

Setup a CI/CD pipeline using the tools of 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.

Git repo: <https://github.com/vommidapuchinni/simple-web-app.git>

Step 1: Create a Git Repository

1. Create a GitHub Account:

- o Sign up or log in to GitHub.

2. Create a New Repository:

- o Click on the "New" button or go to GitHub New Repository.
- o Name the repository, simple-web-app.
- o Keep it public.
- o Do not initialize with a README, .gitignore, or license.
- o Click "Create Repository."

3. Clone the Repository Locally:

- o Open your terminal and run the following commands to clone the repository:
git clone https://github.com/your-username/simple-web-app.git
cd simple-web-app

Step 2: Launch EC2 Instances

1. Launch the Jenkins Server Instance:

- o Go to the AWS EC2 Console.
- o Click **Launch Instance**.

Name and tags

Name: jenkins-server

Application and OS Images (Amazon Machine Image)

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

Recents **Quick Start**

Summary

Number of instances: 1

Software Image (AMI): Amazon Linux 2023.5.2... (ami-066784287e358dad1)

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)

Launch instance

- **Name and Tags:** Name the instance jenkins-server.
- **AMI:** Choose **Ubuntu Server 20.04 LTS** (or any preferred version).

Application and OS Images (Amazon Machine Image)

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

Recents **Quick Start**

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Li

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type

Free tier eligible

ami-086e20dae9224db8 (64-bit (x86)) / ami-096ea6a12ea24a797 (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Ubuntu Server 24.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Summary

Number of instances: 1

Software Image (AMI): Canonical, Ubuntu, 20.04, amd64... (ami-086e20dae9224db8)

Virtual server type (instance type): t2.micro

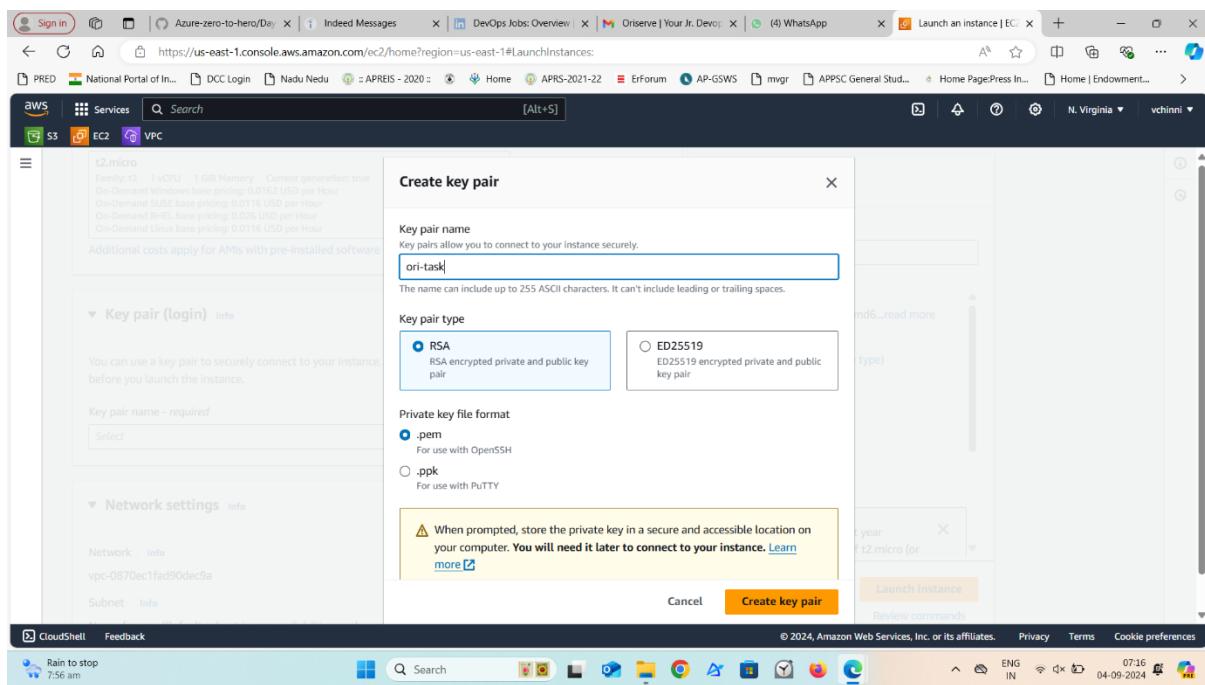
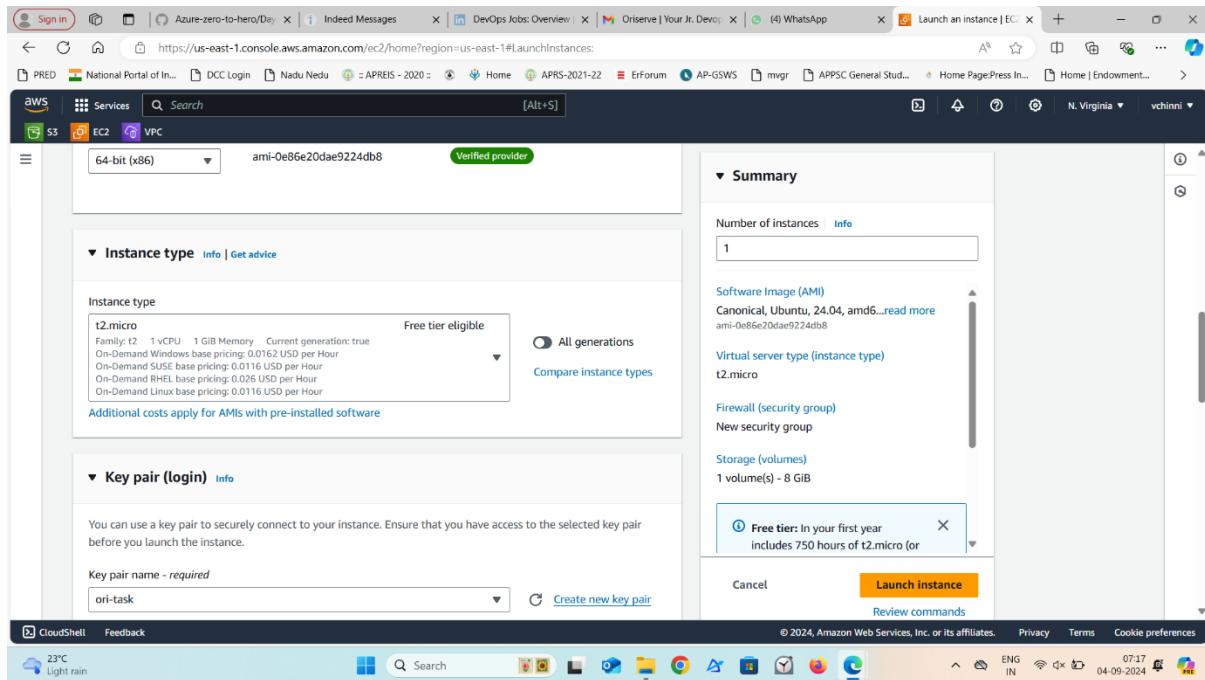
Firewall (security group): New security group

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

Free tier: In your first year includes 750 hours of t2.micro (or)

Launch instance

- **Instance Type:** Choose t2.micro for testing (eligible for free tier).
- **Key Pair:** create a new one.

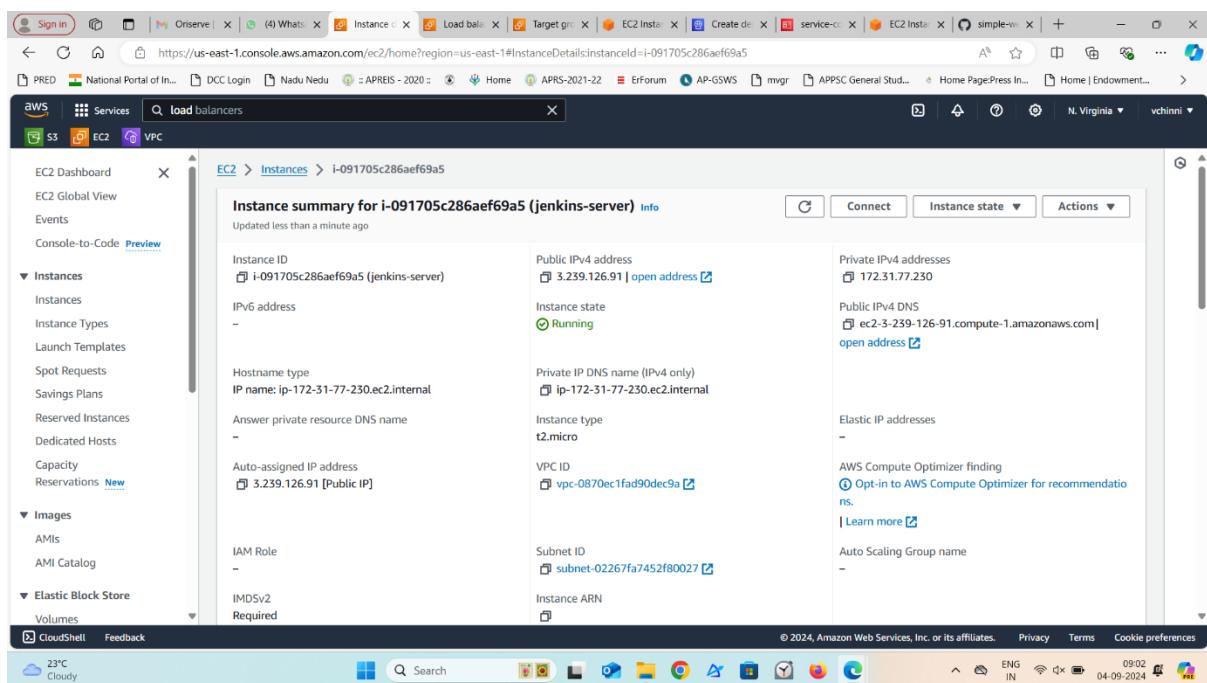
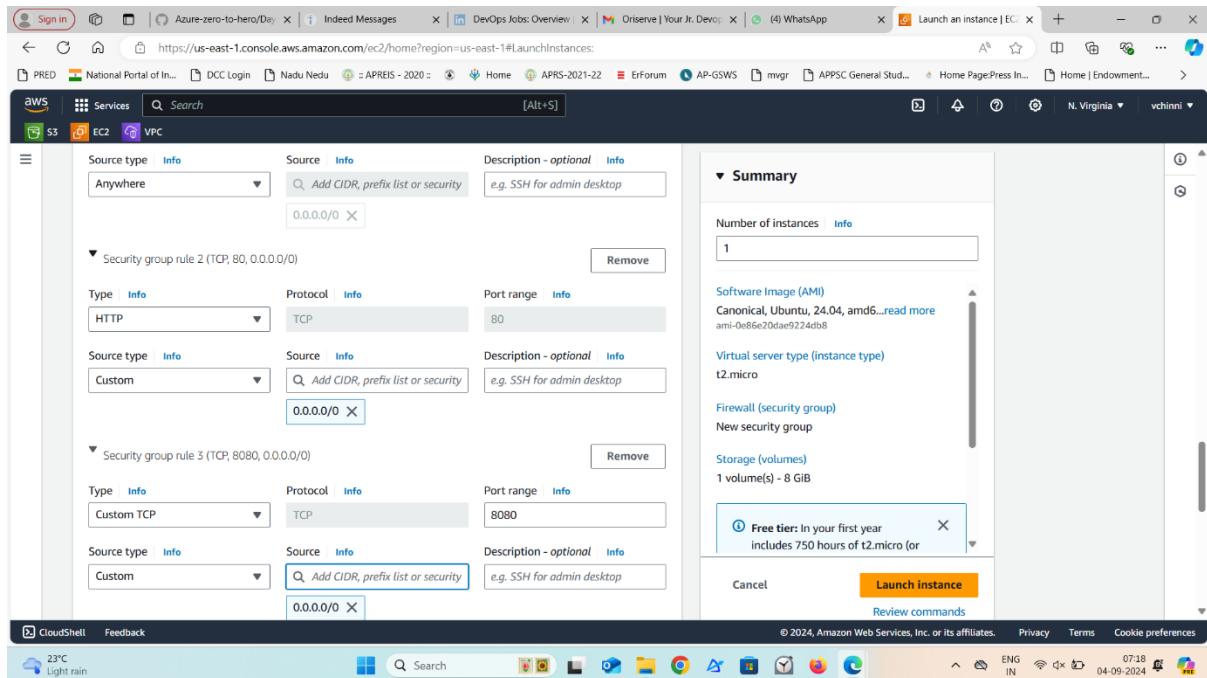


2.

o Network Settings:

- Create a new security group allowing:
 - **SSH (port 22)** from your IP for secure access.
 - **HTTP (port 80)** for web access.
 - **Custom TCP (port 8080)** for Jenkins access.

- Click Launch Instance.



Step 3: Configure Jenkins Server

1. Connect to the Jenkins Server Instance:

- Click on connect

```

Sign in AWS Services Search [Alt+S]
aws EC2 VPC
Memory usage: 21% IPv4 address for enx0: 172.31.77.230
Swap usage: 0%
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-77-230:~$ sudo -i
root@ip-172-31-77-230:~# apt update -y
i-091705c286aef69a5 (jenkins-server)
PublicIPs: 3.239.126.91 PrivateIPs: 172.31.77.230

```

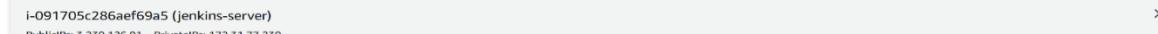


```

Sign in AWS Services Search [Alt+S]
aws EC2 VPC
root@ip-172-31-77-230:~# sudo systemctl enable jenkins
Synchronizing state of jenkins.service with /sysv service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
root@ip-172-31-77-230:~# sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
   Active: active (running) since Wed 2024-09-04 03:50:55 UTC; 25s ago
     Main PID: 4962 (java)
        Tasks: 43 (limit: 1130)
       Memory: 318.0M (peak: 413.2M)
          CPU: 10.578s
         CGroup: /system.slice/jenkins.service
             └─4962 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/War --httpPort=8080

Sep 04 03:36:47 ip-172-31-77-230 jenkins[4962]: 9e19332bbf614d67bd0d02d4c137436
Sep 04 03:36:47 ip-172-31-77-230 jenkins[4962]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Sep 04 03:36:47 ip-172-31-77-230 jenkins[4962]: ****
Sep 04 03:36:47 ip-172-31-77-230 jenkins[4962]: ****
Sep 04 03:36:47 ip-172-31-77-230 jenkins[4962]: ****
Sep 04 03:36:55 ip-172-31-77-230 jenkins[4962]: 2024-09-04 03:36:55.108+0000 [id:29]      INFO  jenkins.InitReactorRunner$1#onAttained: Completed initialization
Sep 04 03:36:55 ip-172-31-77-230 jenkins[4962]: 2024-09-04 03:36:55.143+0000 [id:22]      INFO  hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
Sep 04 03:36:55 ip-172-31-77-230 jenkins[4962]: 2024-09-04 03:36:55.289+0000 [id:45]      INFO  h.m.downloadService$Downloadable#load: Obtained the update site
Sep 04 03:36:55 ip-172-31-77-230 jenkins[4962]: 2024-09-04 03:36:55.293+0000 [id:48]      INFO  hudson.util.Retrier#start: Performed the action check update
root@ip-172-31-77-230:~# cat /var/lib/jenkins/secrets/initialAdminPassword
9e19332bbf614d67bd0d02d4c137436
root@ip-172-31-77-230:~#

```



Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

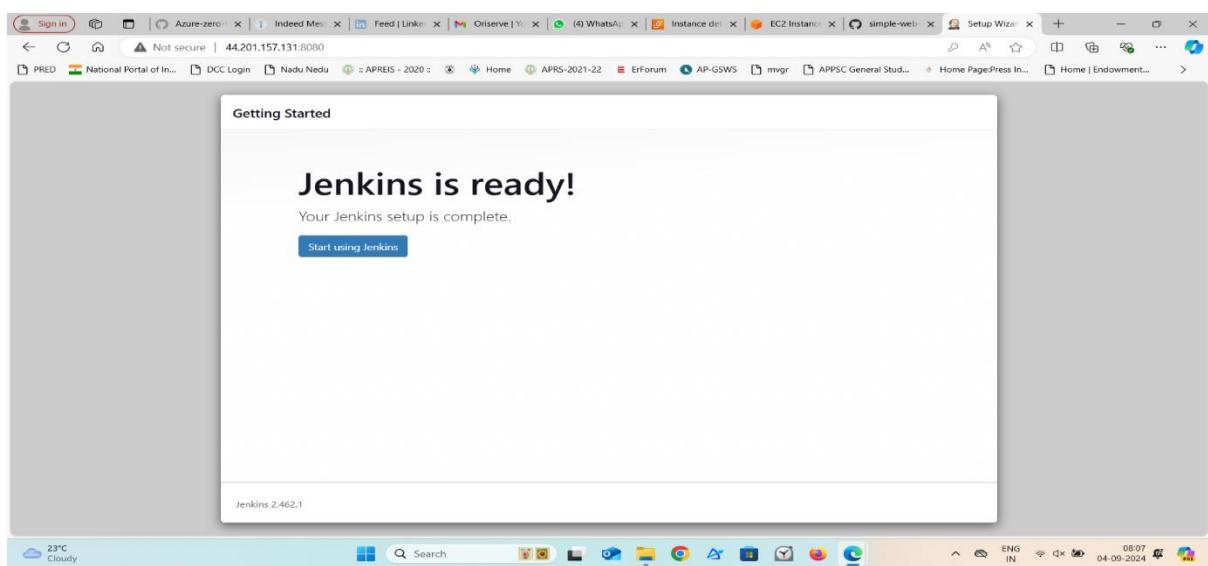
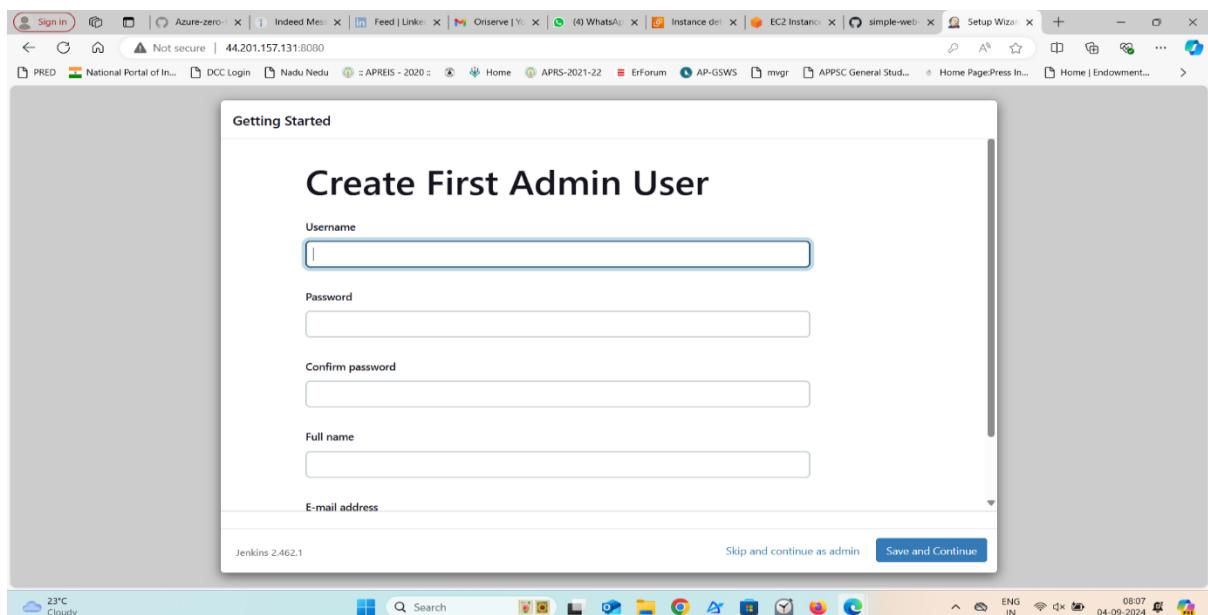
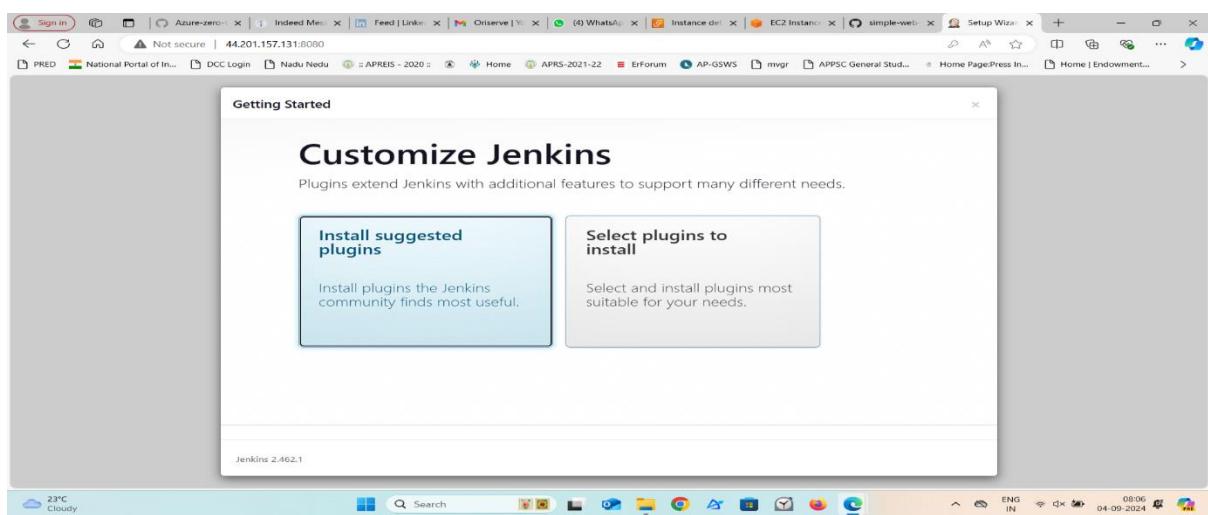
`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Continue





The screenshot shows the Jenkins dashboard. On the left, there's a sidebar with links for 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. Below this are sections for 'Build Queue' (empty) and 'Build Executor Status' (showing 1 Idle and 2 Idle). In the center, a large box says 'Welcome to Jenkins!' with the sub-instruction 'This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.' It includes links for 'Create a job', 'Set up a distributed build', 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. At the bottom right, it says 'REST API Jenkins 2.462.1'. The browser status bar at the bottom shows '23°C Cloudy' and the date '04-09-2024'.

2. Install Jenkins

3. Access Jenkins

Retrieve the initial admin password using:

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

4. Install Required Jenkins Plugins: Install AWS CodeDeploy, Pipeline, and any other required plugins

The screenshot shows the Jenkins 'Manage Jenkins > Plugins' page. The search bar at the top has 'pipeline' typed into it. The results table lists several plugins under the 'Released' category:

Plugin	Description	Last Updated
Pipeline: Stage View	User Interface Pipeline Stage View Plugin.	10 mo ago
AWS CodeDeploy	Artifact Uploader - aws This plugin provides a "post-build" step for AWS CodeDeploy.	4 yr 4 mo ago
AWS CodeBuild	codebuild Build your project on AWS CodeBuild.	1yr 3 mo ago
AWS CodePipeline	aws - Other Post-Build Actions - Source Code Management AWS CodePipeline Integration	10 mo ago
Pipeline: REST API	User Interface	10 mo ago

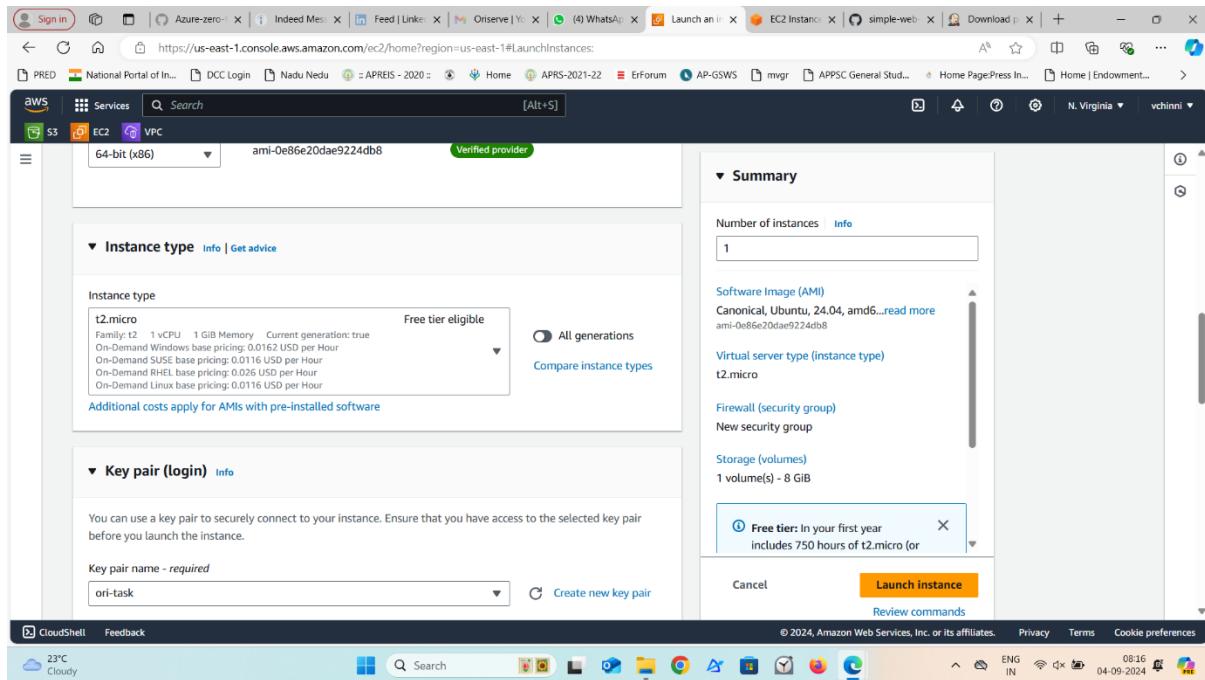
On the left sidebar, there are links for 'Updates', 'Available plugins' (which is selected), 'Installed plugins', 'Advanced settings', and 'Download progress'.

Step 4: Launch Application Server Instances (For CodeDeploy and Scaling):

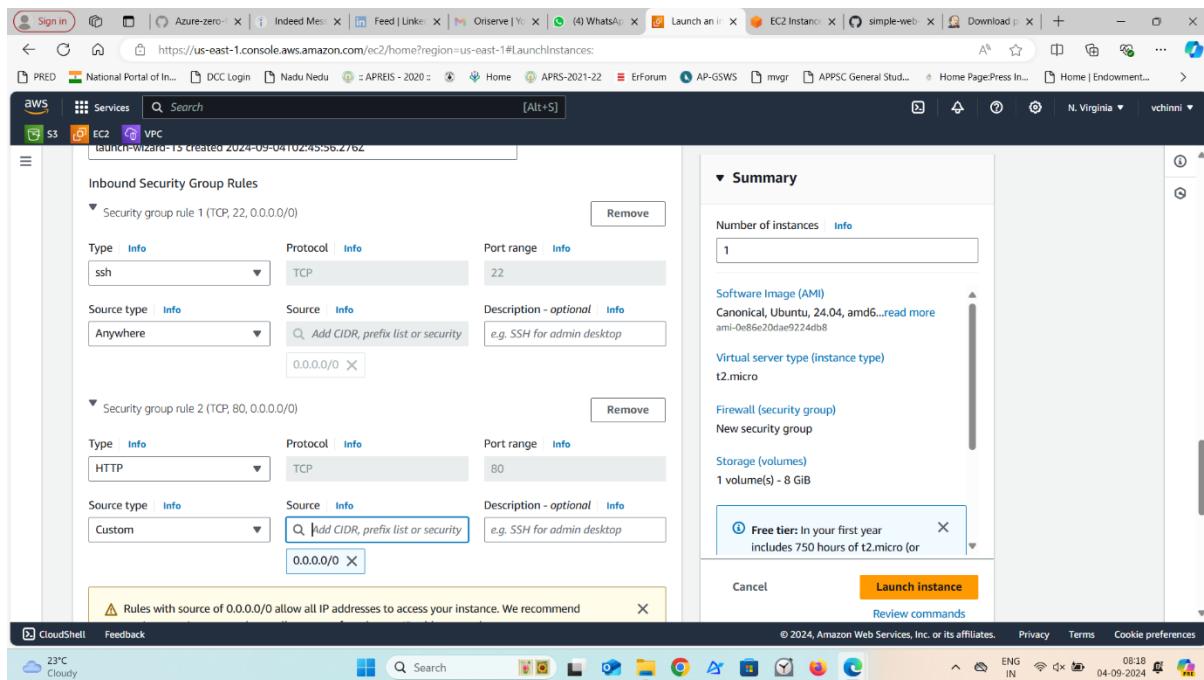
- Repeat the steps above to launch an instance.
- **Name:** Name the instance application-server.

The screenshot shows the AWS EC2 Instances Launch an instance page. In the 'Name and tags' section, the name 'application-server' is entered. Under 'Application and OS Images (Amazon Machine Image)', the search bar is empty. On the right, the 'Summary' panel shows 1 instance, using the Canonical, Ubuntu, 24.04 AMI, t2.micro instance type, and a new security group. A 'Free tier' message indicates 750 hours of t2.micro usage included. The 'Launch instance' button is highlighted in orange.

The screenshot shows the AWS EC2 Instances Application and OS Images page. It displays various AMI options: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and SUSE Linux. The Ubuntu option is selected. The 'Summary' panel on the right shows 1 instance, using the Canonical, Ubuntu, 24.04 AMI, t2.micro instance type, and a new security group. A 'Free tier' message indicates 750 hours of t2.micro usage included. The 'Launch instance' button is highlighted in orange.



- Ensure the instance is within the same VPC and subnet as Jenkins for easier communication.
- Security group settings:
 - **SSH (port 22)** for access.
 - **HTTP (port 80)** for the application.



Step 5: Configure Application Server (For CodeDeploy and Scaling)

1. Connect to the Application Server Instance: click on connect
2. Install Apache and AWS CodeDeploy Agent
3. Attach IAM Role to Application Servers

Screenshot of the AWS EC2 Instances page showing the instance summary for i-0c6646ddf756c596a (application-server). The instance is running and has a Public IPv4 address of 54.80.19.156.

Instance ID	Public IPv4 address	Private IPv4 addresses	
i-0c6646ddf756c596a (application-server)	54.80.19.156 open address	172.31.86.137	
IPv6 address	Instance state	Public IPv4 DNS	
-	Running	ec2-54-80-19-156.compute-1.amazonaws.com open address	
Hostname type	Private IP DNS name (IPv4 only)	Elastic IP addresses	
IP name: ip-172-31-86-137.ec2.internal	ip-172-31-86-137.ec2.internal	-	
Answer private resource DNS name	Instance type	AWS Compute Optimizer finding	
IPv4 (A)	t2.micro	Opt-in to AWS Compute Optimizer for recommendations.	
Auto-assigned IP address	VPC ID	Auto Scaling Group name	
54.80.19.156 [Public IP]	vpc-0870ec1fad90dec9a	-	
IAM Role	Subnet ID	Learn more	
-	subnet-041e243e508c18c9c		
IMDSv2	Instance ARN		
Required	-		

CloudShell session output:

```

Memory usage: 20%           IPV4 address for enX0: 172.31.86.137
Swap usage:  0%             Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-86-137:~$ sudo -i
root@ip-172-31-86-137:~# apt update -y

```

CloudShell session output (continued):

```

i-0c6646ddf756c596a (application-server)
PublicIPs: 54.80.19.156 PrivateIPs: 172.31.86.137

```

```

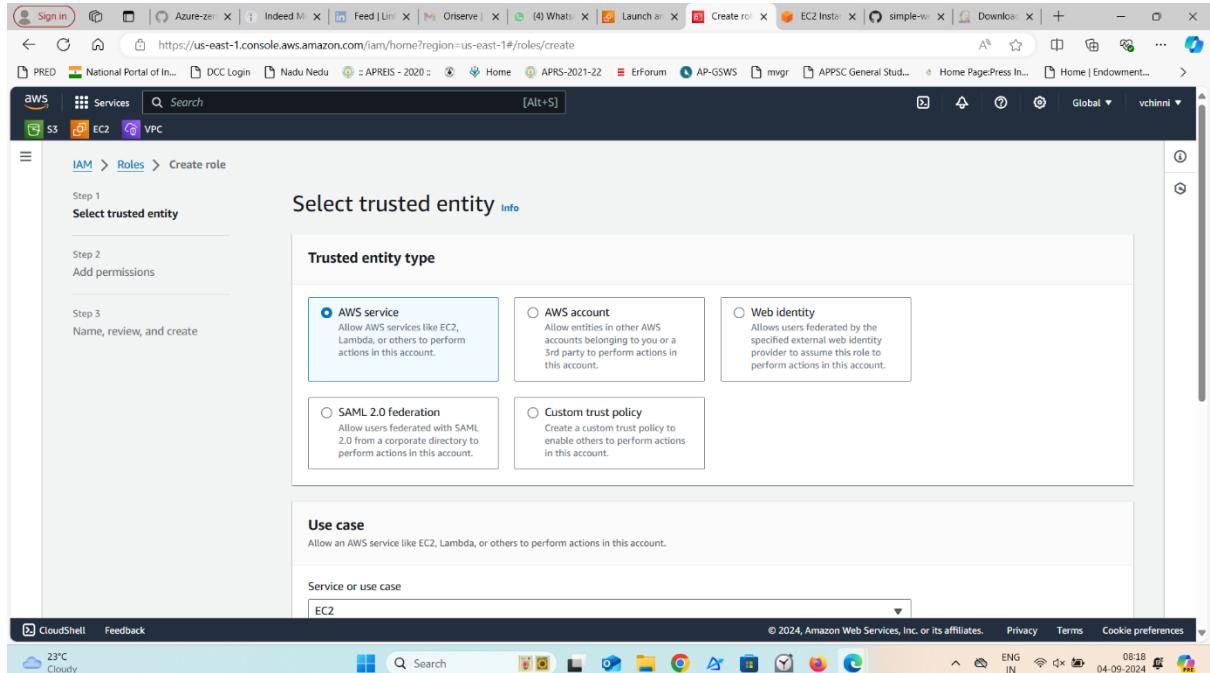
root@ip-172-31-96-137:~# sudo apt install apache2 -y
sudo systemctl start apache2
sudo systemctl enable apache2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libaprutil1 libaprutil1-db4 libaprutil1-ldap libaprutil1t64 liblua5.4-0 ssl-cert
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser
The following NEW packages will be installed:
  apache2-bin apache2-data apache2-utils libaprutil1 libaprutil1-db4 libaprutil1-ldap libaprutil1t64 liblua5.4-0 ssl-cert
0 upgraded, 10 newly installed, 0 to remove and 105 not upgraded.
Need to get 2083 kB of archives.
After this operation, 8094 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1 libaprutil1-db4 libaprutil1-ldap libaprutil1t64 liblua5.4-0 ssl-cert [107 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1t64 amd64 1.6.3-1.1ubuntu7 [91.9 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1-db4 amd64 1.6.3-1.1ubuntu7 [11.2 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1-ldap amd64 1.6.3-1.1ubuntu7 [9116 B]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 liblua5.4-0 amd64 5.4.6-3build2 [166 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2-bin amd64 2.4.58-1ubuntu0.4 [1329 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2-data all 2.4.58-1ubuntu0.4 [163 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2-utils amd64 2.4.58-1ubuntu0.4 [97.1 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2 amd64 2.4.58-1ubuntu0.4 [90.2 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 ssl-cert all 1.1.2ubuntu1 [17.8 kB]
Fetched 2083 kB in 0s (33.8 MB/s)
Preconfiguring packages ...

```

i-0c6646ddf756c596a (application-server)

Public IPs: 54.80.19.156 Private IPs: 172.31.86.157

Ensure your EC2 instance has the AmazonEC2RoleforAWSCodeDeploy policy attached, allowing CodeDeploy access.



Screenshot of the AWS IAM 'Create role' wizard - Step 2: Add permissions

The screenshot shows the 'Add permissions' step of the IAM role creation wizard. The user is selecting permissions from a list. A search bar at the top left contains 'codede'. The results table has columns for Policy name, Type, and Description. Several policies are selected, including 'AmazonEC2RoleforAWSCodeDeploy', 'AWSCodeDeployFullAccess', and 'AWSCodeDeployReadOnlyAccess'.

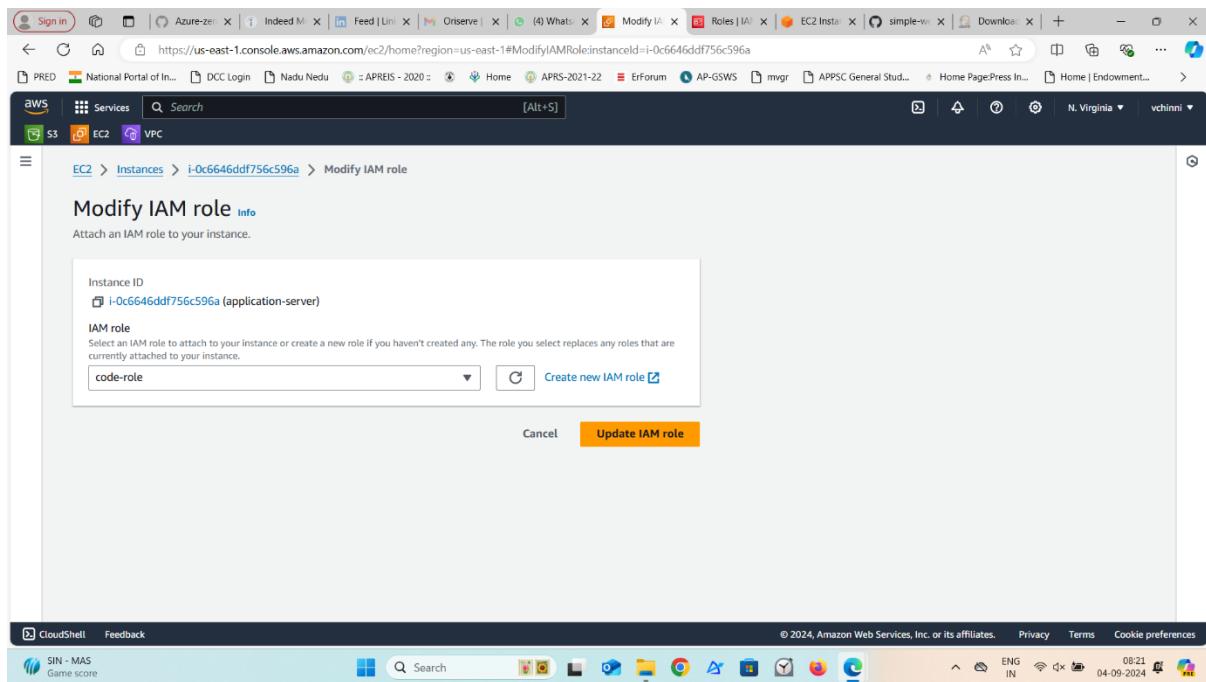
Policy name	Type	Description
<input checked="" type="checkbox"/> AmazonEC2RoleforAWSCodeDeploy	AWS managed	Provides EC2 access to S3 bucket to
<input type="checkbox"/> AmazonEC2RoleforAWSCodeDeployLimited	AWS managed	Provides EC2 limited access to S3 bu
<input type="checkbox"/> AWSCodeDeployDeployerAccess	AWS managed	Provides access to register and deplo
<input checked="" type="checkbox"/> AWSCodeDeployFullAccess	AWS managed	Provides full access to CodeDeploy n
<input type="checkbox"/> AWSCodeDeployReadOnlyAccess	AWS managed	Provides read only access to CodeDe
<input type="checkbox"/> AWSCodeDeployRole	AWS managed	Provides CodeDeploy service access
<input type="checkbox"/> AWSCodeDeployRoleForCloudFormation	AWS managed	Provides CodeDeploy service access

Screenshot of the AWS IAM 'Create role' wizard - Step 3: Name, review, and create

The screenshot shows the 'Name, review, and create' step of the IAM role creation wizard. The 'Role details' section includes a 'Role name' field containing 'code-role' and a 'Description' field containing 'Allows EC2 instances to call AWS services on your behalf.' The 'Step 1: Select trusted entities' section shows a 'Trust policy' field with the value '1-'.

Screenshot of the AWS EC2 Instances page

The screenshot shows the EC2 Instances page for instance 'i-0c6646ddf756c596a'. The instance summary table provides details like Instance ID, Public IP, Instance state, and VPC ID. On the right, a context menu is open under 'Actions' for the instance, showing options like 'Connect', 'Manage instance state', 'Networking', 'Security', and 'Modify IAM role'. The 'Security' option is highlighted.



Step 6: Set Up Auto Scaling for Application Servers

Auto Scaling allows your application to automatically scale up or down based on demand, ensuring availability and performance while optimizing costs.

1. Create a Launch Template

A Launch Template defines the configuration of your EC2 instances, including AMI, instance type, security groups, and key settings. This template will be used to create instances in your Auto Scaling Group.

Steps to Create a Launch Template:

1. **Navigate to the EC2 Launch Templates:**
 - o Go to the AWS EC2 Console.
 - o On the left panel, click on **Launch Templates** under **Instances**.
2. **Create a New Launch Template:**
 - o Click on **Create launch template**.
 - o **Template Name:** Enter a descriptive name
 - o **Template Version Description:** Optionally, add a version description.
3. **Configure the Template Details:**
 - o **AMI (Amazon Machine Image):** Choose the AMI used for your application servers, such as Ubuntu 20.04 LTS.
 - o **Instance Type:** Select an instance type, such as t2.micro or another suitable type based on your needs.
 - o **Key Pair:** Choose the key pair you use for SSH access.

- **Security Groups:** Select the security group that allows HTTP (port 80) and SSH (port 22).
- **IAM Instance Profile:** Choose the IAM role that has permissions for AWS CodeDeploy (AmazonEC2RoleforAWSCodeDeploy).

4. Create the Template:

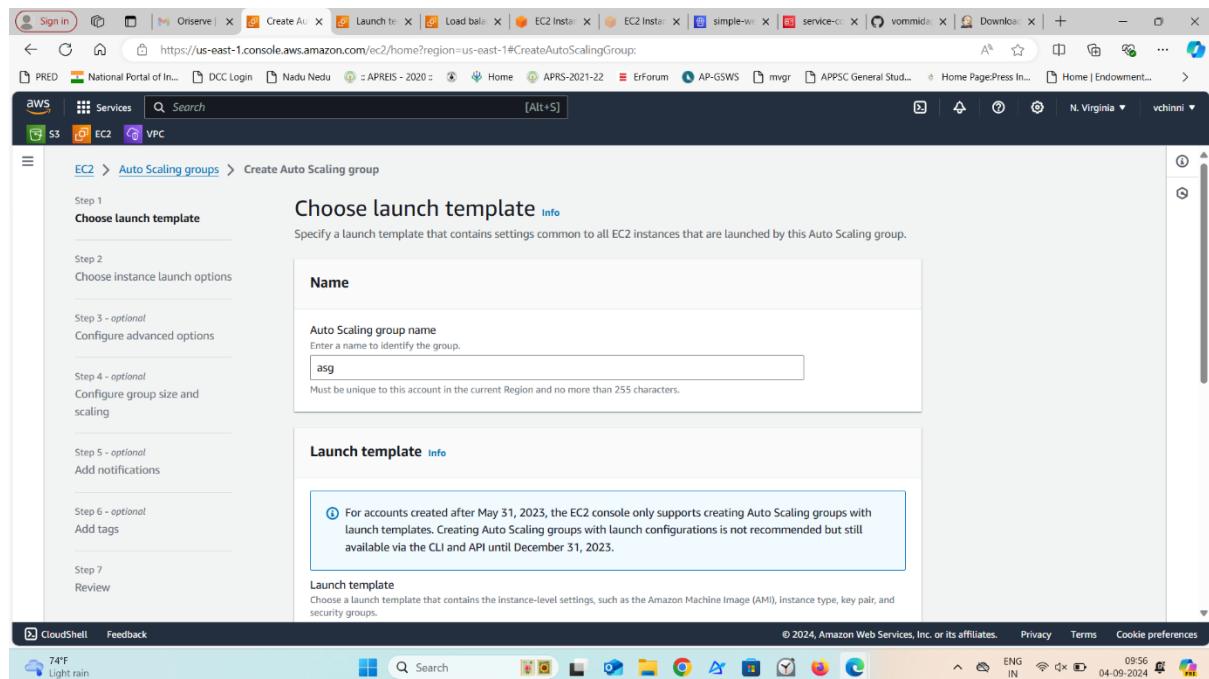
- Review the settings and click **Create launch template**.

Step 7: Create an Auto Scaling Group

The Auto Scaling Group uses the Launch Template to automatically manage the number of instances based on defined scaling policies.

Steps to Create an Auto Scaling Group:

1. **Navigate to Auto Scaling Groups:**
 - Go to Auto Scaling Groups in the AWS EC2 Console.
2. **Create Auto Scaling Group:**
 - Click on **Create Auto Scaling group**.
3. **Configure Group Details:**
 - **Auto Scaling Group Name:** Enter a name
 - **Launch Template:** Select the Launch Template you created



4. Configure Network:

- **VPC:** Choose the same VPC where your application servers will run.
- **Subnets:** Select the subnets where the instances should be launched.

5. Configure Scaling Policies:

- **Desired Capacity:** Set the initial number of instances
- **Minimum Capacity:** Set the minimum number of instances
- **Maximum Capacity:** Set the maximum number of instances

6. Add Scaling Policies:

- **Target Tracking Scaling Policy:** Add a scaling policy based on CPU utilization or another metric.
- For example, set the scaling policy to maintain the average CPU utilization at 50%. The Auto Scaling Group will automatically add or remove instances to maintain this target.

7. Create the Auto Scaling Group:

- Review all settings and click **Create Auto Scaling group**.

Step 8: Create or Select an IAM Role for CodeDeploy

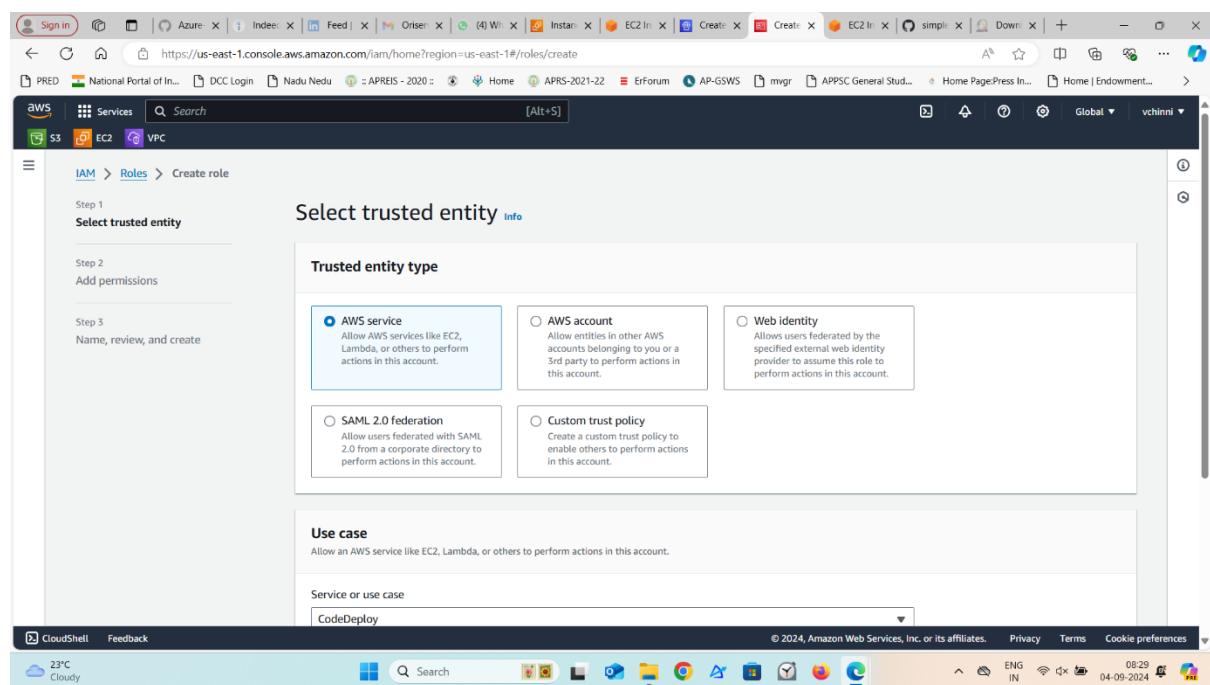
Creating a New IAM Role

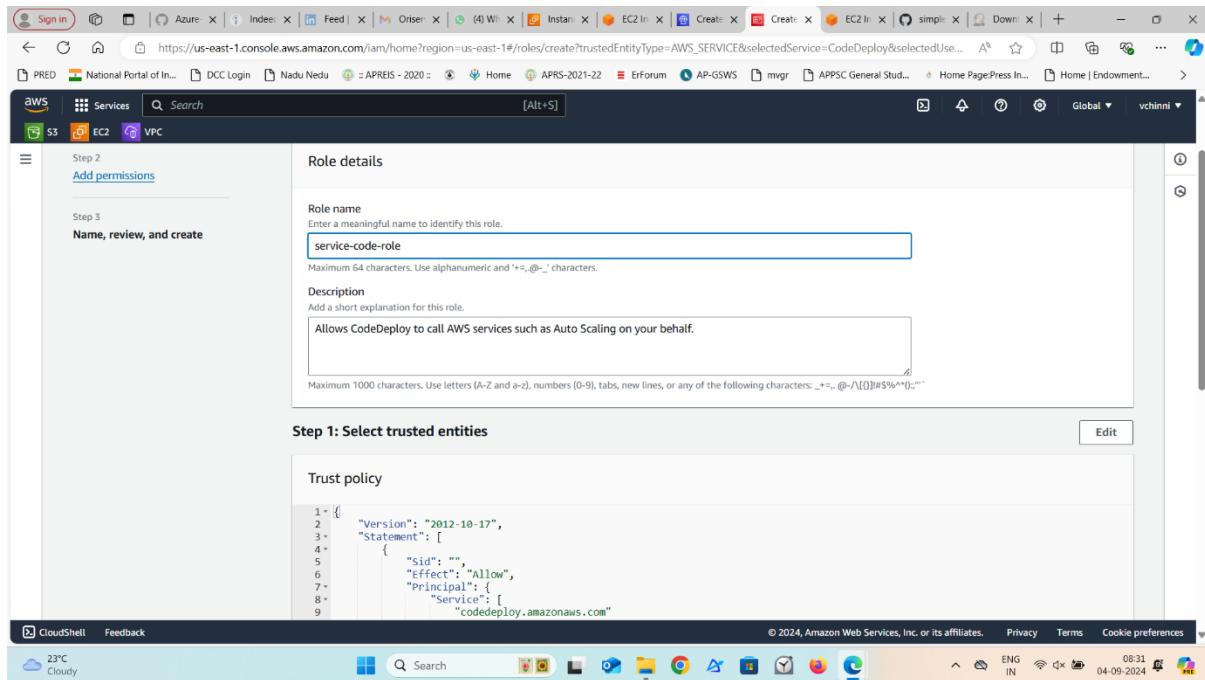
1. Navigate to the IAM Console:

Go to the IAM Management Console.

2. Create a New Role:

- Click on "Roles" in the sidebar.
- Click "Create role".





3.

4. Select Trusted Entity Type:

- Choose "AWS service" as the trusted entity type.
- Select "CodeDeploy" from the list of services.

5. Attach Permissions Policies:

- Attach the "AWSCodeDeployRole" policy, which provides the necessary permissions for CodeDeploy. You can also attach other policies as needed, such as AmazonEC2ReadOnlyAccess if you need read-only access to EC2 resources.

6. Review and Create Role:

- Click "Next: Tags" (optional) to add any tags if needed.
- Click "Next: Review".
- Enter a name for the role (e.g., CodeDeployRole).
- Click "Create role".

7. Attach the Role to CodeDeploy:

- Return to the CodeDeploy Console.
- Navigate to your application and click "Create deployment group".
- In the "Service Role" field, select the role you just created (e.g., CodeDeployRole).

Selecting an Existing IAM Role

If you already have a suitable IAM role:

1. Navigate to the CodeDeploy Console:

Go to the CodeDeploy Console.

2. Create Deployment Group:

- Click on the application you created (e.g., simple-web-app).
- Click "Create deployment group".

3. Select Existing Role:

- In the "Service Role" dropdown, select the IAM role that has the required permissions for CodeDeploy. Ensure that this role has the **AWSCodeDeployRole** policy attached, and any additional policies needed to interact with your EC2 instances and other AWS services.

4. Complete the Configuration:

- Continue with the remaining configuration for your deployment group, such as choosing the deployment type and environment configuration.
- Click "Create deployment group" to finish setting up.

Step 9:

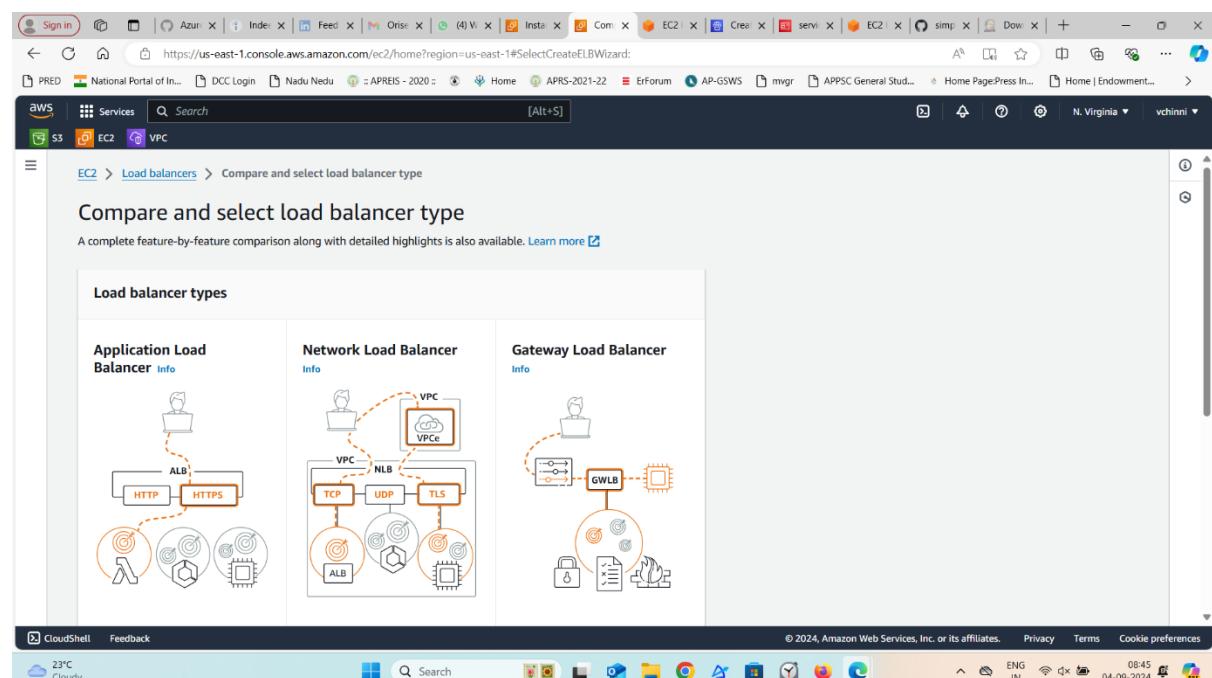
1. Create an Application Load Balancer (ALB)

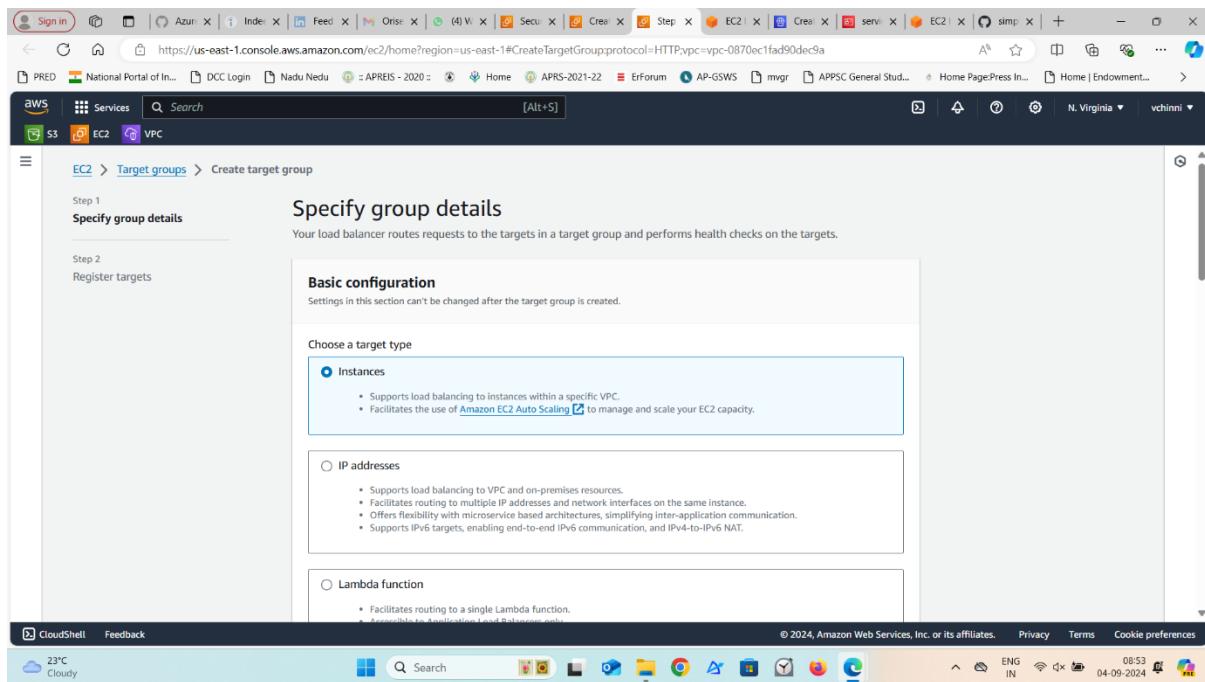
1.1 Navigate to the Load Balancer Console:

- Go to the EC2 Console.
- In the left-hand menu, under "Load Balancing", click "Load balancers".

1.2 Create a New Load Balancer:

- Click "Create Load Balancer".
- Choose "Application Load Balancer" (recommended for HTTP/HTTPS traffic).





1.3 Configure the Load Balancer:

- **Basic Configuration:**
 - **Name:** Enter a name for your load balancer.
 - **Scheme:** Choose "Internet-facing" if your application is public or "Internal" for internal use.
 - **IP address type:** Choose "ipv4".
- **Listeners:** Configure listeners (usually HTTP on port 80 or HTTPS on port 443).
- **Availability Zones:** Select the Availability Zones where your EC2 instances are located.

1.4 Configure Security Settings (for HTTPS):

- **Choose a Security Policy:** If using HTTPS, select an existing SSL certificate or create a new one.
- **Configure Security Groups:** Ensure the security group allows traffic on the ports you specified (e.g., port 80 for HTTP, port 443 for HTTPS).

1.5 Configure Routing:

- **Create a Target Group:**
 - **Target Group Name:** Enter a name for your target group.
 - **Target Type:** Choose "Instance" (if your targets are EC2 instances).

Target group name
target

Protocol : Port
Choose a protocol for your target group that corresponds to the Load Balancer type that will route traffic to it. Some protocols now include anomaly detection for the targets and you can set mitigation options once your target group is created. This choice cannot be changed after creation.

HTTP 80 1-65535

IP address type
Only targets with the indicated IP address type can be registered to this target group.

IPv4 Each instance has a default network interface (eth0) that is assigned the primary private IPv4 address. The instance's primary private IPv4 address is the one that will be applied to the target.

IPv6 Each instance you register must have an assigned primary IPv6 address. This is configured on the instance's default network interface (eth0). [Learn more](#)

VPC
Select the VPC with the instances that you want to include in the target group. Only VPCs that support the IP address type selected above are available in this list.

vpc-0870ec1fad90dec9a
IPv4 VPC CIDR: 172.31.0.0/16

Protocol version

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 23°C Cloudy ENG IN 08:54 04-09-2024

- **Protocol:** Choose HTTP or HTTPS, depending on your setup.
- **Port:** Enter the port where your application is running on the EC2 instances (e.g., port 80).

1.6 Register Targets:

- Register the EC2 instances you want to route traffic to.

Step 1
Specify group details

Step 2
Register targets

Register targets
This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.

Available instances (1/1)

Instance ID	Name	State	Security groups
i-0c6646ddf756c596a	application-server	Running	launch-wizard-13

1 selected

Ports for the selected instances
Ports for routing traffic to the selected instances.

80 1-65535 (separate multiple ports with commas)

Include as pending below

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences Current temp Near record 08:57 04-09-2024

Successfully created the target group: target. Anomaly detection is automatically applied to all registered targets. Results can be viewed in the Targets tab.

target

Details

Target type Instance	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC vpc-0870ec1fad90dec9a
IP address type IPv4	Load balancer None associated		
0 Total targets	0 Healthy	0 Unhealthy	0 Unused
	0 Anomalous		0 Initial
			0 Draining

Targets | **Monitoring** | **Health checks** | **Attributes** | **Tags**

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Create Application Load Balancer

The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, microservices, and containers, based on request attributes. When the load balancer receives a connection request, it evaluates the listener rules in priority order to determine which rule to apply, and if applicable, it selects a target from the target group for the rule action.

How Application Load Balancers work

Basic configuration

Load balancer name
Name must be unique within your AWS account and can't be changed after the load balancer is created.

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Scheme [Info](#)
Scheme can't be changed after the load balancer is created.
 Internet-facing
An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. [Learn more](#)
 Internal
An internal load balancer routes requests from clients to targets using private IP addresses. Compatible with the IPv4 and Dualstack IP address types.

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Load balancer IP address type [Info](#)
Select the front-end IP address type to assign to the load balancer. The VPC and subnets mapped to this load balancer must include the selected IP address types. Public IPv4 addresses have an additional cost.
 IPv4
Includes only IPv4 addresses.
 Dualstack
Includes IPv4 and IPv6 addresses.
 Dualstack without public IPv4
Includes a public IPv6 address, and private IPv4 and IPv6 addresses. Compatible with [internet-facing](#) load balancers only.

Network mapping [Info](#)
The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

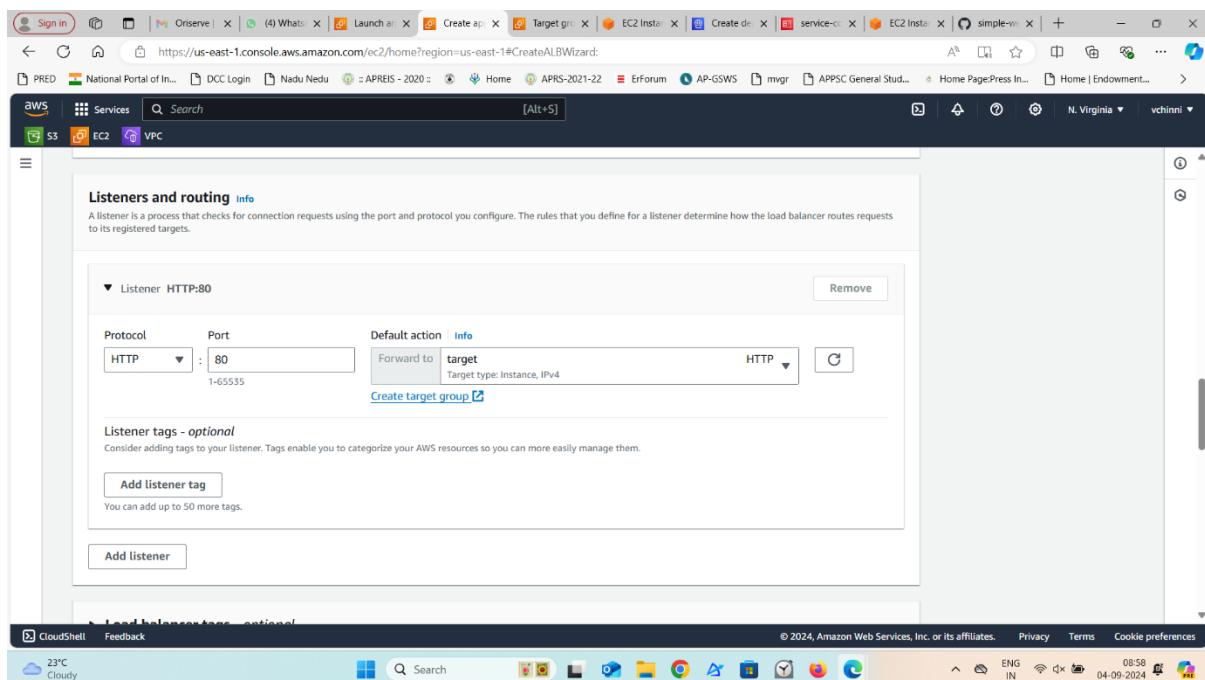
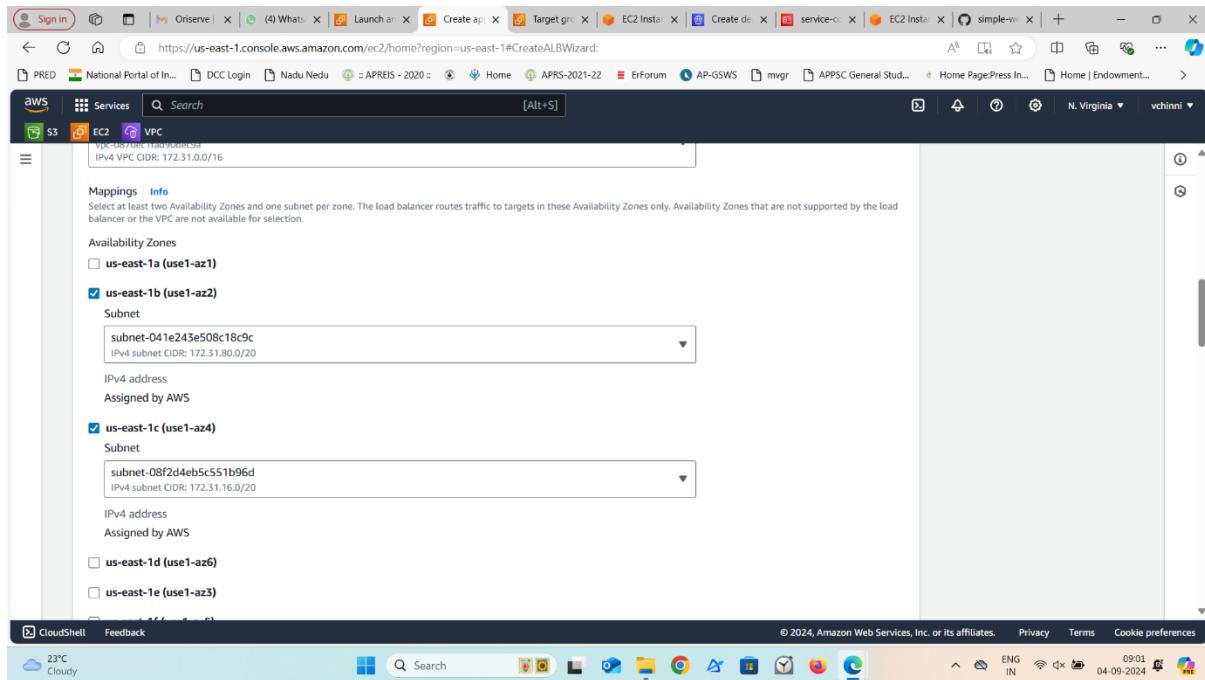
VPC [Info](#)
The load balancer will exist and scale within the selected VPC. The selected VPC is also where the load balancer targets must be hosted unless routing to Lambda or on-premises targets, or if using VPC peering. To confirm the VPC for your targets, view [target groups](#). For a new VPC, [create a VPC](#).

[vpc-0870ec1fad90dec9a](#)
IPv4 VPC CIDR: 172.31.0.0/16

Mappings [Info](#)
Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.

Availability Zones
 us-east-1a (use1-az1)

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



1.7 Review and Create:

- Review your configuration and click "Create".

2. Create a Target Group

2.1 Navigate to the Target Groups Console:

- Go to the EC2 Console.

- In the left-hand menu, under "**Load Balancing**", click "**Target Groups**".

2.2 Create a New Target Group:

- Click "**Create target group**".
- **Target Type:** Choose "Instance".
- **Protocol:** Choose HTTP or HTTPS.
- **Port:** Enter the port where your application is running (e.g., port 80).

2.3 Configure Health Checks:

- Configure health checks based on the endpoint that determines the health of your instances.

2.4 Register Targets:

- Register your EC2 instances with the target group.

2.5 Review and Create:

- Review your configuration and click "**Create**".

3. Integrate Load Balancer and Target Group with CodeDeploy

3.1 Navigate to CodeDeploy Console:

- Go to the CodeDeploy Console.

3.2 Edit/Create Deployment Group:

- Select your application (e.g., simple-web-app).

The screenshot shows a browser window with the AWS CodeDeploy console URL: https://us-east-1.console.aws.amazon.com/codesuite/codedeploy/application/new?region=us-east-1. The page title is 'Create application'. The main content area is titled 'Application configuration' and contains the following fields:

- Application name:** simple-web-app
- Compute platform:** EC2/On-premises
- Tags:** (button labeled 'Add tag')

At the bottom right of the dialog is a large orange 'Create application' button. Below the dialog, the browser's address bar shows the full URL, and the status bar indicates the date and time as 04-09-2024 08:26.

Screenshot of the AWS CodeDeploy console showing the 'Create deployment group' wizard.

Application
Application: simple-web-app
Compute type: EC2/On-premises

Deployment group name
Enter a deployment group name:
simple-web-app
100 character limit

Service role

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 23°C Cloudy Search ENG IN 08:39 04-09-2024

Screenshot of the AWS CodeDeploy console showing the 'Create deployment group' wizard.

Service role
Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.
Q service-code-role

Deployment type
Choose how to deploy your application

In-place
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

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.

Environment configuration
Select any combination of Amazon EC2 Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add to this deployment.

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 23°C Cloudy Search ENG IN 08:39 04-09-2024

Screenshot of the AWS CodeDeploy console showing the 'Create deployment group' wizard.

Environment configuration
Select any combination of Amazon EC2 Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add to this deployment.

Amazon EC2 Auto Scaling groups

Amazon EC2 instances
1 unique matched instance. [Click here for details](#)

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.

Tag group 1

Key	Value - optional
Q Name	Q application-server

Add tag Remove tag

+ Add tag group

On-premises instances

Matching instances

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 23°C Cloudy Search ENG IN 08:40 04-09-2024

Deployment settings

Deployment configuration

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.

CodeDeployDefault.AllAtOnce or Create deployment configuration

Load balancer

Select a load balancer to manage incoming traffic during the deployment process. The load balancer blocks traffic from each instance while it's being deployed to and allows traffic to it again after the deployment succeeds.

Enable load balancing

Load balancer type

Application Load Balancer or Network Load Balancer

Classic Load Balancer

▶ Advanced - optional

Create deployment group

Service role

Enter a service role

Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.

arn:aws:iam::961341527605:role/service-code-role

Deployment type

Choose how to deploy your application

In-place

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

Blue/green

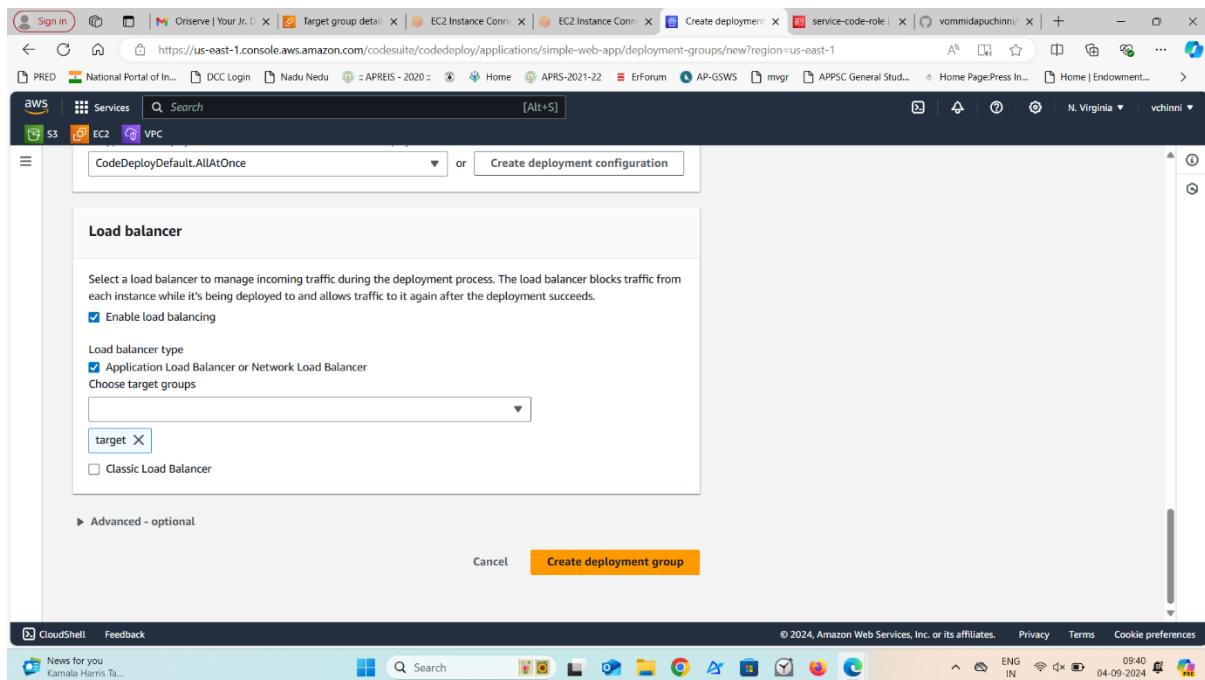
Replaces instances in the deployment group with new instances and deploys the latest application revision to them. After instances in the replacement environment are registered with a load balancer, instances from the original environment are deregistered and can be terminated.

Environment configuration

- Click "Create deployment group" or select an existing one to edit.

3.3 Configure Load Balancer:

- In the "Load Balancer" section:
 - Load Balancer Type:** Choose "Application Load Balancer" (or "Network Load Balancer" if that's what you set up).
 - Load Balancer:** Select the load balancer you created.
 - Target Group:** Select the target group associated with the load balancer.



3.4 Save Configuration:

- Save your deployment group configuration.

Step 10: Configure the Jenkins Pipeline

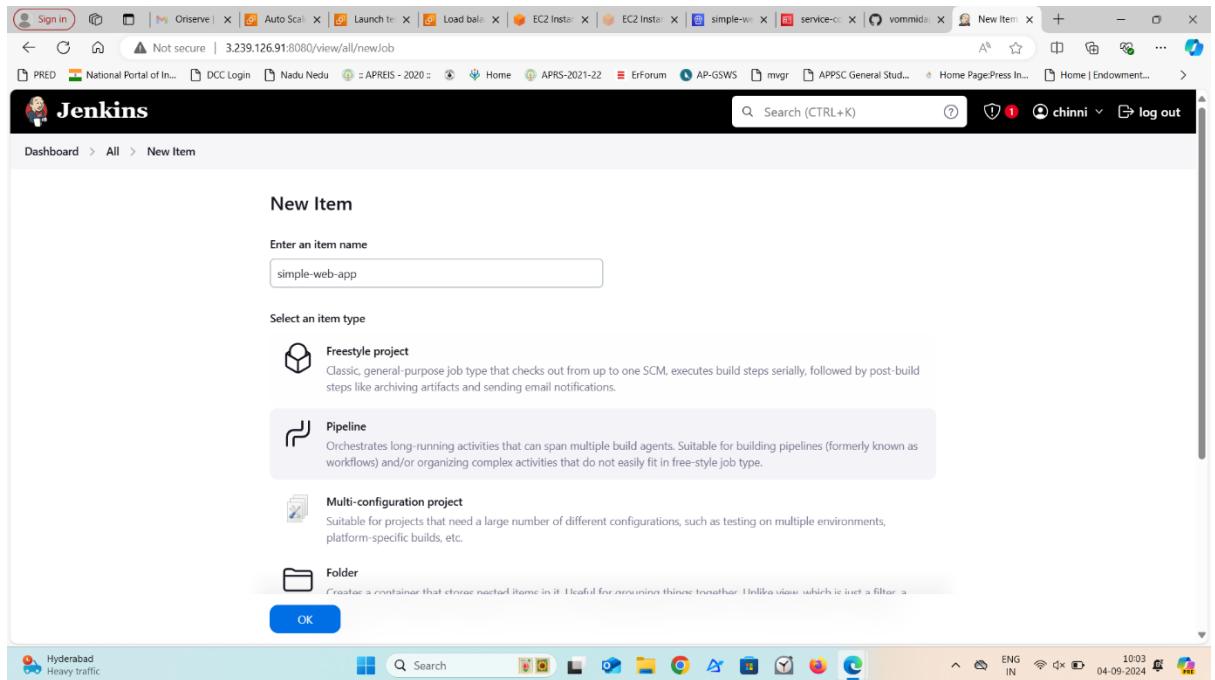
1. Create a New Pipeline Job in Jenkins:

1. Access Jenkins Dashboard:

- Log into your Jenkins server.
- Click on "New Item" in the left sidebar.

2. Create a New Pipeline Job:

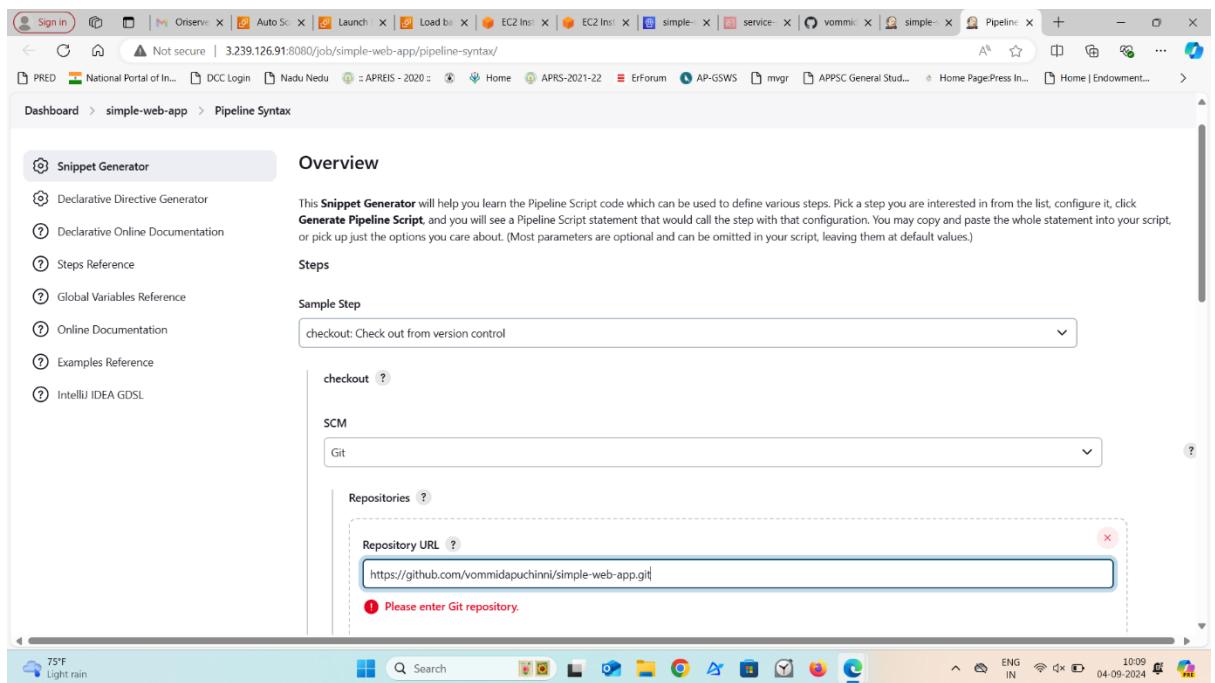
- Enter the name: simple-web-app-pipeline.
- Select "Pipeline" as the job type.
- Click OK.



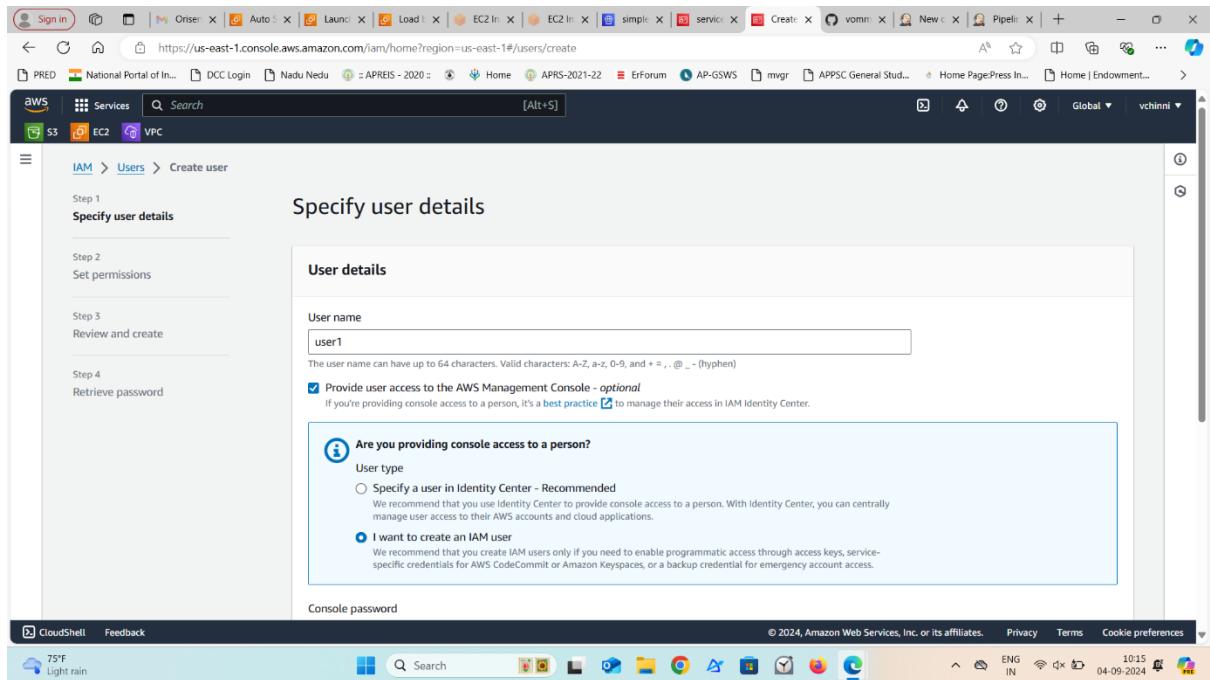
2. Create the Jenkinsfile:

1. Add a Jenkinsfile to Your GitHub Repository:

- o Create a file named Jenkinsfile in the root of your GitHub repository.



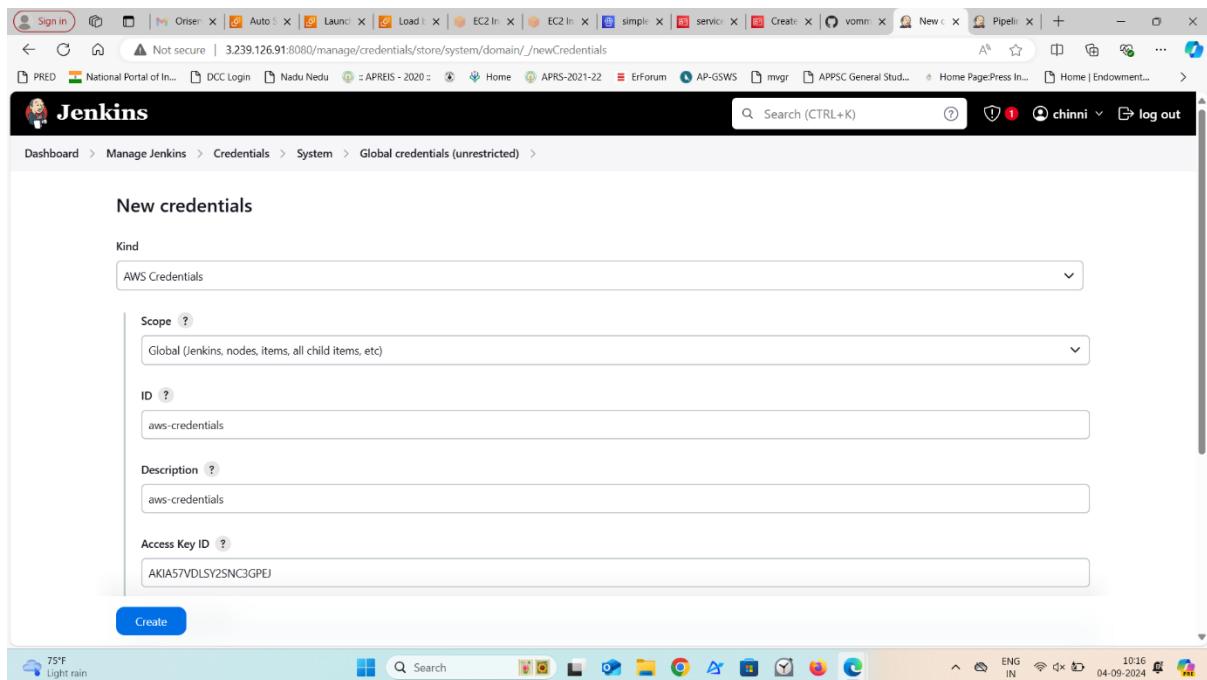
Step 11: Add AWS Credentials in Jenkins



After the AWS Credentials plugin is installed:

1. Add Credentials:

- Go back to **Manage Jenkins → Manage Credentials**.
- Click on the relevant domain (e.g., **Global**).
- Click **Add Credentials**.



2. Select the Correct Credential Type:

- In the **Kind** dropdown, you should now see "**AWS Credentials**".

- Enter your **AWS Access Key ID** and **Secret Access Key**.
- Provide an **ID** and a **description** to identify these credentials.

Create an S3 Bucket

1. Open the S3 Console:

- Go to the Amazon S3 console.

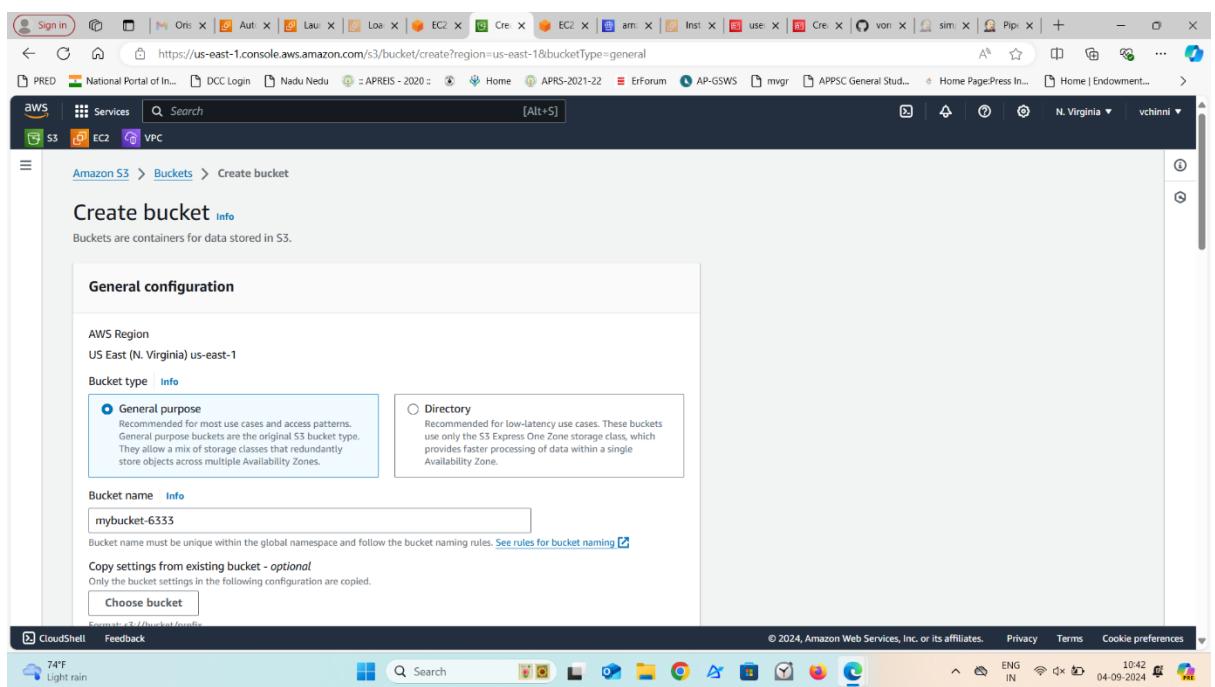
2. Create a New Bucket:

- Click on “Create bucket.”
- Enter a unique name for your bucket.
- Choose the region where you want the bucket to be created.
- Configure options as needed and click “Create bucket.”

3. Upload Deployment Bundle:

- Go to the newly created bucket.
- Click “Upload” and add your deployment bundle (simple-web-app.zip).
- Click “Upload” to complete the process.

4. aws s3 cp ~/Desktop/simple-web-app.zip s3://your-bucket-name/



The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with various AWS services like EC2, Lambda, and CloudWatch. Below it, the main header says "Amazon S3 > Buckets". A banner at the top of the page says "Account snapshot - updated every 24 hours" and "Storage lens provides visibility into storage usage and activity trends". There are tabs for "General purpose buckets" and "Directory buckets", with "General purpose buckets" selected. A search bar allows finding buckets by name. A table lists one bucket: "mybucket-6333". The table columns include Name, AWS Region, IAM Access Analyzer, and Creation date. The bucket was created in "US East (N. Virginia) us-east-1" on September 4, 2024, at 10:42:26 (UTC+05:30). Below the table, the "Objects" tab is selected in the sub-navigation bar. The "Objects" section shows two items: "deployment_bundle.zip" and "simple-web-app.zip", both of which are zip files. The "Actions" button is highlighted in orange.

Update the Jenkinsfile to Use the Credentials:

```
pipeline {
    agent any
    stages {
        stage('Checkout') {
            steps {
```

```

// Checkout code from GitHub repository
checkout scmGit(branches: [[name: '/main']]),
        extensions: [],
        userRemoteConfigs: [[url:
'https://github.com/vommidapuchinni/simple-web-app.git']])
    }
}
stage('Build') {
    steps {
        // Placeholder for build steps
        sh 'echo "Build step (if required)"'
    }
}
stage('Deploy to AWS CodeDeploy') {
    steps {
        // Use AWS credentials from Jenkins
        withCredentials([aws(credentialsId: 'aws-credentials')]) {
            sh ""
            aws deploy create-deployment \
                --application-name simple-web-app \
                --deployment-group-name simple-web-app-deployment-group \
                --deployment-config-name CodeDeployDefault.AllAtOnce \
                --region us-east-1 \
                --s3-location bucket=mybucket,key=simple-web-
app.zip,bundleType=zip
            ""
        }
    }
}
}

```

Screenshot of Jenkins Pipeline View for 'simple-web-app' showing Stage View and Build History.

Stage View:

	checkout	Build	Deploy to AWS CodeDeploy
Average stage times:	515ms	313ms	1s
Sep 04 (No Changes)	429ms	380ms	2s
Sep 04 (No Changes)	602ms failed	247ms failed	81ms failed

Build History:

- Last build (#6), 3 hr 0 min ago
- Last stable build (#6), 3 hr 0 min ago
- Last successful build (#6), 3 hr 0 min ago

Console Output:

```

Started by user chinni
[Pipeline] Start of Pipeline
Kicking off Jenkins in /var/lib/jenkins/workspace/simple-web-app
[Pipeline] stage
[Pipeline] {
[Pipeline]   stage('Checkout') {
[Pipeline]     steps {
[Pipeline]       checkout scm(itm: [[name: 'main']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/vommida/puchinni']])
[Pipeline]     }
[Pipeline]   }
[Pipeline]   stage('Build') {
[Pipeline]     steps {
[Pipeline]       sh 'echo "Build step (if required)"'
[Pipeline]     }
[Pipeline]   }
[Pipeline]   stage('Deploy to AWS CodeDeploy') {
[Pipeline]     steps {
[Pipeline]       withCredentials([[class: 'AmazonWebServicesCredentialsBinding', credentialsId: 'aws-credentials']]) {
[Pipeline]         sh ...
[Pipeline]         aws codedeploy create-deployment \
[Pipeline]           --application-name simple-web-app \
[Pipeline]           --deployment-group-name simple-web-app \
[Pipeline]           --deployment-config-name CodeDeployDefault.AllAtOnce \
[Pipeline]           --s3-location bucket=mybucket-6333,key=simple-web-app.zip,bundleType=zip
[Pipeline]       }
[Pipeline]     }
[Pipeline]   }
[Pipeline] }
[Pipeline] End of Pipeline

```

Configuration Pipeline Script:

```

1 pipeline {
2   agent any
3   stages {
4     stage('Checkout') {
5       steps {
6         checkout scm(itm: [[name: 'main']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/vommida/puchinni']])
7       }
8     }
9     stage('Build') {
10       steps {
11         sh 'echo "Build step (if required)"'
12       }
13     }
14     stage('Deploy to AWS CodeDeploy') {
15       steps {
16         withCredentials([[class: 'AmazonWebServicesCredentialsBinding', credentialsId: 'aws-credentials']]) {
17           sh ...
18           aws codedeploy create-deployment \
19             --application-name simple-web-app \
20             --deployment-group-name simple-web-app \
21             --deployment-config-name CodeDeployDefault.AllAtOnce \
22             --s3-location bucket=mybucket-6333,key=simple-web-app.zip,bundleType=zip
23         }
24       }
25     }
26   }
27 }
28
29

```

Verify CodeDeploy Agent Installation

- Once the installation is complete, you can verify that the CodeDeploy agent is running on your instances:
 - SSH into your EC2 instance** and run: `sudo service codedeploy-agent status`

Install awscli on Jenkins server

```
chmod +x restart_server.sh  
sudo ./restart_server.sh
```

edit trust relationship in iam role for code-role

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "Service": [  
                    "ec2.amazonaws.com",  
                    "codedeploy.amazonaws.com"  
                ]  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

Step 12:

Check Console Navigation

- Go to CodeDeploy Console:**
 - Navigate to the AWS CodeDeploy Console.
- Select Application:**
 - Click on your application name (e.g., simple-web-app).
- Find Deployment Options:**
 - Look for "**Deployments**" in the left-hand menu or under the application details.
- Locate Create Deployment:**

- Inside the "Deployments" section or tab, you should see options like "Create deployment" or "Create new deployment".

2. Check Permissions

Ensure you have the necessary IAM permissions to create deployments:

1. IAM Role/Policy:

- Make sure the IAM user or role you're using has the codedeploy:CreateDeployment permission.

Application and Deployment Group

- **Application:** Select simple-web-app.
- **Deployment Group:** Select simple-web-app.

2. Compute Platform

- **Compute platform:** Ensure EC2/On-premises is selected.

3. Deployment Type

- **Deployment type:** Choose In-place if you want to update the existing instances with the latest application revision.

4. Revision Type

- **Revision type:** Choose My application is stored in Amazon S3.

5. Revision Location

- **Revision location:** Enter the S3 bucket URL where your deployment bundle is stored. Example: s3://mybucket-6333/deployment-bundle.zip.

6. Revision File Type

- **Revision file type:** Confirm or select .zip if your deployment bundle is a zip file.

The screenshot shows the AWS CloudWatch CodeDeploy console with the following details:

Deployment details:

- Application: simple-web-app
- Deployment ID: d-DUVA9DF68
- Status: Succeeded
- Deployment configuration: CodeDeployDefault.AllAtOnce
- Deployment group: simple-web-app
- Initiated by: User action
- Deployment description: -

Revision details:

- Revision location: s3://mybucket-6333/deployment-bundle.zip
- Revision created: Sep 4, 2024 12:33 PM (UTC+5:30)
- Revision description: Application revision registered by Deployment ID: d-B83HFPE68

The browser address bar shows the URL: https://us-east-1.console.aws.amazon.com/codesuite/codedeploy/deployments/d-DUVA9DF68/instances/arn%3Aaws%3Aec2%3Aus-east-1%3A961341527605%3Ain...

The top screenshot shows the AWS CloudWatch Metrics interface for a CodeDeploy deployment. It displays a table of events with columns: Event, Duration, Status, Error code, Start time, and End time. All events show a status of 'Succeeded'.

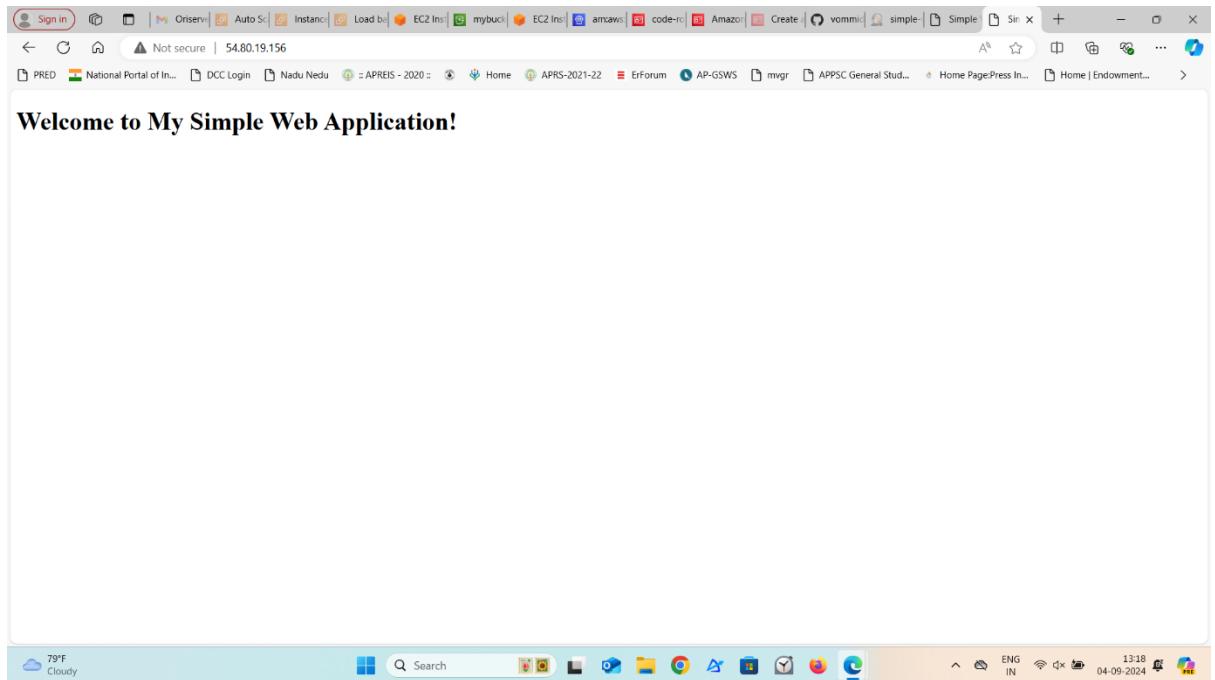
Event	Duration	Status	Error code	Start time	End time
BeforeBlockTraffic	less than one second	Succeeded	-	Sep 4, 2024 1:06 PM (UTC+5:30)	Sep 4, 2024 1:06 PM (UTC+5:30)
BlockTraffic	5 minutes 10 seconds	Succeeded	-	Sep 4, 2024 1:06 PM (UTC+5:30)	Sep 4, 2024 1:11 PM (UTC+5:30)
AfterBlockTraffic	less than one second	Succeeded	-	Sep 4, 2024 1:11 PM (UTC+5:30)	Sep 4, 2024 1:11 PM (UTC+5:30)
ApplicationStop	less than one second	Succeeded	-	Sep 4, 2024 1:11 PM (UTC+5:30)	Sep 4, 2024 1:11 PM (UTC+5:30)
DownloadBundle	less than one second	Succeeded	-	Sep 4, 2024 1:11 PM (UTC+5:30)	Sep 4, 2024 1:11 PM (UTC+5:30)
BeforeInstall	less than one second	Succeeded	-	Sep 4, 2024 1:11 PM (UTC+5:30)	Sep 4, 2024 1:11 PM (UTC+5:30)
Install	less than one second	Succeeded	-	Sep 4, 2024 1:11 PM (UTC+5:30)	Sep 4, 2024 1:11 PM (UTC+5:30)
AfterInstall	less than one second	Succeeded	-	Sep 4, 2024 1:11 PM (UTC+5:30)	Sep 4, 2024 1:11 PM (UTC+5:30)
ApplicationStart	less than one second	Succeeded	-	Sep 4, 2024 1:11 PM (UTC+5:30)	Sep 4, 2024 1:11 PM (UTC+5:30)
ValidateService	less than one second	Succeeded	-	Sep 4, 2024 1:11 PM (UTC+5:30)	Sep 4, 2024 1:11 PM (UTC+5:30)
BeforeAllowTraffic	less than one second	Succeeded	-	Sep 4, 2024 1:11 PM (UTC+5:30)	Sep 4, 2024 1:11 PM (UTC+5:30)
AllowTraffic	2 minutes 35 seconds	Succeeded	-	Sep 4, 2024 1:11 PM (UTC+5:30)	Sep 4, 2024 1:14 PM (UTC+5:30)
AfterAllowTraffic	less than one second	Succeeded	-	Sep 4, 2024 1:14 PM (UTC+5:30)	Sep 4, 2024 1:14 PM (UTC+5:30)

The bottom two screenshots show the AWS EC2 Dashboard. The left one shows the 'Instances' section with four running t2.micro instances. The right one shows a detailed view of the first instance, with a 'Select an instance' dropdown open.

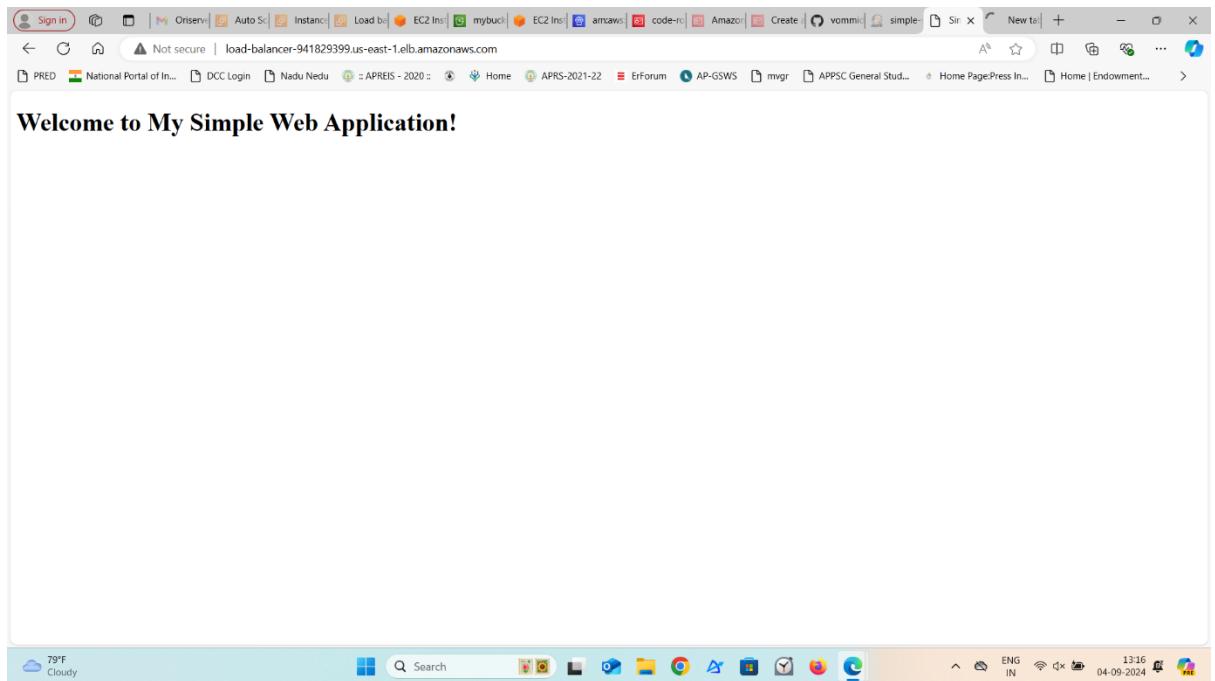
Access via Public IP or Domain Name: For EC2 Instances:

- **Public IP Address:**

- If your EC2 instance has a public IP address, you can access your application by entering this IP address in your web browser.
- Example: <http://<public-ip-address>>

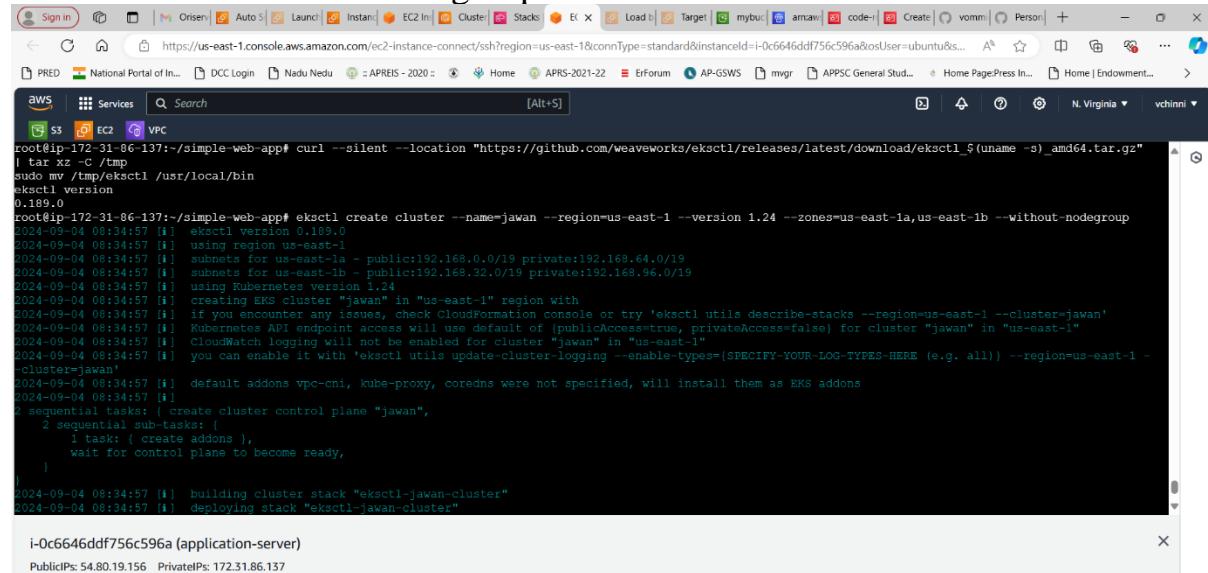


-
- **Elastic Load Balancer (ELB):**
 - If you are using an Elastic Load Balancer (ELB), you can access your application using the ELB DNS name.
 - You can find this DNS name in the EC2 Management Console under "Load Balancers."



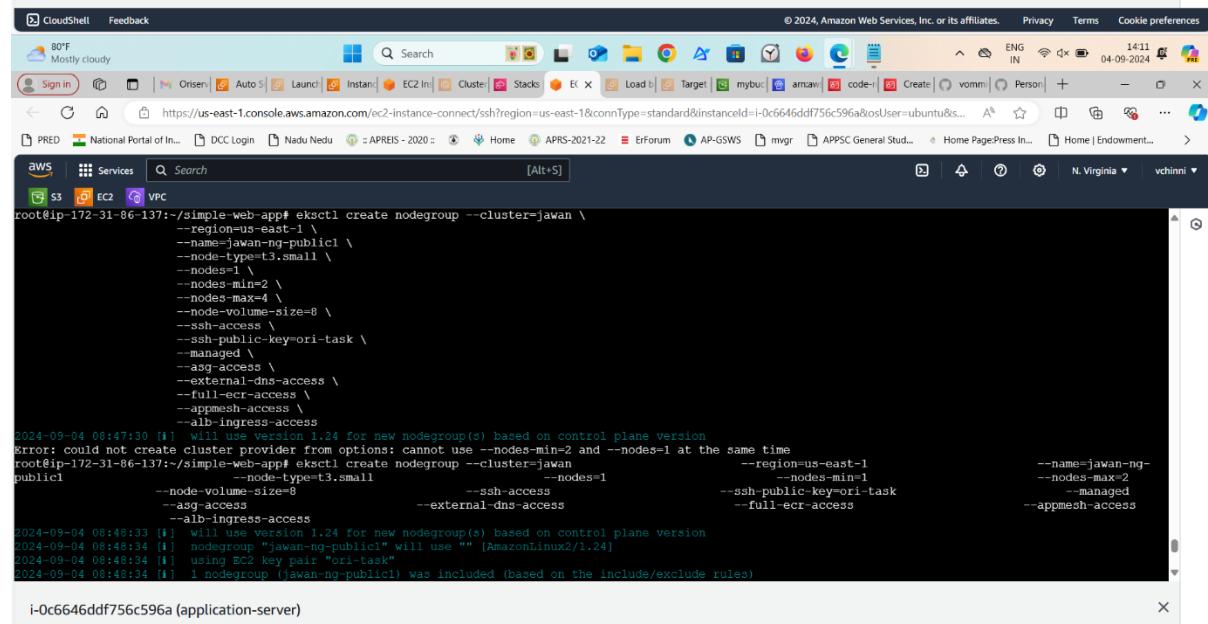
Deploying and accessing with Kubernetes:

Create EKS cluster and node group



root@ip-172-31-86-137:~/simple-web-app# curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_\$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
sudo mv /tmp/eksctl /usr/local/bin
eksctl version
0.189.0
root@ip-172-31-86-137:~/simple-web-app# eksctl create cluster --name=jawan --region=us-east-1 --version 1.24 --zones=us-east-1a,us-east-1b --without-nodegroup
2024-09-04 08:34:57 [!] using region us-east-1
2024-09-04 08:34:57 [!] using subnets for us-east-1a - public:192.168.0.0/19 private:192.168.64.0/19
2024-09-04 08:34:57 [!] subnets for us-east-1b - public:192.168.32.0/19 private:192.168.96.0/19
2024-09-04 08:34:57 [!] using Rubernetes version 1.24
2024-09-04 08:34:57 [!] creating EKS cluster "jawan" in "us-east-1" region with
2024-09-04 08:34:57 [!] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-east-1 --cluster=jawan'
2024-09-04 08:34:57 [!] Kubernetes API endpoint access will use default of (publicAccess=true, privateAccess=false) for cluster "jawan" in "us-east-1"
2024-09-04 08:34:57 [!] CloudWatch logging will not be enabled for cluster "jawan" in "us-east-1"
2024-09-04 08:34:57 [!] you can enable it with 'eksctl utils update-cluster-logging --enable-types=(SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)) --region=us-east-1 --cluster=jawan'
2024-09-04 08:34:57 [!] default addons vpc-cni, kube-proxy, coredns were not specified, will install them as EKS addons
2024-09-04 08:34:57 [!] 2 sequential tasks: (create cluster control plane "jawan",
2024-09-04 08:34:57 [!] 1 task: { create addons },
2024-09-04 08:34:57 [!] wait for control plane to become ready,
2024-09-04 08:34:57 [!])
2024-09-04 08:34:57 [!] building cluster stack "eksctl-jawan-cluster"
2024-09-04 08:34:57 [!] deploying stack "eksctl-jawan-cluster"

i-0c6646ddf756c596a (application-server)
PublicIPs: 54.80.19.156 PrivateIPs: 172.31.86.137



root@ip-172-31-86-137:~/simple-web-app# eksctl create nodegroup --cluster=jawan \
--region=us-east-1 \
--name=jawan-ng-public1 \
--node-type=t3.small \
--nodes=1 \
--nodes-min=2 \
--nodes-max=4 \
--node-volume-size=8 \
--ssh-access \
--ssh-public-key=ori-task \
--managed \
--asg-access \
--external-dns-access \
--full-ecr-access \
--appmesh-access \
--alb-ingress-access
2024-09-04 08:47:30 [!] will use version 1.24 for new nodegroup(s) based on control plane version
Error: could not create cluster provider from options: cannot use --nodes-min=2 and --nodes=1 at the same time
root@ip-172-31-86-137:~/simple-web-app# eksctl create nodegroup --cluster=jawan \
--region=us-east-1 \
--node-type=t3.small \
--nodes=1 \
--nodes-min=1 \
--nodes-max=2 \
--ssh-access \
--ssh-public-key=ori-task \
--full-ecr-access \
--appmesh-access \
--alb-ingress-access
2024-09-04 08:48:33 [!] will use version 1.24 for new nodegroup(s) based on control plane version
2024-09-04 08:48:34 [!] nodegroup "jawan-ng-public1" will use "" [AmazonLinux2/1.24]
2024-09-04 08:48:34 [!] using EC2 key pair "ori-task"
2024-09-04 08:48:34 [!] i node-group (jawan-ng-public1) was included (based on the include/exclude rules)

i-0c6646ddf756c596a (application-server)
PublicIPs: 54.80.19.156 PrivateIPs: 172.31.86.137



The screenshot shows the AWS EKS console interface. On the left, there's a sidebar with navigation links like 'Clusters', 'Amazon EKS Anywhere', 'Related services', 'Console settings', 'Documentation', and 'Submit feedback'. The main area is titled 'Clusters (1) info' and displays a table with one row for the cluster 'jawan'. The table columns include 'Cluster name', 'Status', 'Kubernetes version', 'Support period', 'Upgrade policy', and 'Created'. The cluster is listed as 'Active', running '1.24', with 'Extended support until January 31, 2025'. It was created '12 minutes ago'.

The screenshot shows the AWS CloudFormation console interface. On the left, there's a sidebar with navigation links like 'Stacks', 'StackSets', 'Exports', 'Application Composer', 'Registry', 'Public extensions', 'Activated extensions', 'Publisher', 'Spotlight', and 'Feedback'. The main area is titled 'CloudFormation > Stacks' and displays a table with two rows. The first row is for the stack 'eksctl-jawan-nodegroup-jawan-nginx-public1' with status 'CREATE_COMPLETE' and creation time '2024-09-04 14:18:35 UTC+0530'. The second row is for the stack 'eksctl-jawan-cluster' with status 'CREATE_COMPLETE' and creation time '2024-09-04 14:04:57 UTC+0530'. Both stacks are described as EKS Managed Nodes (SSH access: true) [created by eksctl].

Created a docker file

The screenshot shows an AWS CloudShell terminal window. The user has run the command `cd /tmp` and then `vi Dockerfile` to edit a Dockerfile. The terminal then runs `docker build -t chinnill/web-app:latest .` which outputs several lines of log indicating the download of various Docker packages. Finally, the user runs `docker run -d -p 80:80 chinnill/web-app` to start a container. The terminal also shows the IP address of the EC2 instance as 54.80.19.156 and the private IP as 172.31.86.137.

```

root@ip-172-31-86-137:~/simple-web-app# vi Dockerfile
root@ip-172-31-86-137:~/simple-web-app# docker build -t chinnill/web-app:latest .
Command 'docker' not found, but can be installed with:
snap install docker
apt install docker.io # version 24.0.5, or
apt install docker # version 24.0.7-Ubuntu4.1
apt install podman-docker # version 4.9.3+ds1~ubuntu0.1
See 'snap info docker' for additional versions.
root@ip-172-31-86-137:~/simple-web-app# apt install docker.io -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
lipp python3-pip
libapparmor2 libautofs-tools libcgroups2 mount | cgroup-lite debootstrap docker-buildx docker-compose-v2 docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 82 not upgraded.
Need to get 76.8 MB of archives.
After this operation, 288 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.amazonaws.com/ubuntu/nobinary/amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-1.ec2.archive.amazonaws.com/ubuntu/main/amd64 bridge-utils amd64 1.7.1-lubuntu2 [33.9 kB]
Get:3 http://us-east-1.ec2.archive.amazonaws.com/ubuntu/nobility/amd64 runc amd64 1.1.12-0ubuntu1.1 [6599 kB]
Get:4 http://us-east-1.ec2.archive.amazonaws.com/ubuntu/nobility/main/amd64 containerd amd64 1.7.12-0ubuntu4.1 [38.6 MB]
Get:5 http://us-east-1.ec2.archive.amazonaws.com/ubuntu/nobility/main/amd64 dns-root-data all 2023112702-willysync1 [4450 B]
Get:6 http://us-east-1.ec2.archive.amazonaws.com/ubuntu/nobility/main/amd64 dnsmasq-base amd64 2.90-2build2 [375 kB]
Get:7 http://us-east-1.ec2.archive.amazonaws.com/ubuntu/nobility/universe/amd64 docker.io amd64 24.0.7-0ubuntu4.1 [29.1 MB]

i-0c6646ddff756c596a (application-server)
PublicIPs: 54.80.19.156 PrivateIPs: 172.31.86.137

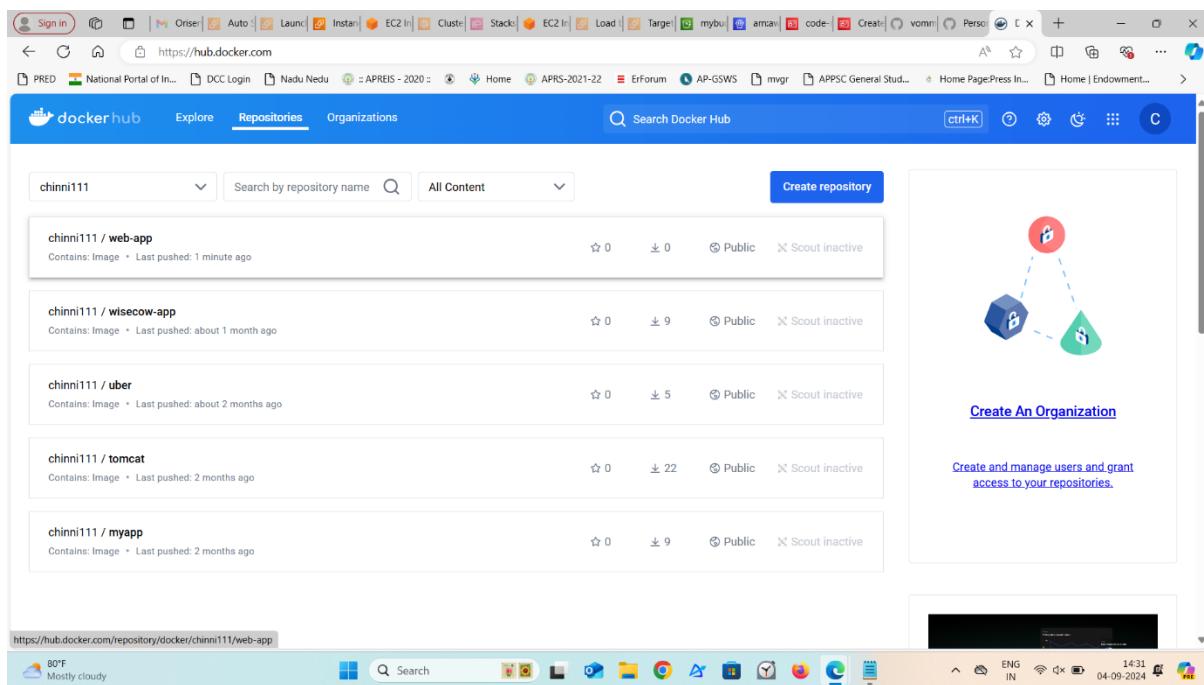
```

Build the docker file

```
aws services search [Alt+S]
SS EC VPC
> 5620E94408e6
Step 5/5 : CMD ["phpapache2","-D", "BACKGROUND"]
--> Running in af1cdbad23ad
Removing intermediate container af1cdbad23ad
> 11bd35cd07cd
Successfully built 11bd35cd07cd
Successfully tagged chinin111/web-app:latest
root@ip-172-31-86-137:~/simple-web-app# docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants better security and is required for organizations using SSO. Learn more at https://docs.docker.com/go/access-tokens/
Username: chinin111
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
root@ip-172-31-86-137:~/simple-web-app# docker push chinin111/web-app:latest
The push refers to repository [docker.io/chinin111/web-app]
ea382b79491: Pushed
7caa52d7fffb3: Pushed
f36fd4bb7334: Mounted from library/ubuntu
latest: digest: sha256:cce7326124c8eb037063e8ddde1b2899973cbdb88140e23ce94b9430a6cdce7e size: 948
root@ip-172-31-86-137:~/simple-web-app# i-0c6646ddf756c596a (application-server)
PublicIPs: 54.80.19.156 PrivateIPs: 172.31.86.137
```

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 80°F Mostly cloudy ENG IN 14:30 04-09-2024

Push to docker hub



Created deployment and service yaml files
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml

check the status of your deployment:

kubectl get deployments kubectl get services

```
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.  
Configure a credential helper to remove this warning. See  
https://docs.docker.com/engine/reference/commandline/login/#credentials-store  
Login Succeeded  
root@ip-172-31-86-137:~/simple-web-app# docker push chinni111/web-app:latest  
The push refers to repository [docker.io/chinni111/web-app]  
ea302fb79491: Pushed  
7caa52d7ff3b: Pushed  
f36fd4bb7334: Mounted from library/ubuntu  
latest: digest: sha256:6ca7326124c29eb0370c3e8ddde1b2899973cbdb80140e23ce94b9430a6cdce7e size: 948  
root@ip-172-31-86-137:~/simple-web-app# vi deployment.yaml  
root@ip-172-31-86-137:~/simple-web-app# vi deployment.yaml  
root@ip-172-31-86-137:~/simple-web-app# vi service.yaml  
root@ip-172-31-86-137:~/simple-web-app# kubectl apply -f deployment.yaml  
kubernetes apply -f service.yaml  
deployment.apps/web-app created  
service/web-app-service created  
root@ip-172-31-86-137:~/simple-web-app# kubectl get deployments  
kubectl get services  
NAME READY UP-TO-DATE AVAILABLE AGE  
web-app 0/1 1 0 6s  
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  
kubernetes ClusterIP 10.100.0.1 <none> 443/TCP 21m  
web-app-service LoadBalancer 10.100.14.97 a4f29d1c666c34e629a017e62a4d623a-1250876994.us-east-1.elb.amazonaws.com 80:31923/TCP 5s  
root@ip-172-31-86-137:~/simple-web-app# ^C  
root@ip-172-31-86-137:~/simple-web-app#  
i-0c6646ddf756c596a (application-server)  
Public IPs: 54.80.19.156 Private IPs: 172.31.86.137
```

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

80°F Mostly cloudy

Sign in Oris... Auto... Launch EC2 In... Clusters EC2 In... Load Target mybu arnava code Create volumn Person chinni

https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instancesv=3;\$case=true%5C.client=false;\$regex=tags:false%5C.client=false

PRED National Portal of In... DCC Login Nadu Nedu APRS-2020 : Home APRS-2021-22 ErForum AP-GWSW mgvr APPSC General Stud... Home Page|Press In... Home | Endowment... N. Virginia vchinni

AWS Services Search [Alt+S]

EC2 VPC

EC2 Dashboard EC2 Global View Events Console-to-Code Instances Instances Instance Types Launch Templates Spot Requests Savings Plans Reserved Instances Dedicated Hosts Capacity Reservations New Images AMIs AMI Catalog Elastic Block Store Volumes

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

80°F Mostly cloudy

Sign in Oris... Auto... Launch EC2 In... Clusters EC2 In... Load Target mybu arnava code Create volumn Person chinni

https://us-east-1.console.aws.amazon.com/ec2/instances?region=us-east-1#Instancesv=5

PRED National Portal of In... DCC Login Nadu Nedu APRS-2020 : Home APRS-2021-22 ErForum AP-GWSW mgvr APPSC General Stud... Home Page|Press In... Home | Endowment... N. Virginia vchinni

AWS Services Search [Alt+S]

Instances (5) Info Last updated less than a minute ago Connect Instance state Actions Launch Instances

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public
application-server	i-0c6646ddf756c596a	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-5
	i-02b380ed8015fb61	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1c	ec2-1
jawan-jawan-ng-pub...	i-07bee9760e6f276f8	Running	t3.small	3/3 checks passed	View alarms +	us-east-1a	ec2-4
	i-0180573e9e60b0771	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-3
jenkins-server	i-091705c286aef69a5	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1f	ec2-1

Select an instance

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

80°F Mostly cloudy

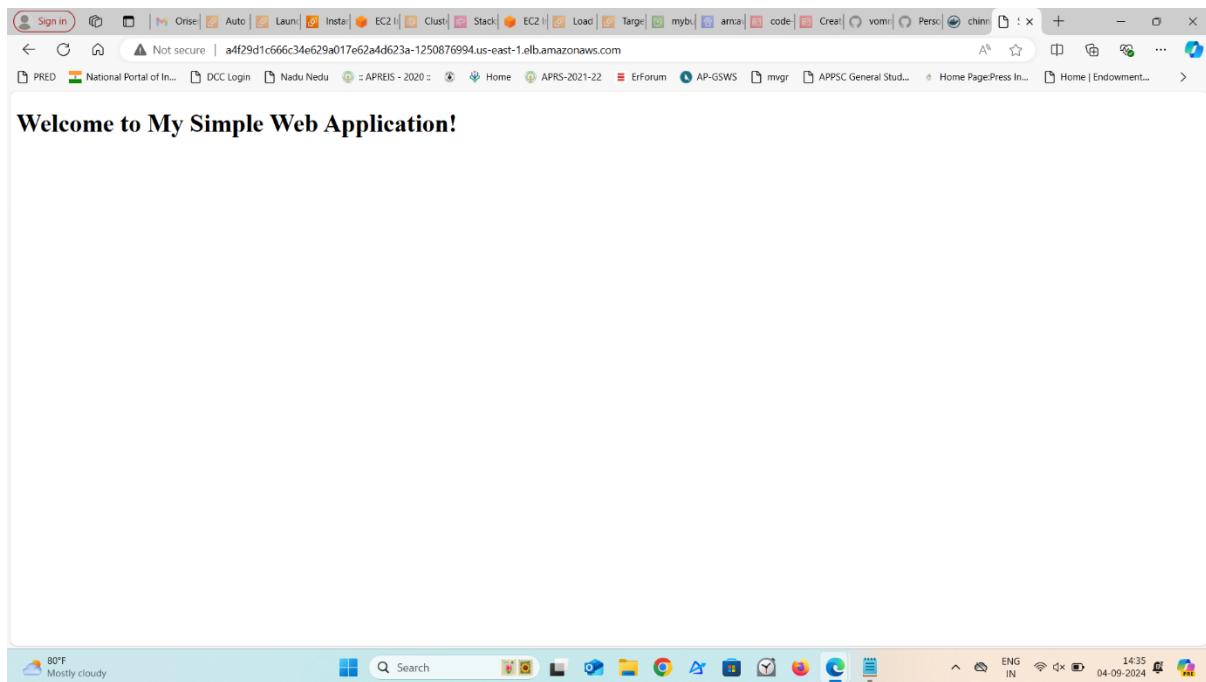
Sign in Oris... Auto... Launch EC2 In... Clusters EC2 In... Load Target mybu arnava code Create volumn Person chinni

https://us-east-1.console.aws.amazon.com/ec2/instances?region=us-east-1#Instancesv=5

PRED National Portal of In... DCC Login Nadu Nedu APRS-2020 : Home APRS-2021-22 ErForum AP-GWSW mgvr APPSC General Stud... Home Page|Press In... Home | Endowment... N. Virginia vchinni

AWS Services Search [Alt+S]

The LoadBalancer service will provide an external IP address to access your application.



Problems Encountered and Solutions

a. AWS CodeDeploy Issues

- Issue: File already exists on the deployment target.
- Solution: Manually removed the existing file or adjusted deployment settings to overwrite or retain existing files.

b. EKS Cluster Deletion

- Issue: EKS cluster was deleted, but resources remained in CloudFormation.
- Solution: Deleted remaining CloudFormation stacks and associated IAM roles.

c. Docker Image Issues

- Issue: Docker image not available in Docker Hub.
- Solution: Built and pushed the Docker image to Docker Hub.

d. Kubernetes Cluster Issues

- Issue: Problems accessing or managing the EKS cluster.
- Solution: Verified AWS CLI and kubectl configurations.

Conclusion: This project demonstrates a comprehensive approach to deploying a simple web application using AWS services, Docker, and Kubernetes. Starting with setting up EC2 instances and utilizing AWS CodeDeploy for automated deployments, the project extends to containerizing the application with Docker and managing it with Kubernetes. This setup enhances the application's scalability, maintainability, and reliability. Through this journey, we addressed deployment challenges, effectively managed resources, and ensured smooth application delivery, providing a robust framework for deploying web applications in a cloud-native environment.