



# Blue Green Deployment

Blue-green deployment is a release management strategy designed to minimize downtime and reduce risks associated with deploying new versions of software. The technique involves maintaining two identical production environments, referred to as "blue" and "green." Only one environment serves live production traffic at any given time.

## How Blue-Green Deployment Works:

### 1. Two Environments:

- **Blue Environment:** The current live production environment, serving all user traffic.
- **Green Environment:** An identical environment that is not currently serving user traffic but is ready for deployment.

### 2. Deployment to the Green Environment:

- The new version of the software is deployed to the green environment.
- Extensive testing is performed on the green environment to ensure the new release is stable and functions as expected. This includes integration, user acceptance, and performance testing.

### 3. Traffic Switch:

- Once testing is successful, traffic is redirected from the blue environment to the green environment.
- This switch can be instantaneous or gradual (a canary release can be used to slowly direct a percentage of traffic to the green environment before fully switching over).

### 4. Monitoring:

- After the switch, the green environment is monitored closely for any issues or bugs.
- If problems arise, traffic can be quickly redirected back to the blue environment, which still holds the previous stable version of the software.

### 5. Clean Up:

- If the new version in the green environment is confirmed to be stable and performing well, the blue environment can be updated with the next release or decommissioned.
- This alternating pattern continues for future deployments.

## Advantages:

- **Zero Downtime:** Because the switch between environments is seamless, downtime is either nonexistent or minimal, providing continuous availability.

- **Easy Rollback:** If a problem is detected with the new version, reverting to the old version is as simple as redirecting traffic back to the blue environment.
- **Thorough Testing:** The new version is tested in a production-like environment (green) before it goes live, reducing the chances of unforeseen issues.

### **Challenges:**

- **Resource Intensive:** Maintaining two fully operational environments can be costly in terms of infrastructure, especially for large-scale applications.
- **Operational Complexity:** Ensuring both environments are identical and keeping them synchronized can be complex, especially when dealing with databases and other stateful services.

### **Use Cases:**

- **High-Availability Systems:** Systems that require near-zero downtime, such as online services, e-commerce platforms, and financial services.
- **Frequent Updates:** Environments where new features or updates are released frequently and need to be quickly and safely deployed.

Blue-green deployment is often used in conjunction with other DevOps practices, such as continuous integration and continuous deployment (CI/CD), to streamline the release process and ensure smooth, reliable deployments.

**The process described involves updating a web application using AWS CodeDeploy with a blue-green deployment strategy. The goal is to release a new version of the application with minimal downtime and risk.**

**First, the deployment settings are configured for blue-green deployment, ensuring that the new version (green) is deployed while the current version (blue) remains live. After making code changes in the application, the new version is pushed to the repository, triggering the deployment pipeline. The deployment process then creates a new set of instances (green fleet) identical to the existing ones (blue fleet) and deploys the new version to them.**

**Once the green fleet is up and running, traffic is gradually shifted from the blue fleet to the green fleet, ensuring a smooth transition. Finally, the old blue fleet is decommissioned, leaving only the new green fleet serving the updated application.**

**End Goal:** Successfully update the web application to a new version with zero downtime, ensuring that users experience a seamless transition without any interruptions.

### **To begin with the Lab:**

1. Now in your AWS Console go to CodeDeploy Applications and open the deployment group. So, when you open your Application, you can see your Deployment groups open the auto-scaling deployment group.

Developer Tools > CodeDeploy > Applications > MyAngularApp

## MyAngularApp

[Notify](#) [Delete application](#)

### Application details

Name MyAngularApp	Compute platform EC2/On-premises
----------------------	-------------------------------------

Deployments Deployment groups Revisions

### Deployment groups

Deployment groups				
<a href="#">View details</a> <a href="#">Edit</a> <a href="#">Create deployment group</a>				
<input type="text"/> Q				
Name	Status	Last attempted deployment	Last successful deployment	Trigger count
<input type="radio"/> MyAutoScalingGroup	Succeeded	Aug 12, 2024 10:13 AM (UTC+5...)	Aug 12, 2024 10:13 AM (UTC+5...)	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

- Now you need to click on Edit, then scroll down to deployment type and choose Blue/green deployment.

Developer Tools > CodeDeploy > Applications > MyAngularApp > MyAutoScalingGroup

## MyAutoScalingGroup

[Edit](#) [Delete](#) [Create deployment](#)

### Deployment group details

Deployment group name MyAutoScalingGroup	Application name MyAngularApp	Compute platform EC2/On-premises
Deployment type In-place	Service role ARN arn:aws:iam::463646775279:role/CodeDeployEC2ServiceRole	Deployment configuration CodeDeployDefault.OneAtATime
Rollback enabled True	Agent update scheduler <a href="#">Learn to schedule update in AWS Systems Manager</a>	

### 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.

- Now scroll down to deployment settings and choose deployment configuration as all at once.
- After that save your changes.

## Deployment settings

Traffic rerouting

- Reroute traffic immediately
- I will choose whether to reroute traffic

---

Choose whether instances in the original environment are terminated after the deployment succeeds, and how long to wait before termination.

- Terminate the original instances in the deployment group
- Keep the original instances in the deployment group running

Days	Hours	Minutes
0	0	0

---

**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](#)

5. Below you can see that our changes have been saved.

Success
  
Deployment group updated

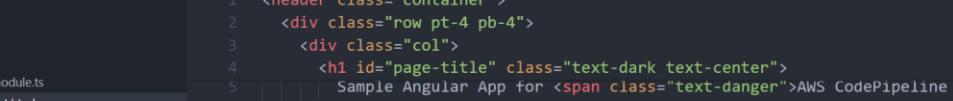
Developer Tools > CodeDeploy > Applications > MyAngularApp > MyAutoScalingGroup

### MyAutoScalingGroup

Edit
Delete
Create deployment

Deployment group details		
Deployment group name MyAutoScalingGroup	Application name MyAngularApp	Compute platform EC2/On-premises
Deployment type Blue/green	Service role ARN arn:aws:iam::463646775279:role/CodeDeployEC2ServiceRole	Deployment configuration CodeDeployDefault.AllAtOnce
Rollback enabled True	Agent update scheduler <a href="#">Learn to schedule update in AWS Systems Manager</a>	

6. Next, we'll release a new version of our web application and trigger a deployment to see how it works.
7. So, open your Angular application in VS Code. Now open the app.component.html file and change the version.



The screenshot shows the file structure of a project named "MY-ANGULAR-PROJECT". The "src" folder contains files like "deploy-scripts", "e2e", "app-routing.module.ts", "app.component.html", "app.component.scss", "app.component.specs.ts", "app.module.ts", "assets", "environments", "favicon.ico", "index.html", "main.ts", "polyfills.ts", and "styles.css". The "app.component.html" file is open in the editor. It contains an HTML template with an "header" component containing a "h1" title, some "hr" tags, and a "h2" section with a "span.badge.bg-info" element. A red box highlights this "span" element. The code is syntax-highlighted with colors for tags, attributes, and classes.

```
<header class="container">
  <div class="row pt-4 pb-4">
    <div class="col">
      <h1 id="page-title" class="text-dark text-center">
        Sample Angular App for <span class="text-danger">AWS CodePipeline Step t
      </h1>
      <hr>
      <hr>
      <h2 class="text-end">
        | <span class="badge bg-info">Version: 7.0</span>
      </h2>
    </div>
  </div>
</header>
```

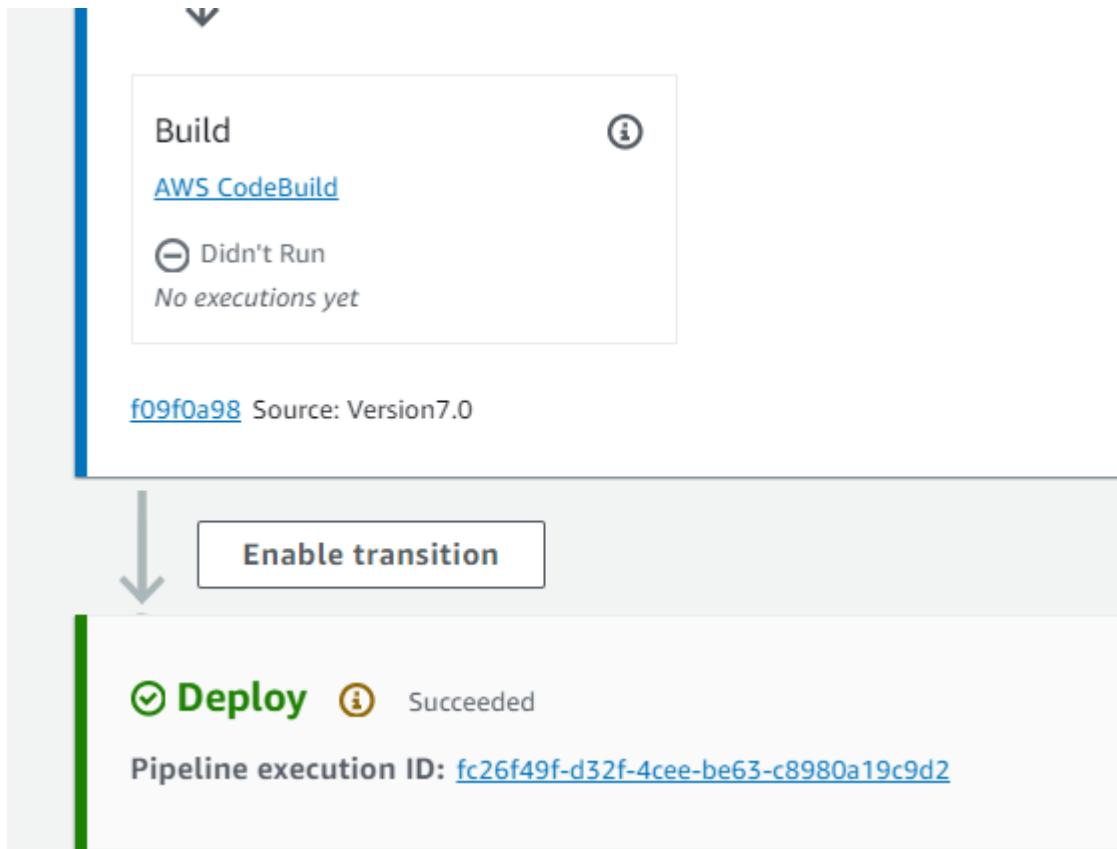
- Now save the changes and open the terminal, then you need to commit the changes and push them to your repository.

```
git commit -a -m "Version7.0"  
git push origin master
```

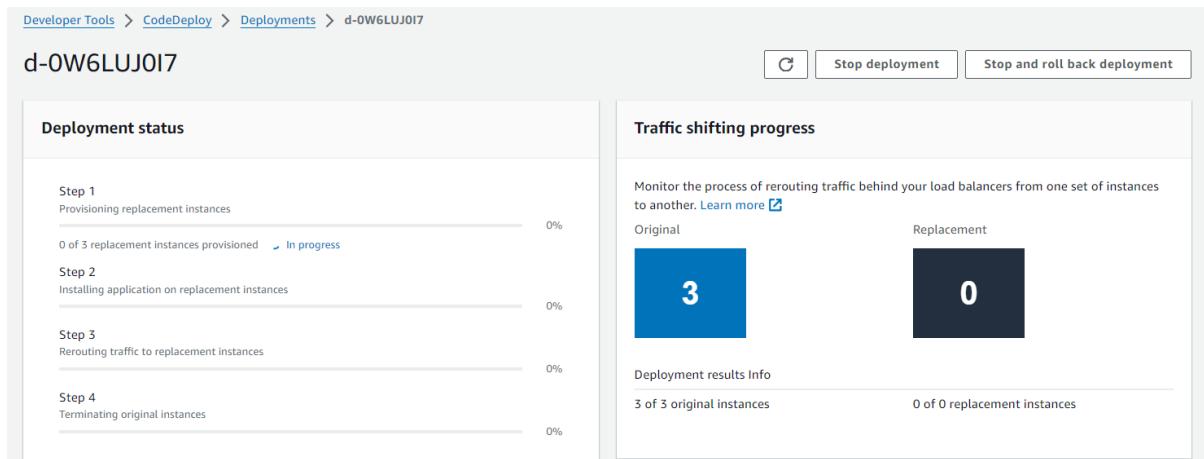
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE AZURE  psh +   ... ^ x

● PS C:\Users\PULKIT\Downloads\my-angular-project> git commit -a -m "Version7.0"
warning: in the working copy of 'src/app/app.component.html', LF will be replaced by CRLF the next time Git touches it
[master f09f0a9] Version7.0
 1 file changed, 1 insertion(+), 1 deletion(-)
● PS C:\Users\PULKIT\Downloads\my-angular-project> git push origin master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 432 bytes | 432.00 KiB/s, done.
Total 5 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.eu-west-1.amazonaws.com/v1/repos/DemoAngularRepo
 1bde6a4..f09f0a9  master -> master
● PS C:\Users\PULKIT\Downloads\my-angular-project> █
```

9. After that go to your Pipeline and as soon as it starts you need to disable the transition between your build stage and deploy. So, that you can enable it later and check the deployment for yourself like we did in our previous labs.
  10. So, when the build stage is executed, you need to enable the transition and wait until the view details option is available in the deployment stage. Then you need to click on it and go to CodeDeploy.



11. Now we have enabled the transition, and we are in CodeDeploy by clicking on view in the CodeDeploy option from our deploy stage.
12. As you see, this time the 'Deployment status' section is different. We see 4 steps of action in the deployment status. Now, we are in the first step. The deployment is provisioning new instances for replacement, or in other words, the green group. Let's scroll down. In the 'Deployment lifecycle events' section, as you see, we have only the original instances at start, the blue group. And they are all in 'Pending' state.
13. They will remain in this state until the new instances in the green fleet are registered to the load balancer.



**Revision details**

Revision location s3://codepipeline-eu-west-1-480538782531/AngularProject01/BuildArtif/4yYoo4s eTag=2f67f98683eb97787c6db3ee688b3ee3	Revision created Just now	Revision description Application revision registered by Deployment ID: d-0W6LUJ017
--	------------------------------	---

**Deployment lifecycle events**

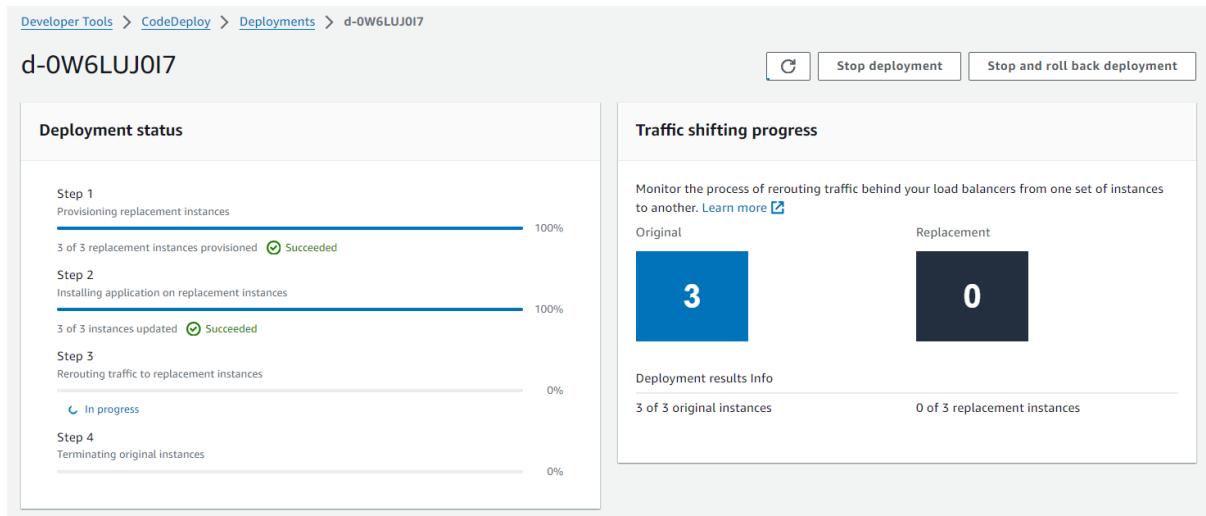
Instance ID	Environment	Traffic	Duration	Status	Most recent event	Events	Start time	End time
i-00a1efbf7a254cdaf	Original	No	-	Pending	-	<a href="#">View events</a>	Aug 12, 2024 11:03 AM (UTC+5:30)	-
i-026d58a2ea0ba6ba0	Original	No	-	Pending	-	<a href="#">View events</a>	Aug 12, 2024 11:03 AM (UTC+5:30)	-
i-052e8bab91526fd0c	Original	No	-	Pending	-	<a href="#">View events</a>	Aug 12, 2024 11:03 AM (UTC+5:30)	-

14. Now, let's switch to the EC2 auto-scaling groups to see what happens there. As you see, now we have a new auto-scaling group. Let's select it from the list. And then, let's open the 'Instances' tab. Yes, we see that it has 3 instances just like the previous one. So, this group will be the green fleet of our blue/green deployment. It was created automatically by CodeDeploy by duplicating our existing auto-scaling group. Besides, the name of the new auto-scaling group starts with CodeDeploy and contains both the deployment group name and the ID of the deployment that created it.

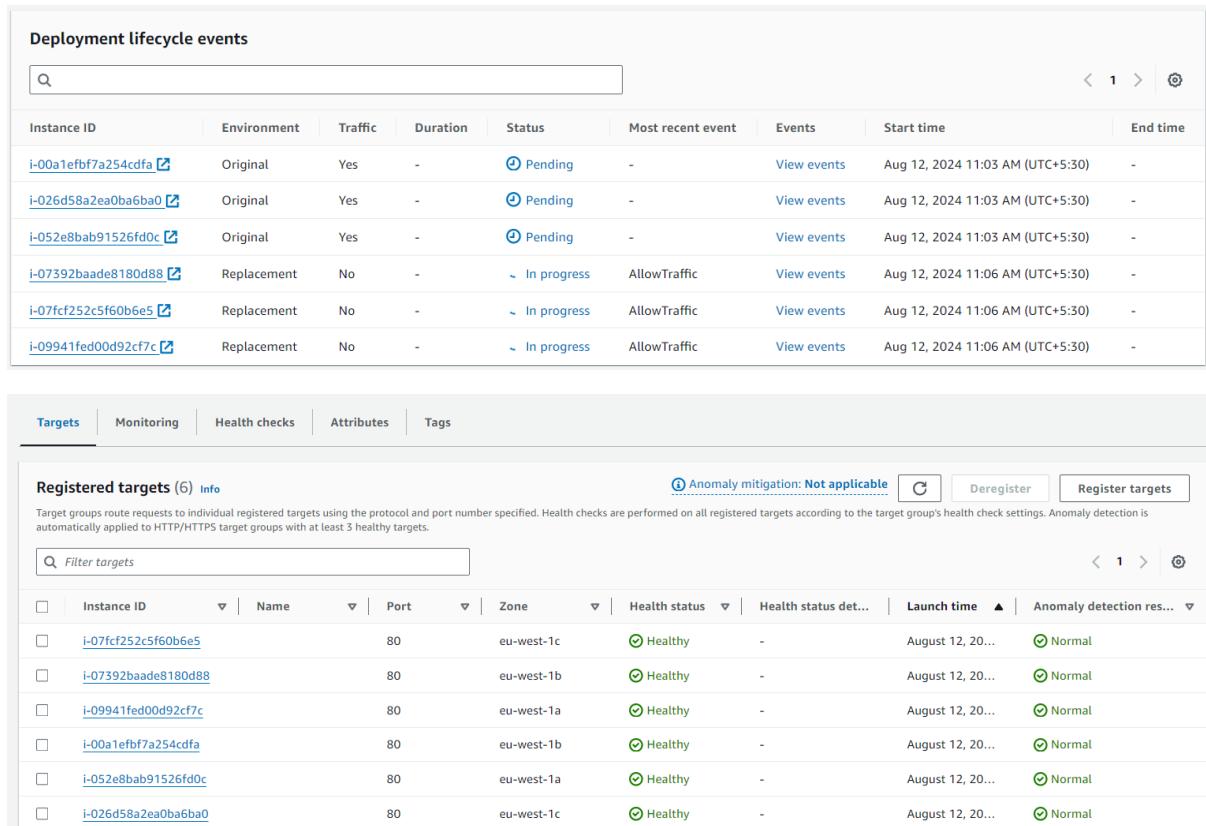
**EC2 > Auto Scaling groups**

Auto Scaling groups (2) <a href="#">Info</a>		<a href="#">C</a>	Launch configurations	Launch templates	Actions	<a href="#">Create Auto Scaling group</a>		
<a href="#">Search your Auto Scaling groups</a>		< 1 > <a href="#">@</a>						
<input type="checkbox"/>	Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability...
<input type="checkbox"/>	<a href="#">CodeDeploy_MyAutoScalingGroup_d-0W6LUJ017</a>	<a href="#">web-server-template</a>   Version Default	3	-	3	3	3	eu-west-1b, ...
<input type="checkbox"/>	<a href="#">my-autoscaling-group</a>	<a href="#">web-server-template</a>   Version Default	3	-	3	3	3	eu-west-1b, ...

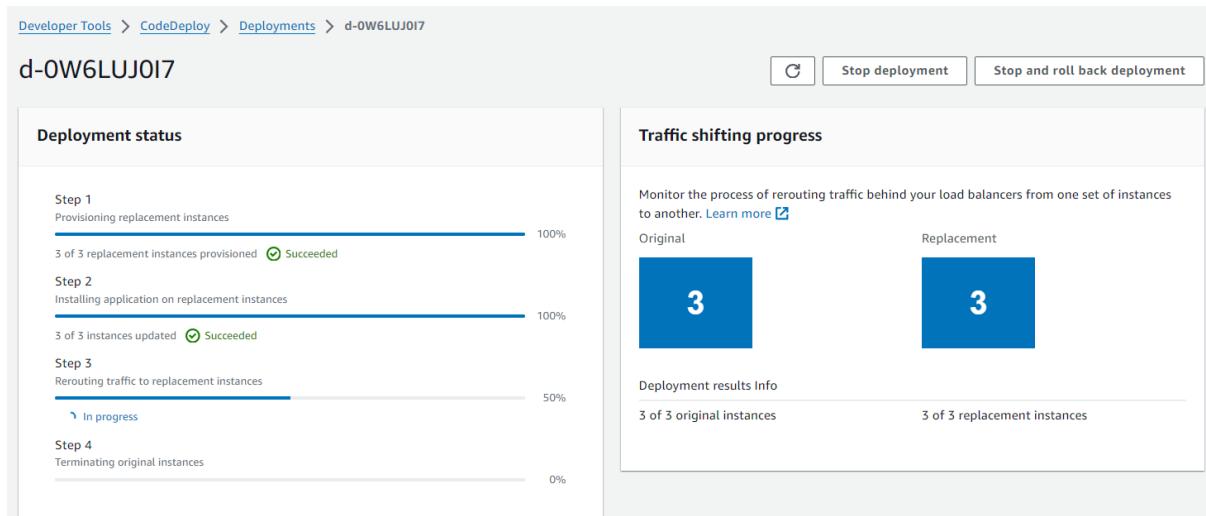
15. So, the first step succeeded. The replacement auto-scaling group and its instances were provisioned successfully. And now, we're in the second step. The new revision will be deployed to the new instances, the green fleet. In the 'Deployment lifecycle events' list, we continue to see only the older instances, the blue fleet.
16. In step 3, the traffic will be shifted from the blue fleet to the green fleet. To achieve this, firstly, the replacement instances, the green fleet will be registered to the load balancer and then the older instances, the blue fleet will be deregistered from it.



17. As you see, now we have 6 instances in the target group and the green fleet instances also seem registered successfully. Because, all the instances here are healthy. So, at this stage, until the deregistration of the blue fleet starts you will see 2 versions in production.



18. Now, step 3 has 50% progress. Besides, on the 'Traffic shifting progress' section the 'Replacement' box became blue as well. So, currently, both the original and the replacement instances are receiving traffic. Let's scroll down to see what is happening on the instances.



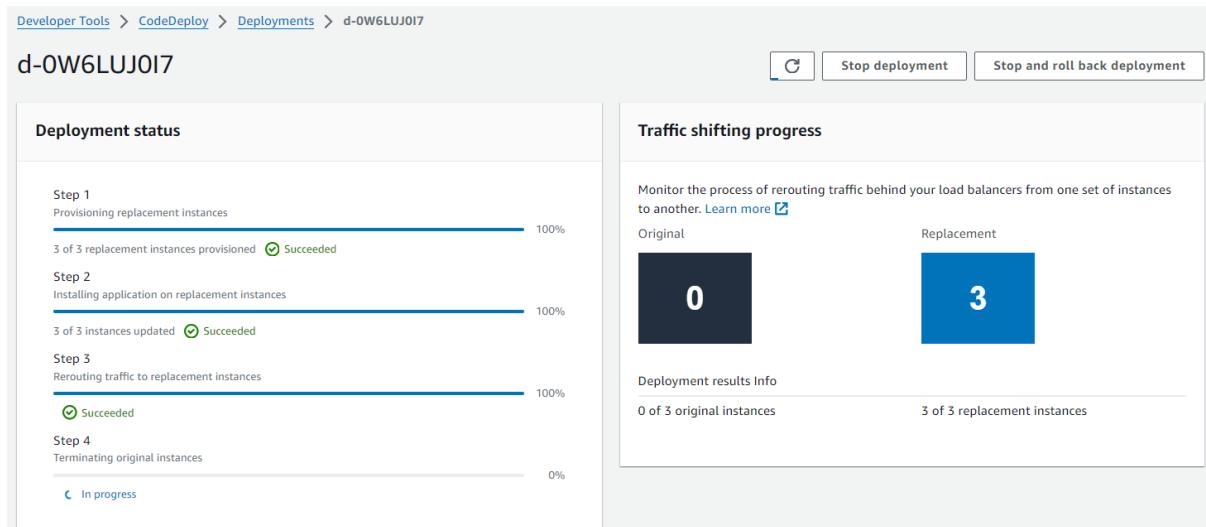
19. Now, as you see, the original instances are in progress. Their last events are `BeforeBlockTraffic`. So, they are being taken offline. The deployments on the replacement instances, our green fleet succeeded as well.

Deployment lifecycle events								
Instance ID	Environment	Traffic	Duration	Status	Most recent event	Events	Start time	End time
i-00a1efbf7a254cdfa	Original	Yes	-	Pending	-	<a href="#">View events</a>	Aug 12, 2024 11:03 AM (UTC+5:30)	-
i-026d58a2ea0ba6ba0	Original	Yes	-	Pending	-	<a href="#">View events</a>	Aug 12, 2024 11:03 AM (UTC+5:30)	-
i-052e8bab91526fd0c	Original	Yes	-	Pending	-	<a href="#">View events</a>	Aug 12, 2024 11:03 AM (UTC+5:30)	-
i-07392baade8180d88	Replacement	Yes	2 minutes 47 seconds	Succeeded	AfterAllowTraffic	<a href="#">View events</a>	Aug 12, 2024 11:06 AM (UTC+5:30)	Aug 12, 2024 11:06 AM (UTC+5:30)
i-07fcf252c5f60b6e5	Replacement	Yes	2 minutes 47 seconds	Succeeded	AfterAllowTraffic	<a href="#">View events</a>	Aug 12, 2024 11:06 AM (UTC+5:30)	Aug 12, 2024 11:06 AM (UTC+5:30)
i-09941fed00d92cf7c	Replacement	Yes	2 minutes 47 seconds	Succeeded	AfterAllowTraffic	<a href="#">View events</a>	Aug 12, 2024 11:06 AM (UTC+5:30)	Aug 12, 2024 11:06 AM (UTC+5:30)

20. Now, we have 3 instances in draining state. These are the original instances, our blue fleet, and the replacement instances in our green fleet are healthy. So, currently, our website is serving only the new version. The switch from blue to green almost completed.

Targets	Monitoring	Health checks	Attributes	Tags							
<b>Registered targets (6) <a href="#">Info</a></b>										Anomaly mitigation: Not applicable	
Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.											
<input type="checkbox"/>	Instance ID	Name	Port	Zone	Health status	Health status det...	Launch time	Anomaly detection res...			
<input type="checkbox"/>	i-07fcf252c5f60b6e5		80	eu-west-1c	<span>Healthy</span>	-	August 12, 20...	<span>Normal</span>	<a href="#">C</a>	<a href="#">Deregister</a>	<a href="#">Register targets</a>
<input type="checkbox"/>	i-07392baade8180d88		80	eu-west-1b	<span>Healthy</span>	-	August 12, 20...	<span>Normal</span>			
<input type="checkbox"/>	i-09941fed00d92cf7c		80	eu-west-1a	<span>Healthy</span>	-	August 12, 20...	<span>Normal</span>			
<input type="checkbox"/>	i-00a1efbf7a254cdfa		80	eu-west-1b	<span>Draining</span>	Target deregistrat...	August 12, 20...	<span>Normal</span>			
<input type="checkbox"/>	i-052e8bab91526fd0c		80	eu-west-1a	<span>Draining</span>	Target deregistrat...	August 12, 20...	<span>Normal</span>			
<input type="checkbox"/>	i-026d58a2ea0ba6ba0		80	eu-west-1c	<span>Draining</span>	Target deregistrat...	August 12, 20...	<span>Normal</span>			

21. Step 3 also finished and our last step started afterwards. Well, it also succeeded. So, our deployment finished. Let's scroll down to the lifecycle events. As you see, now all instances are in 'Succeeded' state.



Deployment lifecycle events								
Instance ID	Environment	Traffic	Duration	Status	Most recent event	Events	Start time	End time
i-00a1efbf7a254cdfa	Original	No	1 minute 11 seconds	<span>Succeeded</span>	AfterBlockTraffic	<a href="#">View events</a>	Aug 12, 2024 11:09 AM (UTC+5:30)	Aug 12, 2024 11:09 AM (UTC+5:30)
i-026d58a2ea0ba6ba0	Original	No	1 minute 11 seconds	<span>Succeeded</span>	AfterBlockTraffic	<a href="#">View events</a>	Aug 12, 2024 11:09 AM (UTC+5:30)	Aug 12, 2024 11:09 AM (UTC+5:30)
i-052e8bab91526fd0c	Original	No	1 minute 11 seconds	<span>Succeeded</span>	AfterBlockTraffic	<a href="#">View events</a>	Aug 12, 2024 11:09 AM (UTC+5:30)	Aug 12, 2024 11:09 AM (UTC+5:30)
i-07392baade8180d88	Replacement	Yes	2 minutes 47 seconds	<span>Succeeded</span>	AfterAllowTraffic	<a href="#">View events</a>	Aug 12, 2024 11:06 AM (UTC+5:30)	Aug 12, 2024 11:06 AM (UTC+5:30)
i-07fcf252c5f60b6e5	Replacement	Yes	2 minutes 47 seconds	<span>Succeeded</span>	AfterAllowTraffic	<a href="#">View events</a>	Aug 12, 2024 11:06 AM (UTC+5:30)	Aug 12, 2024 11:06 AM (UTC+5:30)
i-09941fed00d92cf7c	Replacement	Yes	2 minutes 47 seconds	<span>Succeeded</span>	AfterAllowTraffic	<a href="#">View events</a>	Aug 12, 2024 11:06 AM (UTC+5:30)	Aug 12, 2024 11:06 AM (UTC+5:30)

22. Now we see only the replacement instances here. And they are all healthy.

Targets	Monitoring	Health checks	Attributes	Tags
<b>Registered targets (3) <a href="#">Info</a></b>				
				<a href="#">Anomaly mitigation: Not applicable</a> <a href="#">C</a> <a href="#">Deregister</a> <a href="#">Register targets</a>
Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.				
<input type="text"/> Filter targets				
<input type="checkbox"/>	Instance ID	Name	Port	Zone
<input type="checkbox"/>	i-07fcf252c5f60b6e5	80	eu-west-1c	<span>Healthy</span>
<input type="checkbox"/>	i-07392baade8180d88	80	eu-west-1b	<span>Healthy</span>
<input type="checkbox"/>	i-09941fed00d92cf7c	80	eu-west-1a	<span>Healthy</span>
				Launch time ▲   Anomaly detection res... ▼
				August 12, 20...
				<span>Normal</span>
				August 12, 20...
				<span>Normal</span>
				August 12, 20...
				<span>Normal</span>

23. As you see, the original auto-scaling group we created has been deleted. The deployment just submitted its request to Amazon EC2 service and completed step 4. All instances in this group will be terminated eventually. So, now we have only one auto-scaling group remaining, the green group, which contains the name of our deployment group and the ID of our last deployment in its name.

<a href="#">EC2</a>	>	Auto Scaling groups			
<b>Auto Scaling groups (1) <a href="#">Info</a></b>					
<input type="checkbox"/>	Name	Launch template/configuration	Instances	Status	Desired capacity
<input type="checkbox"/>	CodeDeploy_MyAutoScalingGroup_d-0WG6LUJ0I7	web-server-template   Version Default	3	-	3
			3	3	eu-west-1b, ...

24. Below you can see that our website is up and running.

Congratulations! You successfully built and deployed your code.

This is a simple single-page calculator app developed with Angular and Bootstrap for the build examples on the [AWS CodePipeline Step by Step](#) course.

**Simple Calculator**

Your first input:  Please select an operator:  Your second input:

[Clear](#) [Calculate](#)