

Rules

Type	Protocol	Port range	Source type	Source
SSH	TCP	22	Anywhere	0.0.0.0/0
HTTP	TCP	80	Anywhere	0.0.0.0/0
Custom TCP rule	TCP	0	Custom IP	

+ Add new rule

⚠ 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.

Feedback English (US) ▾

© 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences

1640586101195.pdf Show all

Type here to search 15°C Haze 17:50 ENG 09-01-2022

Launch Configuration created.  
Now create AutoScalingGroup

Step 1  
Choose launch template or configuration

Step 2  
Choose instance launch options

Step 3 (optional)  
Configure advanced options

Step 4 (optional)  
Configure group size and scaling policies

Step 5 (optional)  
Add notifications

Step 6 (optional)  
Add tags

Step 7  
Review

Choose launch template or configuration

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group. If you currently use launch configurations, you might consider migrating to launch templates.

Name

Auto Scaling group name

Enter a name to identify the group.

Code\_deploy\_ASG

Must be unique to this account in the current Region and no more than 255 characters.

Launch configuration

Switch to launch template

Launch configuration

Choose a launch configuration that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

code\_deploy\_LC

Launch configuration

code\_deploy\_LC

AMI ID

ami-0f2a0e3bd8729b743

Date created

Sun Jan 09 2022 17:51:19 GMT+0530 (India Standard Time)

Security groups

sg-053a09161e6896f

Instance type

t2.micro

Key pair name

Feedback English (US) ▾

© 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences

1640586101195.pdf Show all

Type here to search 15°C Haze 17:52 ENG 09-01-2022

Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.

**Step 2**  
Choose instance launch options

**Step 3 (optional)**  
Configure advanced options

**Step 4 (optional)**  
Configure group size and scaling policies

**Step 5 (optional)**  
Add notifications

**Step 6 (optional)**  
Add tags

**Step 7**  
Review

**Network** Info

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

**VPC**

Choose the VPC that defines the virtual network for your Auto Scaling group.

vpc-f5e2f9b  
172.31.0.0/16 Default

**Create a VPC**

**Availability Zones and subnets**

Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

Select Availability Zones and subnets  
ap-south-1a | subnet-59f0ca31  
172.31.32.0/20 Default

ap-south-1b | subnet-e76e05ab  
172.31.0.0/20 Default

ap-south-1c | subnet-53f94428  
172.31.16.0/20 Default

Create a subnet

Cancel Previous Skip to review Next

Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

**Step 3 (optional)**  
Configure advanced options

**Step 4 (optional)**  
Configure group size and scaling policies

**Step 5 (optional)**  
Add notifications

**Step 6 (optional)**  
Add tags

**Step 7**  
Review

**Attach to an existing load balancer**

Select the load balancers that you want to attach to your Auto Scaling group.

Choose from your load balancer target groups  
This option allows you to attach Application, Network, or Gateway Load Balancers.

Choose from Classic Load Balancers

**Existing load balancer target groups**

Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.

Select target groups  
code-deployT | HTTP Application Load Balancer: code-deploy-elb

**Health checks - optional**

Health check type: Info

EC2 Auto Scaling automatically replaces instances that fail health checks. If you enabled load balancing, you can enable ELB health checks in addition to the EC2 health checks that are always enabled.

Cancel Previous Skip to review Next

I have attached Application Load Balancer to it which is already Created

The screenshot shows the AWS Auto Scaling console interface. On the left, a sidebar lists steps from Step 2 to Step 7. Step 2 is 'Choose instance launch options', Step 3 is 'Configure advanced options', Step 4 is 'Configure group size and scaling policies', Step 5 is 'Add notifications', Step 6 is 'Add tags', and Step 7 is 'Review'. The main area is titled 'Group size - optional' and contains fields for Desired capacity (set to 2), Minimum capacity (set to 2), and Maximum capacity (set to 2). Below this is a section titled 'Scaling policies - optional' with a radio button for 'None' selected.

## Give Desired and max capacity to 2

The screenshot shows the AWS EC2 Auto Scaling groups page. The sidebar on the left includes sections for New EC2 Experience, EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (with sub-options like Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Scheduled Instances, Capacity Reservations), Images, and Elastic Block Store. The main content area displays the 'Auto Scaling groups (1)' table. The table has columns for Name, Launch template/configuration, Instances, Status, Desired capacity, Min, Max, and Availability Zones. One row is shown for 'code\_deploy\_ASG' with 'code\_deploy\_LC' as the launch configuration, 0 instances, 'Updating capacity' status, Desired capacity 2, Min 2, Max 2, and Availability Zones 'ap-south-1c, ap-south-1d'.

## Auto Scaling Group is created.

We can see as soon as ASG created , the two instances are up.

Screenshot of the AWS CloudWatch Metrics console showing two metrics for the instance i-0fe8dc7ded70c8812. The metrics are:

Metric	Value
CodeDeploy Task State	Task Failed
CodeDeploy Task Duration (Seconds)	~100

The screenshot also shows the AWS Lambda function configuration for the Lambda@Edge trigger, which is set to invoke the function on 'All' events.

## **Step 6 - Create one Jenkins Instance and install jenkins software in it.**

I have used the below link to install JENKINS server into the ec2 instance -

<https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkins-on-AWS/>

**NOTE** - i have also installed git in it , using Command -

- yum install git -y

# Step 7 - Create CodeDeploy

The screenshot shows the AWS CodeDeploy landing page. On the left, there's a sidebar titled 'Developer Tools' with a 'CodeDeploy' section expanded, showing options like 'Source', 'Artifacts', 'Build', 'Deploy', 'Getting started', 'Deployments', 'Applications', 'Deployment configurations', 'On-premises instances', 'Pipeline', and 'Settings'. Below this is a search bar and links for 'Feedback' and 'English (US)'. The main content area features the 'AWS CodeDeploy' logo and the tagline 'Automate code deployments to maintain application uptime'. It includes a brief description of what CodeDeploy does, a 'Create AWS CodeDeploy deployment' button, a 'Pricing (US)' table, and a 'How it works' section with a video thumbnail. The bottom of the page has a footer with links for 'Privacy', 'Terms', and 'Cookie preferences', along with system status information like '15°C Haze' and '18:10 09-01-2022'.

## Create Application

The screenshot shows the 'Create application' configuration page. At the top, there's a breadcrumb trail: 'Developer Tools > CodeDeploy > Applications > Create application'. The main form is titled 'Application configuration' and contains fields for 'Application name' (with 'code-deploy-App' entered) and 'Compute platform' (set to 'EC2/On-premises'). There are 'Cancel' and 'Create application' buttons at the bottom. The bottom of the page has a footer with links for 'Privacy', 'Terms', and 'Cookie preferences', along with system status information like '14°C Haze' and '18:10 09-01-2022'.

## Now Create Deployment Group.

The screenshot shows the 'Create deployment group' wizard in the AWS CodeDeploy console. The steps completed so far are:

- Application**: Application selected is 'code-deploy-App', Compute type is 'EC2/On-premises'.
- Deployment group name**: Name entered is 'code-deploy-DG'.
- Service role**: Service role selected is 'arn:aws:iam::616448194411:role/codedeployrole'.

The next step is **Deployment type**, which is currently empty.

Now Choose Amazon Ec2 ASG which we have Created earlier.

The screenshot shows the 'Choose how to deploy your application' step in the AWS CodeDeploy console. Two options are available:

- In-place**: Updates the instances in the deployment group with the latest application revision. During a deployment, each instance will be briefly taken offline for its update.
- Blue/green**: 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.

The next step is **Environment configuration**, where the user selects the deployment environment. Under 'Amazon EC2 Auto Scaling groups', 'Code\_deploy\_ASG' is selected. Other options include 'On-premises instances'.

Now choose Application Load balancer which we have created earlier.

The screenshot shows the AWS CodeDeploy console interface. In the top navigation bar, the URL is `ap-south-1.console.aws.amazon.com/codesuite/codedeploy/applications/code-deploy-App/deployment-groups/new?region=ap-south-1`. The main area is titled "Deployment settings". Under "Deployment configuration", there is a dropdown menu set to "CodeDeployDefault.AllAtOnce" and a button to "Create deployment configuration". Below this is the "Load balancer" section, which includes a checkbox for "Enable load balancing" and a radio button selected for "Application Load Balancer or Network Load Balancer". A dropdown menu for "Choose a target group" shows "code-deployTG" selected. At the bottom of the page, there is a "Create deployment group" button.

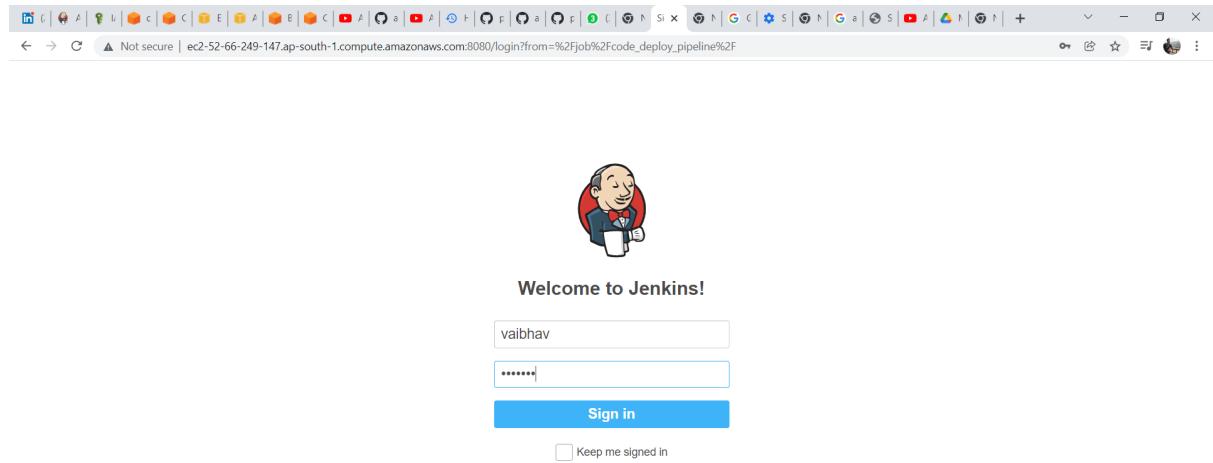
**Deployment Group Is Created.**  
Now we will use Jenkins for Creating a pipeline that will use git as a source and will run the CodeDeploy.

The screenshot shows the AWS CodeDeploy console after a deployment group has been created. The success message "Deployment group created" is displayed at the top. The main area shows the deployment group details for "code-deploy-DG". The deployment group name is "code-deploy-DG", the application name is "code-deploy-App", and the deployment type is "In-place". The service role ARN is listed as `arn:aws:iam::616448194411:role/codedeplyrole`. The deployment configuration is set to "CodeDeployDefault.AllAtOnce". Below this, the "Environment configuration: Amazon EC2 Auto Scaling groups" section shows a single entry with the name "Code\_deploy\_ASG". At the bottom, there is a "Triggers" section and a "Create deployment" button.

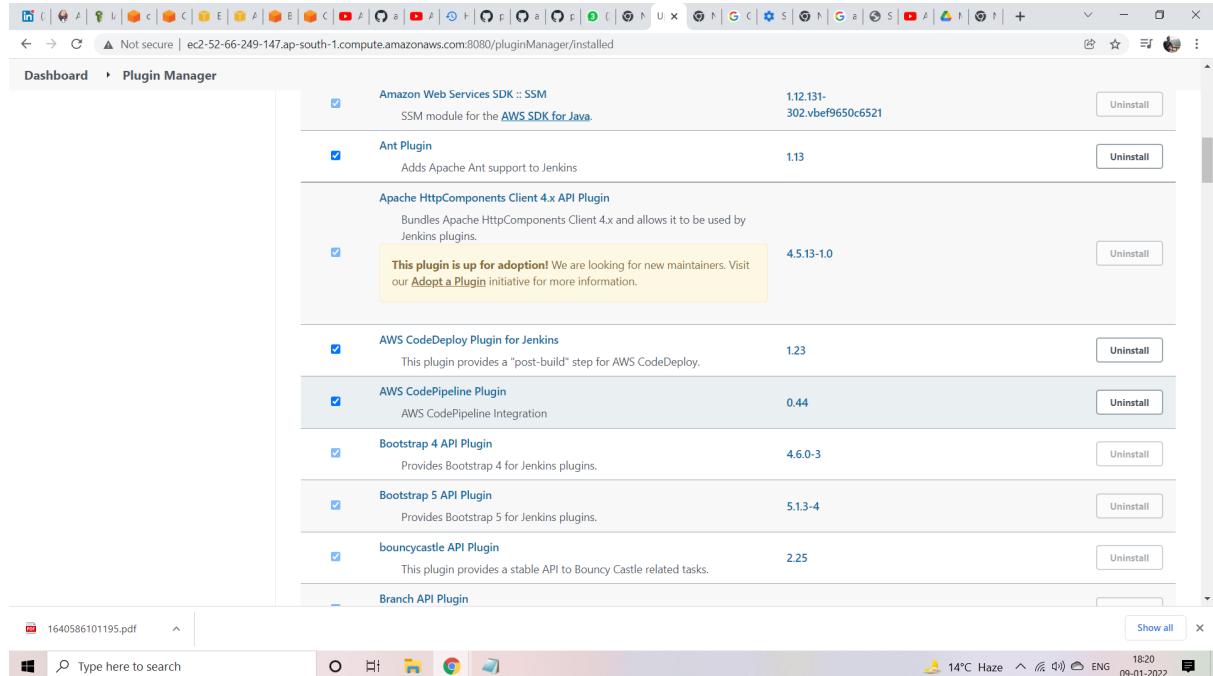
## Step 8 - Create an S3 bucket.

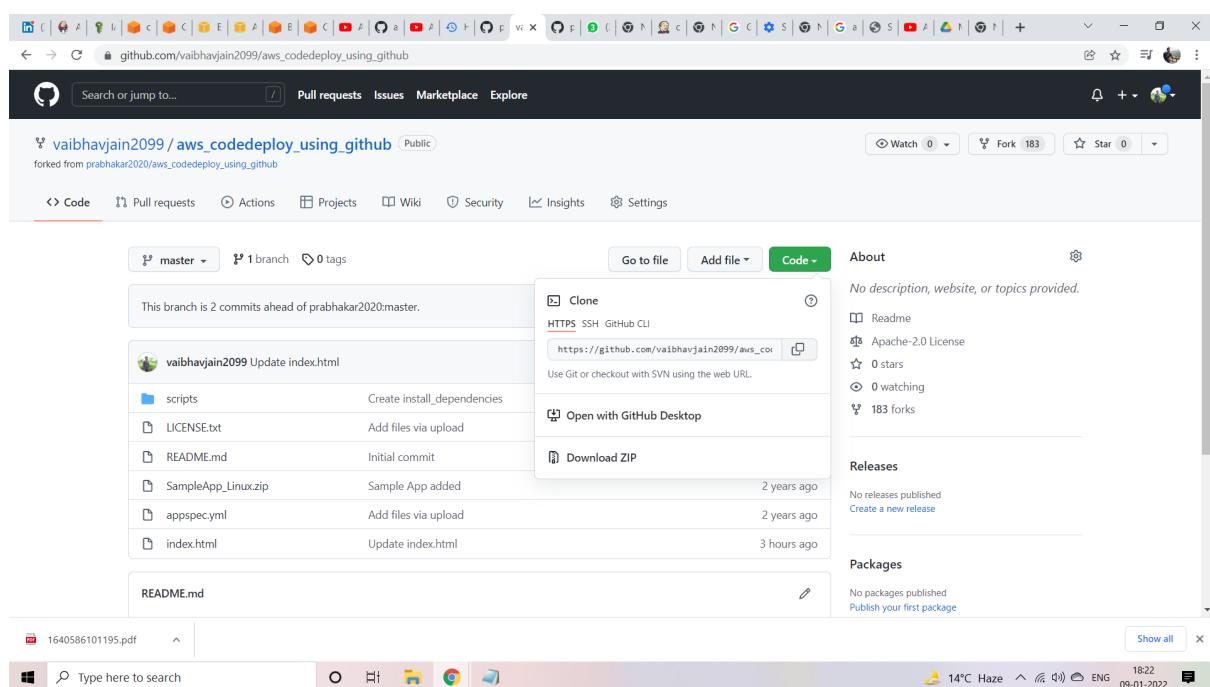
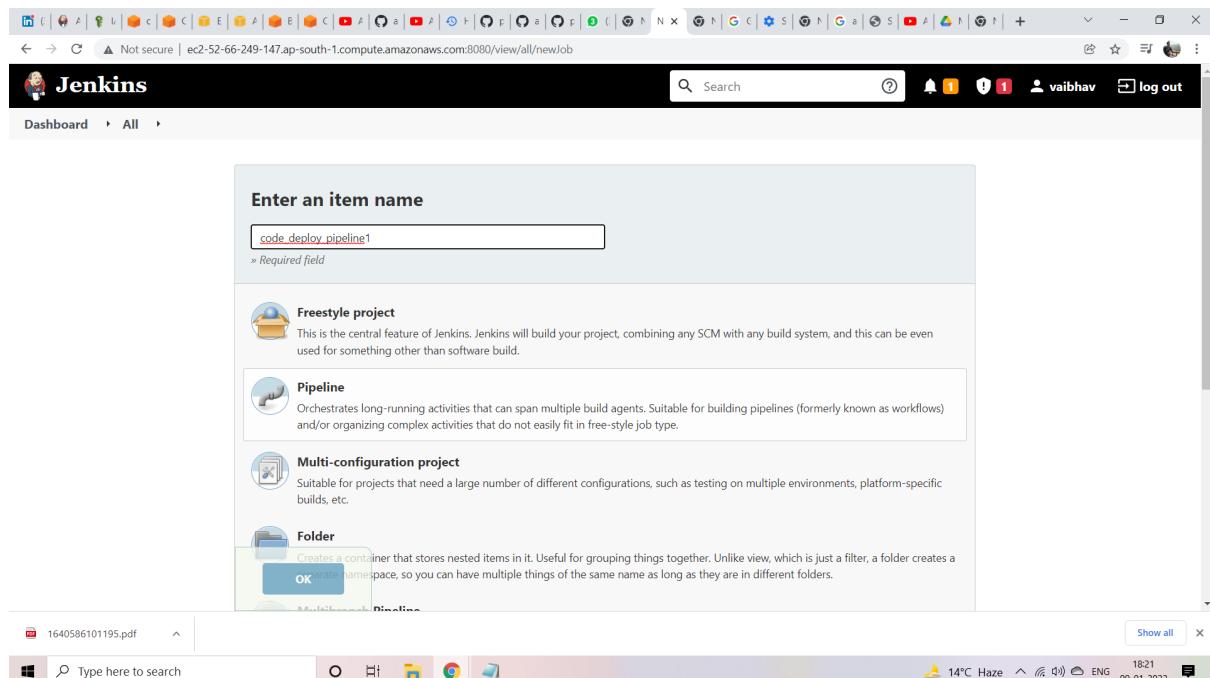
The screenshot shows the AWS S3 console interface. On the left, a sidebar menu includes 'Buckets', 'Access Points', 'Object Lambda Access Points', 'Multi-Region Access Points', 'Batch Operations', 'Access analyzer for S3', 'Block Public Access settings for this account', 'Storage Lens' (with 'Dashboards' and 'AWS Organizations settings' sub-options), 'Feature spotlight', and 'AWS Marketplace for S3'. The main content area displays the 'code-deploy-buck1' bucket. The 'Objects' tab is selected, showing one object: 'code-deploy-folder/'. A table below lists this object, showing its name, type (Folder), last modified (dash), size (dash), and storage class (dash). Above the table are buttons for Actions, Create folder, and Upload, along with a search bar for 'Find objects by prefix' and a 'Show versions' toggle. The top navigation bar shows the URL 's3.console.aws.amazon.com/s3/buckets/code-deploy-buck1?region=ap-south-1&tab=objects' and the AWS logo. The bottom of the screen shows the Windows taskbar with various pinned icons and system status information.

## Step 9 - Go to Jenkins Server



- Install AWS Code Deploy plugin in it





[https://github.com/vaibhavjain2099/aws\\_codedeploy\\_using.github.git](https://github.com/vaibhavjain2099/aws_codedeploy_using.github.git)

Create a jenkins pipeline by taking GIT as a source , Trigger - Poll SCM ( \* \* \* \* \* ) which means it will run job in every minute. Put S3 Bucket name etc.

The screenshot shows two consecutive views of the AWS CodePipeline configuration interface for a pipeline named "code\_deploy\_pipeline1".

**Source Code Management Tab:**

- General:** Includes checkboxes for "Disable this project" and "Execute concurrent builds if necessary".
- Source Code Management:** Set to "Git".
  - Repositories:** A single repository is configured with the URL [https://github.com/vaibhavjain2099/aws\\_codedeploy\\_using\\_github.git](https://github.com/vaibhavjain2099/aws_codedeploy_using_github.git).
  - Credentials:** Associated with the repository.
- Buttons:** "Save" and "Apply".

**Build Triggers Tab:**

- General:** Includes checkboxes for "Trigger builds remotely (e.g., from scripts)", "Build after other projects are built", "Build periodically", "GitHub hook trigger for GITScm polling", and "Poll SCM".
- Build Triggers:** A section where the "Poll SCM" trigger is selected, showing a scheduled run time of "Sunday, January 9, 2022 12:53:31 PM UTC". A note states: "⚠ Do you really mean "every minute" when you say \*\*\*\*? Perhaps you meant "H\*\*\*\*" to poll once per hour".
- Build Environment:** Includes checkboxes for "Delete workspace before build starts" and "Use secret text(s) or file(s)".
- Buttons:** "Save" and "Apply".

Choose post-build-action as deploy an application AWS CodeDeploy

The screenshot shows the Jenkins 'Build Environment' configuration page. A context menu is open over the 'Post-build Actions' section, listing various options such as 'Deploy an application to AWS CodeDeploy'. The 'Deploy an application to AWS CodeDeploy' option is highlighted with a blue background.

Dashboard > code\_deploy\_pipeline1 >

General Source Code Management Build Triggers Build Environment Build Post-build Actions

Build Environment

Aggregate downstream test results  
Archive the artifacts  
Build other projects  
Publish JUnit test result report  
Record fingerprints of files to track usage  
Git Publisher  
Deploy an application to AWS CodeDeploy  
AWS CodePipeline Publisher  
E-mail Notification  
Editable Email Notification  
Set GitHub commit status (universal)  
Set build status on GitHub commit (deprecated)  
Delete workspace when build is done

Add post-build action ▾

Save Apply

A screenshot of a Windows taskbar. It shows several pinned icons and an open file named '1640586101195.pdf'. The system tray indicates the date as 09-01-2022, the time as 18:23, and the weather as 14°C Haze.

Than give all info of CodeDeploy like app name , deployment group name , s3 bucket and prefix name.

The screenshot shows the Jenkins 'Post-build Actions' configuration page specifically for AWS CodeDeploy. The 'Deploy an application to AWS CodeDeploy' action is selected. The configuration fields include:

- AWS CodeDeploy Application Name: code-deploy-App
- AWS CodeDeploy Deployment Group: code-deploy-DG
- AWS CodeDeploy Deployment Config: (empty)
- AWS Region: AP SOUTH 1
- S3 Bucket: code-deploy-buck1
- S3 Prefix: code-deploy-folder
- Subdirectory: (empty)

At the bottom are 'Save' and 'Apply' buttons.

Dashboard > code\_deploy\_pipeline1 >

General Source Code Management Build Triggers Build Environment Build Post-build Actions

Post-build Actions

Deploy an application to AWS CodeDeploy

AWS CodeDeploy Application Name ?  
code-deploy-App

AWS CodeDeploy Deployment Group ?  
code-deploy-DG

AWS CodeDeploy Deployment Config

AWS Region  
AP SOUTH 1

S3 Bucket ?  
code-deploy-buck1

S3 Prefix ?  
code-deploy-folder

Subdirectory ?

Save Apply

A screenshot of a Windows taskbar. It shows an open file named '#4-6126511468818...zip' and an open file named '1640586101195.pdf'. The system tray indicates the date as 09-01-2022, the time as 18:28, and the weather as 14°C Haze.

Than for Authorization we can use Aws Acess keys and Secret keys.

The screenshot shows the Jenkins configuration interface for a pipeline job named "code\_deploy\_pipeline1". The "Post-build Actions" tab is selected. Under "AWS Access Key", the "Use Access/Secret keys" option is selected, and the "AWS Access Key" field is empty. Under "AWS Secret Key", the "Use Access/Secret keys" option is selected, and the "AWS Secret Key" field is empty. There is also an "Add post-build action" dropdown menu. At the bottom are "Save" and "Apply" buttons.

# Step 10 - Build Jenkins Pipeline

The screenshot shows the Jenkins Project dashboard for 'code\_deploy\_pipeline1'. The left sidebar contains links for Status, Changes, Workspace, Build Now (with a 'Build Now' button), Delete Project, Git Polling Log, and Rename. The main content area displays the project name 'Project code\_deploy\_pipeline1', a 'Workspace' section, and a 'Recent Changes' section. Below these are 'Permalinks' and a 'Build History' section showing a single build from Jan 9, 2022, at 1:01 PM. The bottom status bar indicates 'ec2-52-66-249-147.ap-south-1.compute.amazonaws.com:8080/job/code\_deploy\_pipeline1/b...' and 'Jenkins 2.319.1'.

## Check console output.

The screenshot shows the Jenkins Console Output for build #1. The left sidebar includes links for Status, Changes, and Console Output (which is selected). The main content area displays the 'Console Output' section with a green checkmark icon. It shows the log output for building the project, which includes cloning from a GitHub repository, fetching upstream changes, and creating a zip file for deployment. The log concludes with 'Finished: SUCCESS'. The bottom status bar indicates 'ec2-52-66-249-147.ap-south-1.compute.amazonaws.com:8080/job/code\_deploy\_pipeline1/1/console' and 'Jenkins 2.319.1'.

As soon as we will hit the Build Now button the job will run and trigger for code deploy deployment . we can see here it is in progress.

Deployment Id	Status	Deployment type	Compute platform	Application	Deployment group	Revision location	Initiating event	Start time	End time
d-0OKXMG75E	In progress	In-place	EC2/On-premises	code-deploy-App	code-deploy-DG	s3://code-d...	User action	Jan 9, 2022 6:31 PM (UTC+5:30)	-

Application	Deployment ID	Status
code-deploy-App	d-0OKXMG75E	In progress

Deployment configuration	Deployment group	Initiated by
CodeDeployDefault.AllAtOnce	code-deploy-DG	User action

Deployment description
Deployment created by Jenkins

Revision details	
Revision location	Revision created
s3://code-deploy-buck1/code-deploy-folder/#1-4945971947809810849.zip	2 minutes ago
eTag=1b48056ed9fc5e5fc4f2e1e0b0bd0c	
Revision description	
Application revision registered via Jenkins	

Deployment lifecycle events						
Instance ID	Duration	Status	Most recent event	Events	Start time	End time
i-06fea4edcee638141	-	In progress	BlockTraffic	<a href="#">View events</a>	Jan 9, 2022 6:31 PM (UTC+5:30)	-
i-0fe8dc7ded70c8812	-	In progress	BlockTraffic	<a href="#">View events</a>	Jan 9, 2022 6:31 PM (UTC+5:30)	-

The screenshot shows the AWS CodeDeploy console with a deployment event table. The table has columns for Event, Duration, Status, Error code, Start time, and End time. The events listed are:

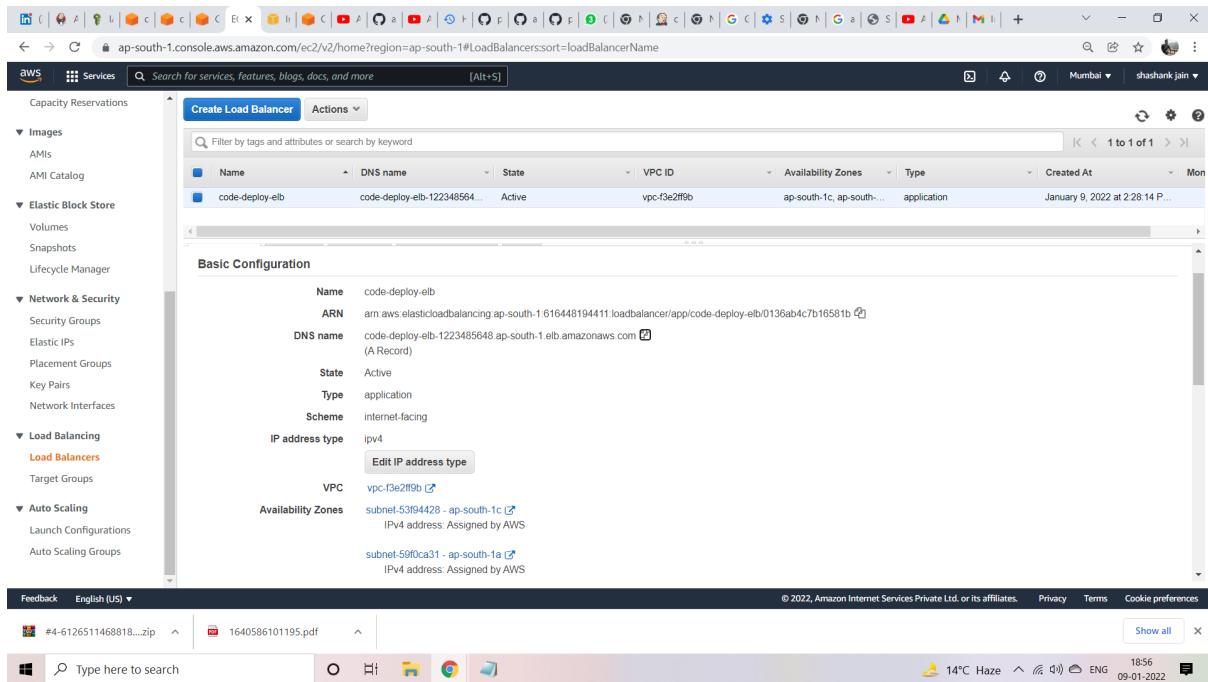
Event	Duration	Status	Error code	Start time	End time
BeforeBlockTraffic	less than one second	<span style="color: green;">Succeeded</span>	-	Jan 9, 2022 6:31 PM (UTC+5:30)	Jan 9, 2022 6:31 PM (UTC+5:30)
BlockTraffic	-	<span style="color: blue;">In progress</span>	-	Jan 9, 2022 6:31 PM (UTC+5:30)	-
AfterBlockTraffic	-	<span style="color: blue;">Pending</span>	-	-	-
ApplicationStop	-	<span style="color: blue;">Pending</span>	-	-	-
DownloadBundle	-	<span style="color: blue;">Pending</span>	-	-	-
BeforeInstall	-	<span style="color: blue;">Pending</span>	-	-	-
Install	-	<span style="color: blue;">Pending</span>	-	-	-
AfterInstall	-	<span style="color: blue;">Pending</span>	-	-	-
ApplicationStart	-	<span style="color: blue;">Pending</span>	-	-	-
ValidateService	-	<span style="color: blue;">Pending</span>	-	-	-
BeforeAllowTraffic	-	<span style="color: blue;">Pending</span>	-	-	-
AllowTraffic	-	<span style="color: blue;">Pending</span>	-	-	-
AfterAllowTraffic	-	<span style="color: blue;">Pending</span>	-	-	-

On the left, there is a sidebar with navigation links for Source, Artifacts, Build, Deploy, Pipeline, and Settings. At the bottom, there are tabs for Feedback and English (US), and a search bar.

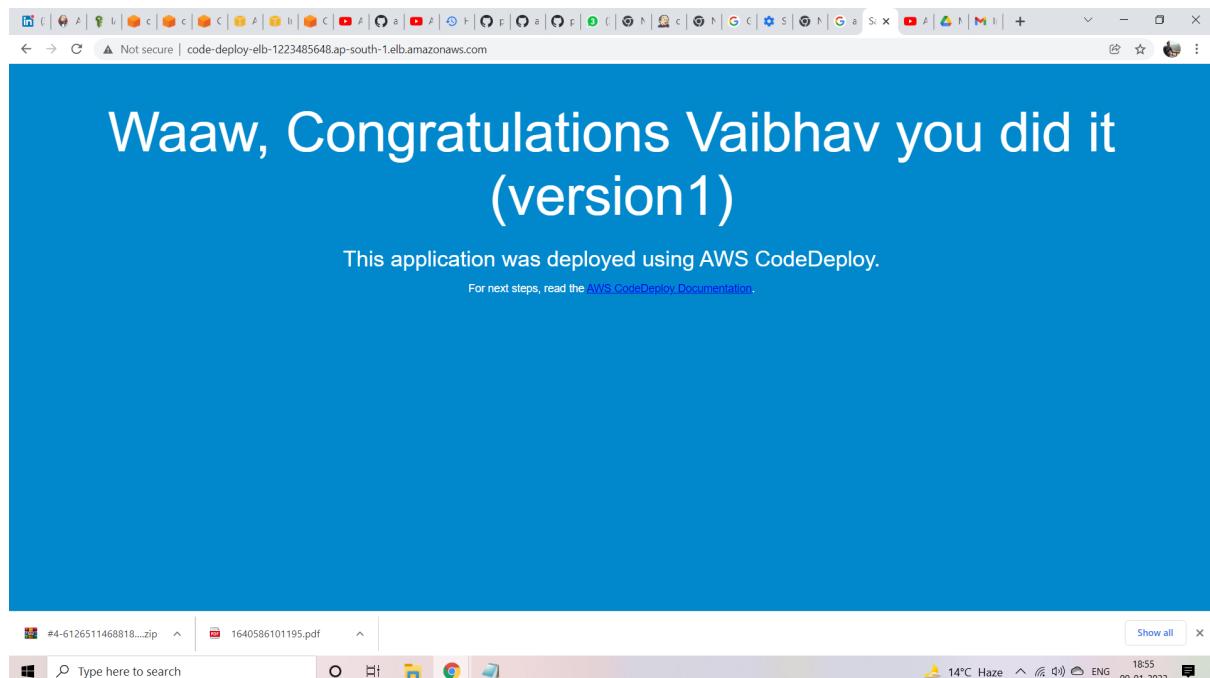
Here we can see , it take time , as we have choosed the option of build all at once.

## **Step 11 -**

Go to AWS Load balancer portal and go to that particular ELB you have Created and copy the DNS and paste it on browser , your application will be running .



This is the first version of application. Now if i made any change to the code it will automatically reflect to the Web browser within some minutes.



We can check that as soon as we run the jenkins job it run the deployment the data will get stored in s3 bucket.

A screenshot of the AWS S3 console. The URL in the address bar is "s3.console.aws.amazon.com/s3/buckets/code-deploy-buck1?region=ap-south-1&amp;prefix=code-deploy-folder&amp;showversions=false". The left sidebar shows "Amazon S3" with sections for Buckets, Storage Lens, and Feature spotlight. The main content area shows a list of objects under the prefix "code-deploy-folder/". There are 6 objects listed, all of which are zip files. The table includes columns for Name, Type, Last modified, Size, and Storage class. The last modified column shows dates ranging from January 9, 2022, to January 10, 2022. The size column shows file sizes from 10.6 KB to 10.6 MB. The storage class column shows Standard for all files. At the bottom of the page, there are links for Feedback, English (US), Privacy, Terms, and Cookie preferences.

Commit changes

Update index.html

Add an optional extended description...

Commit directly to the `master` branch.

Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

We can see as soon as we changed the code in index.html file , a new deployment will be triggered by jenkins job .

Console Output

```

Started by an SCM change
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/code_deploy_pipeline1
The recommended git tool is: NONE
using credential 7fc5df9b-3dc8-46b3-9b6d-81bfe3f34bd
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/code_deploy_pipeline1/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/vaibhavjain2099/aws_codedeploy_using_github.git # timeout=10
Fetching upstream changes from https://github.com/vaibhavjain2099/aws_codedeploy_using_github.git
> git --version # timeout=10
> git -version # git version 2.32.0*
using GIT_ASKPASS to set credentials
> git fetch --tags --force --progress -- https://github.com/vaibhavjain2099/aws_codedeploy_using_github.git +refs/heads/*:refs/remotes/origin/*
timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision d7a0bd5003975578831152c3d24364050c08b6b (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f d7a0bd5003975578831152c3d24364050c08b6b # timeout=10
Commit message: "Update index.html"
> git rev-list --no-walk 4db5e785adeee4714a9ebfbe3fc44bc6b04500 # timeout=10
Zipping files into /tmp/f3-8580387213537281487.zip
Uploading zip to s3://code-deploy-buck1/code-deploy-folder/#3-8580387213537281487.zip
Creating deployment with revision at {RevisionType: S3,S3Location: {Bucket: code-deploy-buck1,Key: code-deploy-folder/#3-8580387213537281487.zip,BundleType: zip,ETag: b050f2435e1c2a9aad8b2ebd3272998},}
Finished: SUCCESS

```

# In progress

The screenshot shows the AWS CodeDeploy console with the URL [ap-south-1.console.aws.amazon.com/codesuite/codedeploy/deployments?region=ap-south-1&deployments.meta=eyJmljlp7InRleHQiOlfIiSwicyf6e30slm4lQ\]UwlC\]pljowfQ](https://ap-south-1.console.aws.amazon.com/codesuite/codedeploy/deployments?region=ap-south-1&deployments.meta=eyJmljlp7InRleHQiOlfIiSwicyf6e30slm4lQ]UwlC]pljowfQ). The left sidebar is titled 'Developer Tools' and includes sections for Source, Artifacts, Build, Deploy, Pipeline, and Settings. The Deploy section is expanded, showing 'CodeDeploy' with sub-options like Getting started, Deployments, Applications, Deployment configurations, On-premises instances, Pipeline, and Settings. The main content area is titled 'Deployment history' and lists three deployments:

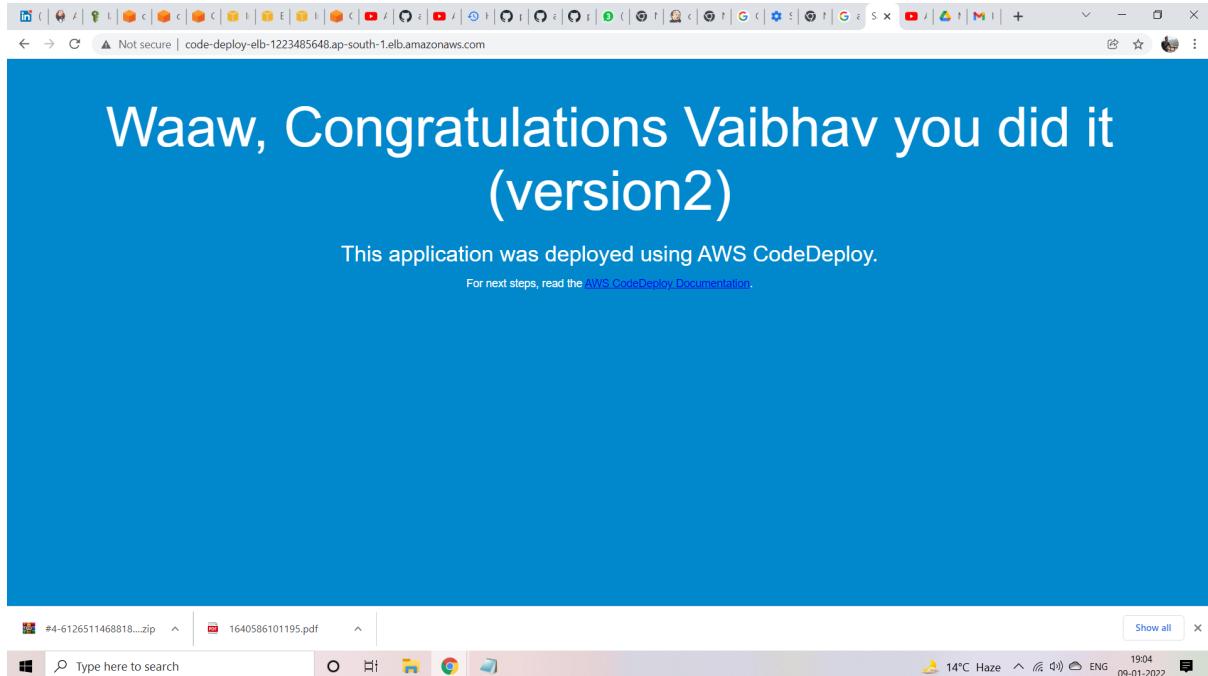
Deployment Id	Status	Deployment type	Compute platform	Application	Deployment group	Revision location	Initiating event	Start time	End time
d-THOHIM75E	In progress	In-place	EC2/On-premises	code-deploy-App	code-deploy-DG	s3://code-d...	User action	Jan 9, 2022 6:59 PM (UTC+5:30)	-
d-XM1GFL75E	Succeeded	In-place	EC2/On-premises	code-deploy-App	code-deploy-DG	s3://code-d...	User action	Jan 9, 2022 6:47 PM (UTC+5:30)	Jan 9, 2022 6:53 PM (UTC+5:30)
d-0OKXMG75E	Failed	In-place	EC2/On-premises	code-deploy-App	code-deploy-DG	s3://code-d...	User action	Jan 9, 2022 6:31 PM (UTC+5:30)	Jan 9, 2022 6:36 PM (UTC+5:30)

At the bottom, there are tabs for Feedback, English (US), Show all, Privacy, Terms, and Cookie preferences. The system status bar shows 14°C Haze, ENG, 19:01, and 09-01-2022.

The screenshot shows the AWS CodeDeploy console with the URL [ap-south-1.console.aws.amazon.com/codesuite/codedeploy/deployments/d-THOHIM75E/instances/arn%3Aaws%3Aec2%3Aap-south-1%3Aa616448194411%3Ainstance/i-06fea4edcee638141?regi...](https://ap-south-1.console.aws.amazon.com/codesuite/codedeploy/deployments/d-THOHIM75E/instances/arn%3Aaws%3Aec2%3Aap-south-1%3Aa616448194411%3Ainstance/i-06fea4edcee638141?regi...). The left sidebar is identical to the previous screenshot. The main content area is titled 'Revision details' and shows the revision location as s3://code-deploy-buck1/code-deploy-folder/#3-0580357213537281487.zip?#Tag=b058f2435e1c2a9aad8b2ebd3272398. It also displays the revision created 4 minutes ago and the revision description 'Application revision registered via Jenkins'. Below this, a table shows the event timeline:

Event	Duration	Status	Error code	Start time	End time
BeforeBlockTraffic	less than one second	Succeeded	-	Jan 9, 2022 6:59 PM (UTC+5:30)	Jan 9, 2022 6:59 PM (UTC+5:30)
BlockTraffic	5 minutes 1 second	Succeeded	-	Jan 9, 2022 6:59 PM (UTC+5:30)	Jan 9, 2022 7:04 PM (UTC+5:30)
AfterBlockTraffic	less than one second	Succeeded	-	Jan 9, 2022 7:04 PM (UTC+5:30)	Jan 9, 2022 7:04 PM (UTC+5:30)
ApplicationStop	less than one second	Succeeded	-	Jan 9, 2022 7:04 PM (UTC+5:30)	Jan 9, 2022 7:04 PM (UTC+5:30)
DownloadBundle	-	Pending	-	-	-
BeforeInstall	-	Pending	-	-	-
Install	-	Pending	-	-	-
AfterInstall	-	Pending	-	-	-
ApplicationStart	-	Pending	-	-	-
ValidateService	-	Pending	-	-	-
BeforeAllowTraffic	-	Pending	-	-	-
AllowTraffic	-	Pending	-	-	-

At the bottom, there are tabs for Feedback, English (US), Show all, Privacy, Terms, and Cookie preferences. The system status bar shows 14°C Haze, ENG, 19:04, and 09-01-2022.



And here is the Our Final Output .

We can see that it shows the recent changed done by us in the html file.

Since we have used Auto Scaling Group and using ELB and its url to access the application , we can confirm that it scalable such that when load increases the number of servers scale up and down making sure the new servers have the updated code.

Github repo used -

[https://github.com/vaibhavjain2099/aws\\_codedeploy\\_using\\_github.git](https://github.com/vaibhavjain2099/aws_codedeploy_using_github.git)

Pardon me if i missed any step in between.

**Thank You**

