



Creating Deployment groups for Auto Scaling and Load Balancing

The task is about deploying an application using AWS services for high availability and reliability. Initially, you create an EC2 Auto Scaling group and an Application Load Balancer (ALB) to handle traffic and ensure your application remains available even during high demand. You start by creating an Amazon Machine Image (AMI) from your current EC2 instance, then use it to set up an Auto Scaling group through a launch template. Afterward, you configure the ALB to distribute traffic across instances.

Next, you update your CodeDeploy setup to work with the new Auto Scaling group and ALB, ensuring deployments can be managed efficiently. Finally, you update your deployment pipeline to reflect these changes. The end goal is to automate and scale your application deployment, enhancing its availability and reliability.



To begin with the Lab:

1. In the previous labs, we used AWS CodeDeploy to deploy to a single EC2 instance. But in real life, you primarily use EC2 auto-scaling groups and elastic load balancers to achieve high availability and reliability on AWS. So, in this lecture, you will learn to create deployment groups for them and update your pipelines accordingly.
2. To begin with, we will create an application load balancer and an EC2 auto-scaling group for our sample application. Now we need an EC2 launch template to create an auto-scaling group.
3. Therefore, we will first create an Amazon machine image, AMI, from our current EC2 instance to use in our launch template. But CodeDeploy raises errors if it finds files in the deployment folder that weren't deployed by the same deployment group.
4. So, let's reconnect to our EC2 instance to clean up. If you remember, we deploy to the '/var/www/my-angular-project' folder. As you see, it contains files from our previous deployments. They were also deployed by CodeDeploy but by our existing deployment group. Then, let's use the Linux 'rm' command with 'sudo' and the '-f' option to forcefully remove files and then append a slash followed by an asterisk to the folder path to select all files for deletion.

```
ls /var/www/my-angular-project  
sudo rm -f /var/www/my-angular-project
```

```
[ec2-user@ip-172-31-24-187 ~]$ ls /var/www/my-angular-project  
3rdpartylicenses.txt favicon.ico index.html main.d4af6a00da617af8.js polyfills.f35e31f90dcc29a9.js runtime.21abf645c946126d.js styles.4af885ee8aa244ff.css  
[ec2-user@ip-172-31-24-187 ~]$ sudo rm -f /var/www/my-angular-project/*  
[ec2-user@ip-172-31-24-187 ~]$ ls /var/www/my-angular-project  
[ec2-user@ip-172-31-24-187 ~]$
```



Step 1: Creating AMI and Launch template

5. Now go back to EC2 and stop your instance. Below you can see that your instance has been stopped.

Instances (1) Info									
C Connect Instance state ▾ Actions ▾ Launch instances ▾									
<input type="text"/> Find Instance by attribute or tag (case-sensitive)		All states ▾							
<input type="checkbox"/>	Name ↗	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4
<input type="checkbox"/>	i-0322de312a501d0a6	Stopping	View details	t2.micro	-	eu-west-1a	ec2-18-201-191-79.eu...	18.201.191...	18.201.191...

6. Now you need to select your instance and click on Actions then you need to choose Image and template and click on Create Image.

The screenshot shows the AWS EC2 Instances page with one instance selected. The 'Actions' dropdown menu is open, and the 'Image and templates' option is highlighted with a blue border. Other options like 'Create image' and 'Create template from instance' are also visible in the dropdown.

7. Now here just give your image a name and keep everything to default then create your image.

Create image [Info](#)

An image (also referred to as an AMI) defines the programs and settings that are applied when you launch an EC2 instance. You can create an image from the configuration of an existing instance.

Instance ID	<input type="text"/> i-0322de312a501d0a6
Image name	<input type="text"/> web-server-ami
Maximum 127 characters. Can't be modified after creation.	
Image description - optional	<input type="text"/> Image description
Maximum 255 characters	

8. Now come to AMI and check whether it is available or not. When it is available you need to go to the Lauch template section in EC2 and click to create a Lauch template for auto-scaling.

Amazon Machine Images (AMIs) (1) Info									
C Recycle Bin EC2 Image Builder Actions ▾ Launch instance from AMI									
<input type="text"/> Find AMI by attribute or tag		Owned by me ▾							
AMI name	AMI ID	Source	Owner	Visibility	Status	Creation			
web-server-ami	ami-039524f90c5bbf2aa	463646775279/web-server-ami	463646775279	Private	Available	2024/08/			

The screenshot shows the 'Compute' section of the AWS console, specifically the 'EC2 launch templates' page. The page title is 'EC2 launch templates' and the subtitle is 'Streamline, simplify and standardize instance launches'. Below the title, there is a brief description of what launch templates do. At the bottom right, there is a large orange button labeled 'Create launch template'.

9. First you need to give a name to your launch template then enable auto-scaling guidance.

Launch template name and description

Launch template name - *required*

web-server-template

Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '*', '@'.

Template version description

a production webserver template

Max 255 chars

Auto Scaling guidance | [Info](#)

Select this if you intend to use this template with EC2 Auto Scaling

Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

▶ [Template tags](#)
▶ [Source template](#)

10. Then in the application and OS images section choose My AMI and then choose your web server AMI that you just created.

Recents | [My AMIs](#) | [Quick Start](#)

Owned by me Shared with me

[Browse more AMIs](#)
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

web-server-ami
ami-039524f90c5bbf2aa
2024-08-09T08:45:36.000Z Virtualization: hvm ENA enabled: true Root device type: ebs

Description

-

Architecture	AMI ID
x86_64	ami-039524f90c5bbf2aa

11. After that in the instance type choose t2.micro and in the key pair it is up to you to include a key pair or not.

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

[Advanced](#)

Instance type

t2.micro	Free tier eligible		
Family: t2	1 vCPU	1 GiB Memory	Current generation: true
On-Demand RHEL base pricing:	0.027 USD per Hour		
On-Demand Linux base pricing:	0.0126 USD per Hour		
On-Demand SUSE base pricing:	0.0126 USD per Hour		
On-Demand Windows base pricing:	0.0172 USD per Hour		

[All generations](#)

[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

▼ Key pair (login) [Info](#)

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

Key pair name

Don't include in launch template [▼](#) [Create new key pair](#)

12. Then in the Network settings you just need to choose your existing security group.

▼ Network settings [Info](#)

Subnet [Info](#)

Don't include in launch template [▼](#) [Create new subnet](#)

When you specify a subnet, a network interface is automatically added to your template.

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Select existing security group [▼](#) Create security group

Security groups [Info](#)

Select security groups [▼](#)

web-server-SG sg-0624722342afb7606 [X](#)

VPC: vpc-0c86506eb35652747 [Compare security group rules](#)

► Advanced network configuration

13. In the end expand the advanced details option and choose the IAM role that you had created in the previous lab to attach it to your instance.

14. After that just launch your template.

The screenshot shows the 'Advanced details' section of the IAM instance profile configuration. It includes fields for selecting an IAM role ('WebServerRole') and specifying the 'Hostname type' ('Don't include in launch template'). A 'Create new IAM profile' button is also visible.

15. Below you can see your launch template.

The screenshot displays the 'Launch template details' page for 'web-server-template'. It shows basic information like the launch template ID (lt-02ffc00780dd15563), name (web-server-template), default version (1), and owner (arn:aws:iam::463646775279:user/cfpulkit). The 'Details' tab is selected, showing the 'Launch template version details' section. This section includes a dropdown for the version (1 (Default)), a description ('a production webserver template'), creation date (2024-08-09T09:09:11.000Z), and created by (arn:aws:iam::463646775279:user/cfpulkit). Other tabs for 'Versions', 'Template tags', 'Instance details', 'Storage', 'Resource tags', 'Network interfaces', and 'Advanced details' are also present.

👉 Step 2: Creating Application Load Balancer

1. Now go to Load Balancer in EC2 and click on Create Load Balancer. Then choose to create an **Application Load Balancer**.
2. Now give the load balancer a name and keep other things to default and scroll down.

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.

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.

3. Then in the network mappings you need to choose all the Availability Zones so that the traffic can be balanced easily.

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

eu-west-1a (euw1-az2)

Subnet

IPv4 subnet CIDR: 172.31.16.0/20

IPv4 address

Assigned by AWS

eu-west-1b (euw1-az3)

Subnet

IPv4 subnet CIDR: 172.31.32.0/20

IPv4 address

Assigned by AWS

eu-west-1c (euw1-az1)

Subnet

IPv4 subnet CIDR: 172.31.0.0/20

IPv4 address

Assigned by AWS

4. Now for the security group click on create a new security group.

Security groups | Info

A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can

[create a new security group](#)

Security groups



default

sg-09fa758bfe5efbf33 VPC: vpc-0c86506eb35652747

- To create a new security group, you just need to give it a name and description then in the inbound rules you need to open Port 80 HTTP from everywhere and create your security group.

The screenshot shows the 'Basic details' section with a security group name 'albSecuritygroup' and a VPC 'vpc-0c86506eb35652747'. In the 'Inbound rules' section, a rule is being added for 'HTTP' on port '80' from 'Anywhere' to '0.0.0.0/0'. An 'Add rule' button is visible.

- Now come back to ALB and choose your security group, also remove the default SG.

The screenshot shows the 'Security groups' selection interface. A dropdown menu lists 'albSecuritygroup' with its ID 'sg-0221c20dc1bfb127' and VPC 'vpc-0c86506eb35652747'. A 'Remove' button is shown next to the selected group.

- Then we also need a target group for our ALB. So, click on create target group.

The screenshot shows the 'Listeners and routing' configuration for an ALB. It lists a listener for 'HTTP:80' with 'Protocol' set to 'HTTP' and 'Port' set to '80'. The 'Default action' is set to 'Forward to' with a dropdown menu showing 'Select a target group'. A 'Create target group' link is also present. Below the listener, there are sections for 'Listener tags - optional' and an 'Add listener' button.

- To create a target group first in the target type, choose Instances then scroll down, and you need to give a name to your target group. Leave the rest of the settings to default and move to the next page.

9. After that you don't need to choose any target now so, just click on create target group.

Basic configuration

Settings in this section can't be changed after the target group is created.

Choose a target type

Instances

- Supports load balancing to instances within a specific VPC.
- Facilitates the use of [Amazon EC2 Auto Scaling](#) to manage and scale your EC2 capacity.

Target group name

alb-target-group

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

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

10. Now come back to ALB and choose your target group. After that go ahead and create your load balancer.

Listeners and routing [Info](#)

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

▼ Listener HTTP:80

Remove

Protocol	Port	Default action	Info
HTTP	: 80 1-65535	Forward to	alb-target-group Target type: Instance, IPv4 Create target group

Listener tags - optional

Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

Add listener tag

You can add up to 50 more tags.

Add listener

11. Below you can see that our load balancer is in provisioned state.

EC2 > Load balancers > MyALB			
MyALB			 Actions ▾
▼ Details			
Load balancer type	Status	VPC	Load balancer IP address type
Application	⌚ Provisioning	vpc-0c86506eb35652747	IPv4
Scheme	Hosted zone	Availability Zones	Date created
Internet-facing	Z32012XQLNTSW2	subnet-0a8028e0138997c60 eu-west-1a (euw1-az2)	August 9, 2024, 14:55 (UTC+05:30)
		subnet-02a285152ac76631a eu-west-1b (euw1-az3)	
		subnet-06aa1c311e5a2981f eu-west-1c (euw1-az1)	
Load balancer ARN		DNS name Info	
		arn:aws:elasticloadbalancing:eu-west-1:463646775279:loadbalancer/app/MyALB/4c54f40b3a6704c4	MyALB-900842786.eu-west-1.elb.amazonaws.com (A Record)

😊 Step 3: Creating Auto Scaling Group

- Now you need to go to the Auto Scaling group and click to create one. Now on step 1 you need to give a name to your auto scaling group. Then you need to choose your template and move forward.

Choose launch template [Info](#)

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group.

Name
Auto Scaling group name Enter a name to identify the group. <input type="text" value="my-autoscaling-group"/> Must be unique to this account in the current Region and no more than 255 characters.

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

web-server-template ▼ C

[Create a launch template](#) ↗

Version

Default (1) ▼ C

[Create a launch template version](#) ↗

Description a production webserver template	Launch template web-server-template ↗ lt-02ffc00780dd15563	Instance type t2.micro
AMI ID ami-039524f90c5bbf2aa	Security groups -	Request Spot Instances No
Key pair name -	Security group IDs sg-0624722342afb7606 ↗	

Additional details

Storage (volumes) -	Date created Fri Aug 09 2024 14:39:11 GMT+0530 (India Standard Time)
------------------------	--

Cancel Next

- now you need to choose all the Availability zones and subnet from your default subnet and click on Next.

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-0c86506eb3565274 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	
---------------------------------------	--

eu-west-1a subnet-0a8028e0138997c60	
---------------------------------------	--

172.31.16.0/20 Default	
------------------------	--

eu-west-1b subnet-02a285152ac76631a	
---------------------------------------	--

172.31.32.0/20 Default	
------------------------	--

eu-west-1c subnet-06aa1c311e5a2981f	
---------------------------------------	--

172.31.0.0/20 Default	
-----------------------	--

[Create a subnet](#)

[Cancel](#)

[Skip to review](#)

[Previous](#)

[Next](#)

3. Then you need to choose to attach to an existing load balancer, then choose your existing Target groups.

Load balancing Info

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

No load balancer

Traffic to your Auto Scaling group will not be fronted by a load balancer.

Attach to an existing load balancer

Choose from your existing load balancers.

Attach to a new load balancer

Quickly create a basic load balancer to attach to your Auto Scaling group.

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



alb-target-group | HTTP



Application Load Balancer: MyALB

- After that scroll down to health checks and turn on elastic load balancing health checks and change the health check grace period to 1800 seconds.

Health checks

Health checks increase availability by replacing unhealthy instances. When you use multiple health checks, all are evaluated, and if at least one fails, instance replacement occurs.

EC2 health checks

 Always enabled

Additional health check types - *optional* | [Info](#)

Turn on Elastic Load Balancing health checks Recommended

Elastic Load Balancing monitors whether instances are available to handle requests. When it reports an unhealthy instance, EC2 Auto Scaling can replace it on its next periodic check.

 EC2 Auto Scaling will start to detect and act on health checks performed by Elastic Load Balancing. To avoid unexpected terminations, first verify the settings of these health checks in the [Load Balancer console](#) 

Turn on VPC Lattice health checks

VPC Lattice can monitor whether instances are available to handle requests. If it considers a target as failed a health check, EC2 Auto Scaling replaces it after its next periodic check.

Health check grace period | [Info](#)

This time period delays the first health check until your instances finish initializing. It doesn't prevent an instance from terminating when placed into a non-running state.

1800  seconds

- Now in the group size choose the capacity to 2 in all aspects. Then move to the next page which will be the review page. Now click to create your auto-scaling group.

Group size Info

Set the initial size of the Auto Scaling group. After creating the group, you can change its size to meet demand, either manually or by using automatic scaling.

Desired capacity type
Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

Units (number of instances)

Desired capacity
Specify your group size.

2

Scaling Info

You can resize your Auto Scaling group manually or automatically to meet changes in demand.

Scaling limits
Set limits on how much your desired capacity can be increased or decreased.

Min desired capacity	Max desired capacity
2	2
Equal or less than desired capacity	Equal or greater than desired capacity

6. Below you can see that our auto-scaling group is updating.

The screenshot shows the AWS EC2 Auto Scaling Groups page. It displays a table with one row for the group 'my-autoscaling-group'. The group is associated with the launch template 'web-server-template' and is currently at version Default. The 'Desired capacity' is set to 2, and the 'Min' and 'Max' values are also 2. The 'Availability Zones' listed are eu-west-1b, eu-west-1c, and eu-west-1a. The status of the group is 'Updating capacity...', indicating that the scaling action is in progress.

😊 Step 5: Update Deployment Groups

1. Now go to CodeDeploy to update the deployment group. So, first go to applications then inside your application and from here you need to create a deployment group.

The screenshot shows the AWS CodeDeploy Applications page. On the left, there is a navigation sidebar with sections for Source, Artifacts, Build, Deploy, Pipeline, and Settings. The 'Applications' section is expanded, and the 'Application' sub-section is selected, indicated by a red box around it. The main content area shows the details for the application 'MyAngularApp'. Under the 'Deployment groups' tab, there is a table listing a single group named 'TaggedEC2instances'. This group has a status of 'Succeeded', a last attempted deployment on Aug 8, 2024 at 4:59 PM (UTC+5:30), and a last successful deployment on Aug 8, 2024 at 4:59 PM (UTC+5:30). A red box highlights the 'Create deployment group' button in the top right corner of the deployment groups table.

2. Now you need to give this deployment group a name then choose your IAM role which you created for your previous deployment group.

Create deployment group

Application

Application
MyAngularApp
Compute type
EC2/On-premises

Deployment group name

Enter a deployment group name

100 character limit

Service role

Enter a service role

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

3. Then keep deployment type as In-place.

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.

4. After that you have to choose Amazon EC2 Auto Scaling groups.

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

2 unique matched instances. [Click here for details](#) 

You can select up to 10 Amazon EC2 Auto Scaling groups to deploy your application revision to.

my-autoscaling-group 

▶ **Termination hook**

Amazon EC2 instances

On-premises instances

Matching instances

2 unique matched instances. [Click here for details](#) 

5. Now in the end you need to keep deployment settings to default and click to enable load balancing then to choose the load balancer type to ALB and choose your target group. After that create your deployment group.

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

Choose target groups

alb-target-group X

Classic Load Balancer

▶ Advanced - optional
Cancel
Create deployment group

6. Below you can see that our deployment group has been created.

Developer Tools > CodeDeploy > Applications > MyAngularApp

MyAngularApp ⚠ Notify ▾ Delete application

Application details	
Name MyAngularApp	Compute platform EC2/On-premises

Deployments Deployment groups Revisions

Deployment groups				
View details Edit Create deployment group				
Q < 1 > @				
Name	Status	Last attempted deployment	Last successful deployment	Trigger count
<input type="radio"/> MyAutoScalingGroup	-	-	-	0
<input type="radio"/> TaggedEC2Instances	Succeeded	Aug 8, 2024 4:59 PM (UTC+5:30)	Aug 8, 2024 4:59 PM (UTC+5:30)	0

😊 Step 6: Update the Pipeline

1. Now navigate to the code pipeline and update the deployment stage of our pipeline. Here you need to click on edit.

Developer Tools > CodePipeline > Pipelines > AngularProject01

AngularProject01

Pipeline type: V2 Execution mode: QUEUED

Actions:

- Notify
- Edit** (highlighted with a red box)
- Stop execution
- Clone pipeline
- Release change

- Then scroll down to the bottom choose your deploy stage and click on edit. Now we need to edit our deploy stage for that click on the highlighted icon.

Edit: Deploy

Actions:

- Add entry condition
- Add success condition
- Add failure condition
- + Add action group

DeploytoEC2 (AWS CodeDeploy)

+ Add action

+ Add stage

Configure automatic rollback on stage failure

- Here you just need to change the deployment group as you can see below. Then save your change.

Edit action

Action name
Choose a name for your action
DeploytoEC2
No more than 100 characters

Action provider
AWS CodeDeploy

Region
Europe (Ireland)

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

BuildArtifact
No more than 100 characters

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

Deployment group
Choose a deployment group that you have already created in the AWS CodeDeploy console. Or create a deployment group in the AWS CodeDeploy console and then return to this task.
MyAutoScalingGroup

Variable namespace - optional
Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. [Learn more](#)

- Now we are going to release our changes manually. For that, you need to click on Release change.

Developer Tools > CodePipeline > Pipelines > AngularProject01

AngularProject01

Pipeline type: V2 Execution mode: QUEUED

Actions:

- Notify
- Edit
- Stop execution
- Clone pipeline
- Release change** (highlighted with a red box)

- Now you need to wait until your pipeline gets executed. Below you can see that your pipeline has been executed.

The screenshot shows the AWS CodePipeline console with three stages:

- Source Stage:** Status: Succeeded. Pipeline execution ID: [9a93ddcb-26fa-4ddd-8ca0-8e2711214a24](#). Contains a step for AWS CodeCommit with status Succeeded - 15 minutes ago. A "View details" button is present. To the right, there are three green checkmarks.
- Build Stage:** Status: Succeeded. Pipeline execution ID: [9a93ddcb-26fa-4ddd-8ca0-8e2711214a24](#). Contains two steps: UnitTEst (AWS CodeBuild) and Build (AWS CodeBuild), both with status Succeeded. Each has a "View details" button. To the right, there are three green checkmarks.
- Deploy Stage:** Status: Succeeded. Pipeline execution ID: [9a93ddcb-26fa-4ddd-8ca0-8e2711214a24](#). Contains one step, DeploytoEC2 (AWS CodeDeploy), with status Succeeded - 6 minutes ago. A "View details" button is present. To the right, there is one green checkmark.

- Now if you want to see details regarding deployment then you can do that yourself and go to the Application load balancer and copy the DNS name.
- Then paste that DNS name in a new tab, here you can see that your application is online.

