

The screenshot shows the AWS CloudShell interface with the following configuration:

- ECS Service Name:** customer-microservice
- Load balancers:**
 - Choose a load balancer:** microservicesLB
 - Production listener port:** HTTP: 80
 - Test listener port - optional:** HTTP: 8080
 - Target group 1 name:** customer-tg-two
 - Target group 2 name:** customer-tg-one

- In the **Deployment settings** section:

- **Traffic rerouting:** Choose **Reroute traffic immediately**.
- **Deployment configuration:** Choose **CodeDeployDefault.ECSAllAtOnce**.
- **Original revision termination:** Days: 0, Hours: 0, Minutes: 5

The screenshot shows the 'Deployment settings' configuration screen with the following options:

- Traffic rerouting:** Reroute traffic immediately
- Deployment configuration:** CodeDeployDefault.ECSAllAtOnce
- Original revision termination:** Days: 0, Hours: 0, Minutes: 5

At the bottom, there is a 'Create deployment group' button.

- Choose **Create deployment group**.

The screenshot shows three sequential AWS CodeDeploy application configuration pages:

- Deployment settings**: Set traffic rerouting to "Reroute traffic immediately". Deployment configuration is set to "CodeDeployDefault.ECSAllAtOnce". Original revision termination is set to 0 days, 0 hours, and 5 minutes.
- Success**: Confirmation message "Deployment group created". Deployment group details show name: "microservices-customer", application name: "microservices", compute platform: "Amazon ECS", deployment type: "Blue/green", service role ARN: "arn:aws:iam::082451908674:role/DeployRole", and rollback enabled: "False".
- microservices**: Application details show name: "microservices", compute platform: "Amazon ECS". Deployment groups tab is selected, showing a table of deployment groups. The table has columns: Name, Status, Last attempted deploy..., Last successful deploy..., and Trigger count. One entry is shown: "microservices-customer" with status "-".

3. Create a CodeDeploy deployment group for the *employee* microservice. Specify the same settings that you did in the prior step, except for the following:

The screenshot shows the AWS CodeDeploy console. On the left, a navigation sidebar lists various services under 'CodeDeploy'. The main area displays the 'microservices' application details, including its name and compute platform (Amazon ECS). Below this, the 'Deployment groups' tab is selected, showing a table with one entry: 'microservices-customer'.

Name	Status	Last attempted deploy...	Last successful deploy...	Trigger count
microservices-customer	-	-	-	0

- **Deployment group name:** Enter microservices-employee
- **ECS service name:** Choose employee-microservice.
- **Target group 1 name:** Choose employee-tg-two.
- **Target group 2 name:** Choose employee-tg-one.

The screenshot shows the 'Create deployment group' wizard. Step 1, 'Application', is selected. It shows the 'microservices' application has been chosen, with 'Compute type' set to 'Amazon ECS'.

Deployment group name
Enter a deployment group name
microservices-employed
100 character limit

Service role
Enter a service role

Deployment group name

Enter a deployment group name
microservices-employee

Service role

Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.
arn:aws:iam::082451908674:role/DeployRole

Environment configuration

Choose an ECS cluster name
microservices-serverlesscluster

Choose an ECS service name
employee-microservice

Load balancers

Choose an ECS service name
employee-microservice

Load balancers

Choose a load balancer
microservicesLB

Production listener port
HTTP: 80

Test listener port - optional
A test listener is required if you want to test your replacement version before traffic reroutes to it.
HTTP: 8080

Target group 1 name
employee-tg-two

Target group 2 name
employee-tg-one

Deployment settings

Traffic rerouting
Choose whether traffic reroutes to the replacement environment immediately or waits for you to start the rerouting process
 Reroute traffic immediately
 Specify when to reroute traffic

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.
 CodeDeploy/Default:ECSSAllAtOnce or [Create deployment configuration](#)

Original revision termination
Specify how long CodeDeploy waits before it terminates the original task set. After termination starts, you cannot rollback manually or automatically
 Days: 0 Hours: 0 Minutes: 5

► Advanced - optional

[Cancel](#) [Create deployment group](#)

Success
Deployment group created

Developer Tools > CodeDeploy > Applications > [microservices](#) > microservices-employee

microservices-employee

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

Deployment group details

Deployment group name	Application name	Compute platform
microservices-employee	microservices	Amazon ECS
Deployment type	Service role ARN	Deployment configuration
Blue/green	arn:aws:iam::082451908674:role/DeployRole	CodeDeploy/Default:ECSSAllAtOnce
Rollback enabled		
False		

Environment configuration

ECS cluster name	ECS service name
microservices-serverlesscluster	employee-microservice

Developer Tools > CodeDeploy > Applications > [microservices](#)

microservices

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

Application details

Name	Compute platform
microservices	Amazon ECS

[Deployments](#) [Deployment groups](#) [Revisions](#)

Deployment groups

Name	Status	Last attempted deploy...	Last successful deploy...	Trigger count
microservices-employee	-	-	-	0
microservices-customer	-	-	-	0

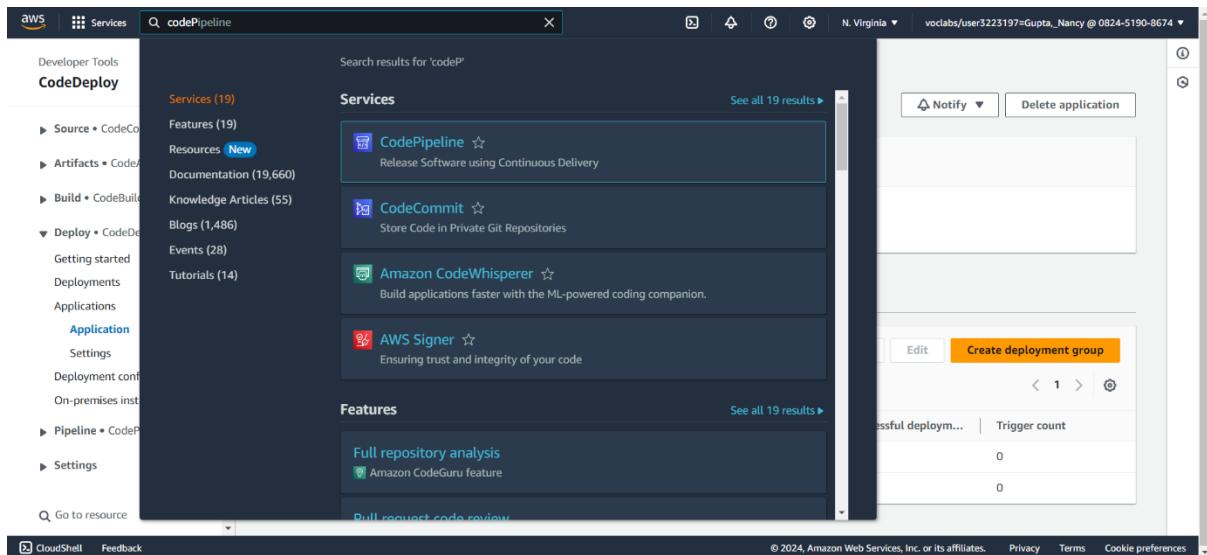
Task 8.2: Create a pipeline for the *customer* microservice

In this task, you will create a pipeline to update the *customer* microservice. When you first define the pipeline, you will configure CodeCommit as the source and CodeDeploy as the service that is responsible for deployment. You will then edit the pipeline to add the Amazon ECR service as a second source.

With an Amazon ECS blue/green deployment, which you will specify in this task, you provision a new set of containers, which CodeDeploy installs the latest version of your application on. CodeDeploy then reroutes load balancer traffic from an existing set of containers, which run the previous version of your application, to the new set of containers, which run the latest version. After traffic is rerouted to the new containers, the existing containers can be terminated. With a blue/green deployment, you can test the new application version before sending production traffic to it.

References

- The AWS Academy Cloud Architecting and AWS Academy Cloud Developing courses include hands-on labs that explore CodePipeline features.
- [AWS CodePipeline User Guide](#)



1. In the CodePipeline console, create a *customer* pipeline with the following settings:

The screenshot shows the AWS CodePipeline Pipelines page. The left sidebar has a tree view with nodes like Source, Artifacts, Build, Deploy, Pipeline, Getting started, Pipelines, Settings, Go to resource, and Feedback. The main area is titled 'Pipelines' and shows a table with columns: Name, Latest execution status, Latest source revisions, Latest execution started, and Most recent executions. A search bar is at the top. Below the table, it says 'No results' and 'There are no results to display.' The bottom right corner includes copyright information: © 2024, Amazon Web Services, Inc. or its affiliates.

- **Pipeline name:** Enter update-customer-microservice

The screenshot shows the 'Choose pipeline settings' step of a wizard. On the left, a sidebar lists steps: Step 1 (Choose pipeline settings), Step 2 (Add source stage), Step 3 (Add build stage), Step 4 (Add deploy stage), and Step 5 (Review). The main area is titled 'Pipeline settings'. It has a 'Pipeline name' field containing 'update-customer-microservice', a note about pipeline type, and a radio button for 'V2' which is selected. Below that is an 'Execution mode' section with options: Superseded (radio button), Queued (radio button selected), and Parallel. The 'Service role' field contains 'arn:aws:iam::082451908674:role/PipelineRole'. The bottom right corner includes copyright information: © 2024, Amazon Web Services, Inc. or its affiliates.

- **Service role:** Choose the ARN for PipelineRole.

The screenshot shows the 'Service role' configuration step. It has two options: 'New service role' (radio button) and 'Existing service role' (radio button selected). The 'Role ARN' field contains 'arn:aws:iam::082451908674:role/PipelineRole'. Below is a 'Variables' section with a note about pipeline level variables and a 'Add variable' button. A warning message says 'The first pipeline execution will fail if variables have no default values.' At the bottom is a 'Advanced settings' section. The bottom right corner includes copyright information: © 2024, Amazon Web Services, Inc. or its affiliates.

- **Source provider:** Choose **AWS CodeCommit**.

- **Repository name:** Choose **deployment**.

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1 Choose pipeline settings

Step 2 Add source stage

Step 3 Add build stage

Step 4 Add deploy stage

Step 5 Review

Add source stage

Source

Source provider: AWS CodeCommit

Repository name: deployment

Branch name: dev

Change detection options:

- Amazon CloudWatch Events (recommended)
- AWS CodePipeline

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

Note: You have defined two CodeCommit repositories. The *deployment* repository contains the Amazon ECS task definition files and CodeDeploy AppSpec files that your pipeline will need, so that is the one you choose here.

- **Branch name:** Choose **dev**.

Add deploy stage

Step 5 Review

Repository name: deployment

Branch name: dev

Change detection options:

- Amazon CloudWatch Events (recommended)
- AWS CodePipeline

Output artifact format:

- CodePipeline default
- Full clone

Cancel Previous Next

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

Note: Skip the build stage.

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1 Choose pipeline settings

Step 2 Add source stage

Step 3 Add build stage

Step 4 Add deploy stage

Step 5 Review

Add build stage Info

Step 3 of 5

Build - optional

Build provider

This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.

Cancel Previous Skip build stage Next

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

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1 Choose pipeline settings

Step 2 Add source stage

Step 3 Add build stage

Step 4 Add deploy stage

Step 5 Review

Add build stage Info

Step 3 of 5

Build - optional

Build provider

Your pipeline will not include a build stage. Are you sure you want to skip this stage?

Cancel Skip Next

○ Deploy provider: Amazon ECS (Blue/Green)

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

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1 Choose pipeline settings

Step 2 Add source stage

Step 3 Add build stage

Step 4 Add deploy stage

Step 5 Review

Add deploy stage Info

Step 4 of 5

You cannot skip this stage

Pipelines must have at least two stages. Your second stage must be either a build or deployment stage. Choose a provider for either the build stage or deployment stage.

Deploy

Deploy provider

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

Amazon ECS (Blue/Green) ▼

Region

US East (N. Virginia) ▼

AWS CodeDeploy application name

Choose one of your existing applications, or create a new one in AWS CodeDeploy.

Create application

AWS CodeDeploy deployment group

Choose one of your existing deployment groups, or create a new one in AWS CodeDeploy.

Cancel Next

- Region: US East (N. Virginia).

- **AWS CodeDeploy application name:** microservices

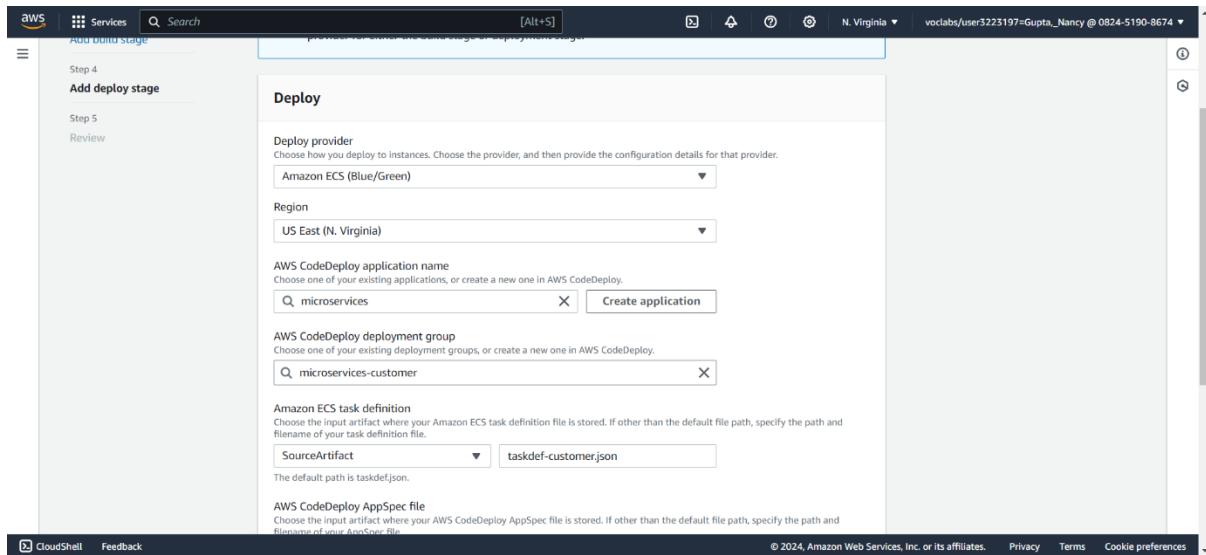
The screenshot shows the AWS CodePipeline 'Create new pipeline' wizard at Step 4: 'Add deploy stage'. A prominent message box states: 'You cannot skip this stage. Pipelines must have at least two stages. Your second stage must be either a build or deployment stage. Choose a provider for either the build stage or deployment stage.' Below this, the 'Deploy' section is visible, showing the 'Deploy provider' dropdown set to 'Amazon ECS (Blue/Green)', the 'Region' dropdown set to 'US East (N. Virginia)', and the 'AWS CodeDeploy application name' input field containing 'microservices'.

- **AWS CodeDeploy deployment group:** microservices-customer

The screenshot shows the AWS CodePipeline 'Create new pipeline' wizard at Step 4: 'Add deploy stage'. A message box states: 'You cannot skip this stage. Pipelines must have at least two stages. Your second stage must be either a build or deployment stage. Choose a provider for either the build stage or deployment stage.' Below this, the 'Deploy' section is visible, showing the 'Deploy provider' dropdown set to 'Amazon ECS (Blue/Green)', the 'Region' dropdown set to 'US East (N. Virginia)', the 'AWS CodeDeploy application name' input field containing 'microservices', and the 'AWS CodeDeploy deployment group' input field containing 'microservices-customer'.

- **Under Amazon ECS task definition:**

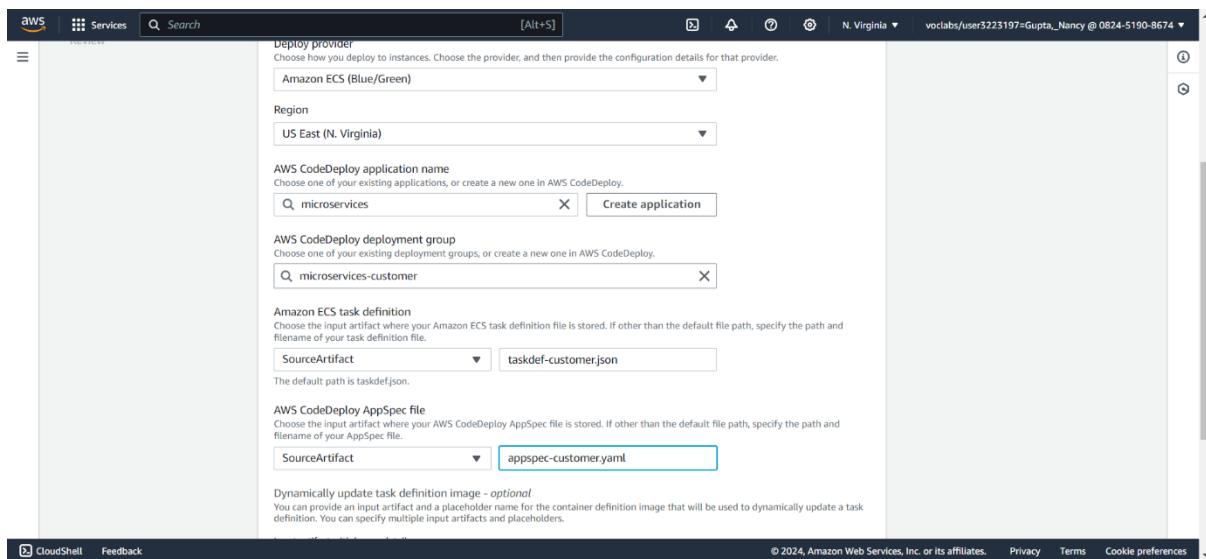
- Set a **SourceArtifact** with a value of taskdef-customer.json



- Under **AWS CodeDeploy AppSpec file**:

- Set a **SourceArtifact** with a value of **appspec-customer.yaml**

Note: Leave the *Dynamically update task definition image* fields blank for now.



aws Services Search [Alt+S] N. Virginia v vocabs/user3223197=Gupta,_Nancy @ 0824-5190-8674 ▾

microservices-customer

Amazon ECS task definition
Choose the input artifact where your Amazon ECS task definition file is stored. If other than the default file path, specify the path and filename of your task definition file.

SourceArtifact taskdef-customer.json

The default path is taskdef.json.

AWS CodeDeploy AppSpec file
Choose the input artifact where your AWS CodeDeploy AppSpec file is stored. If other than the default file path, specify the path and filename of your AppSpec file.

SourceArtifact appspec-customer.yaml

Dynamically update task definition image - optional
You can provide an input artifact and a placeholder name for the container definition image that will be used to dynamically update a task definition. You can specify multiple input artifacts and placeholders.

Input artifact with image details

Placeholder text in the task definition

Configure automatic rollback on stage failure

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

aws Services Search [Alt+S] N. Virginia v vocabs/user3223197=Gupta,_Nancy @ 0824-5190-8674 ▾

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1 Choose pipeline settings
Step 2 Add source stage
Step 3 Add build stage
Step 4 Add deploy stage
Step 5 Review

Review info
Step 5 of 5

Step 1: Choose pipeline settings

Pipeline settings

Pipeline name: update-customer-microservice
Pipeline type: V2
Execution mode: QUEUED
Artifact location: A new Amazon S3 bucket will be created as the default artifact store for your pipeline
Service role name: arn:aws:iam::082451908674:role/PipelineRole

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

aws Services Search [Alt+S] N. Virginia v vocabs/user3223197=Gupta,_Nancy @ 0824-5190-8674 ▾

Variables

Name	Default value	Description
No variables		
No variables defined at the pipeline level in this pipeline.		

Step 2: Add source stage

Source action provider

Source action provider: AWS CodeCommit
RepositoryName: deployment
Default branch: dev
PollForSourceChanges: false
OutputArtifactFormat:

Step 3: Add build stage

Build action provider

Build stage
No build

Step 4: Add deploy stage

Deploy action provider

Deploy action provider
Amazon ECS (Blue/Green)

ApplicationName
microservices

DeploymentGroupName
microservices-customer

TaskDefinitionTemplateArtifact
SourceArtifact

TaskDefinitionTemplatePath
taskdef-customer.json

AppSpecTemplateArtifact
SourceArtifact

AppSpecTemplatePath
appspec-customer.yaml

Configure automatic rollback on stage failure
Disabled

Cancel Previous Create pipeline

Note: After you create the pipeline, it will immediately start to run and will eventually fail on the Deploy stage. Ignore that for now and continue to the next step.

Developer Tools **CodePipeline**

- Source • CodeCommit
- Artifacts • CodeArtifact
- Build • CodeBuild
- Deploy • CodeDeploy
- Pipeline • CodePipeline
 - Getting started
 - Pipelines
 - Pipeline**
 - History
 - Settings
- Settings

Go to resource Feedback

Success
Congratulations! The pipeline update-customer-microservice has been created.

Create a notification rule for this pipeline

Developer Tools > CodePipeline > Pipelines > update-customer-microservice

update-customer-microservice Notify ▾ Edit Stop execution Clone pipeline Release change

Pipeline type: V2 Execution mode: QUEUED

Source In progress

Source
AWS CodeCommit
In progress - Just now
View details

Deploy Didn't Run

Disable transition Start rollback

CloudShell Feedback

The screenshot shows the AWS CodePipeline console. On the left, there's a navigation sidebar with options like Source, Artifacts, Build, Deploy, Pipeline, and Settings. The main area displays a pipeline named 'CodePipeline' with three stages: Source, Deploy, and Pipeline. The Source stage is shown as 'Succeeded - Just now' with a green checkmark. The Deploy stage is shown as 'Failed' with a red X. A red arrow points from the Source stage to the Deploy stage. There are buttons for 'Start rollback', 'Retry stage', and 'Retry failed actions' next to the Deploy stage.

2. Edit the *update-customer-microservice* pipeline to add another source.

The screenshot shows the AWS CodePipeline console with the message 'Congratulations! The pipeline update-customer-microservice has been created.' The pipeline is named 'update-customer-microservice' and is of type V2 with execution mode set to QUEUED. The Source stage is listed as 'Succeeded' with a green checkmark. The pipeline URL is us-east-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/.../edit?region=us-east-...

- In the **Edit: Source** section, choose **Edit stage**, then add an action with these details:

The screenshot shows the 'Edit update-customer-microservice' dialog. In the 'Edit: Pipeline properties' section, the pipeline type is set to V2 and the execution mode is set to QUEUED. In the 'Edit: Variables' section, it says 'Pipeline type V2 required' and shows a table with columns 'Name', 'Default value', and 'Description'. The table is currently empty, indicating 'No variables defined at the pipeline level in this pipeline.'

No triggers

This pipeline does not have any source actions that support triggers. Triggers are only supported for the CodeStarSourceConnection action type.

Source AWS CodeCommit

Deploy Amazon ECS (Blue/Green)

No triggers

This pipeline does not have any source actions that support triggers. Triggers are only supported for the CodeStarSourceConnection action type.

Source AWS CodeCommit

+ Add action

+ Add stage

- **Action name:** Image
- **Action provider:** Amazon ECR
- **Repository name:** customer
- **Image tag:** latest
- **Output artifacts:** image-customer

Edit action

Action name
Choose a name for your action

No more than 100 characters

Action provider

Repository name
Choose an Amazon ECR repository as the source location.

Image tag - optional
Choose the image tag that triggers your pipeline when a change occurs in the image repository.

If an image tag is not selected, defaults to latest

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](#)

Output artifacts
Choose a name for the output of this action.

No more than 100 characters

aws Services Search [Alt+S] N. Virginia v vocabs/user3223197=Gupta_Nancy @ 0824-5190-8674

Developer Tools CodePipeline

- Source • CodeCommit
- Artifacts • CodeArtifact
- Build • CodeBuild
- Deploy • CodeDeploy
- Pipeline • CodePipeline
 - Getting started
 - Pipelines
 - Pipeline**
 - History
 - Settings
 - Settings
- Go to resource
- Feedback

Edit: Triggers

No triggers

This pipeline does not have any source actions that support triggers. Triggers are only supported for the CodeStarSourceConnection action type.

Edit Source

+ Add action group

Source	<input type="text" value="AWS CodeCommit"/>	Image	<input type="text" value="Amazon ECR"/>	+ Add action
X	X	X	X	X

+ Add action group

+ Add stage

3. Edit the **deploy** action of the **update-customer-microservice** pipeline.

- Edit the **update-customer-microservice** pipeline

aws Services Search [Alt+S] N. Virginia v vocabs/user3223197=Gupta_Nancy @ 0824-5190-8674

Developer Tools CodePipeline

- Source • CodeCommit
- Artifacts • CodeArtifact
- Build • CodeBuild
- Deploy • CodeDeploy
- Pipeline • CodePipeline
 - Getting started
 - Pipelines
 - Pipeline**
 - History
 - Settings
 - Settings
- Go to resource
- Feedback

Edit: Deploy

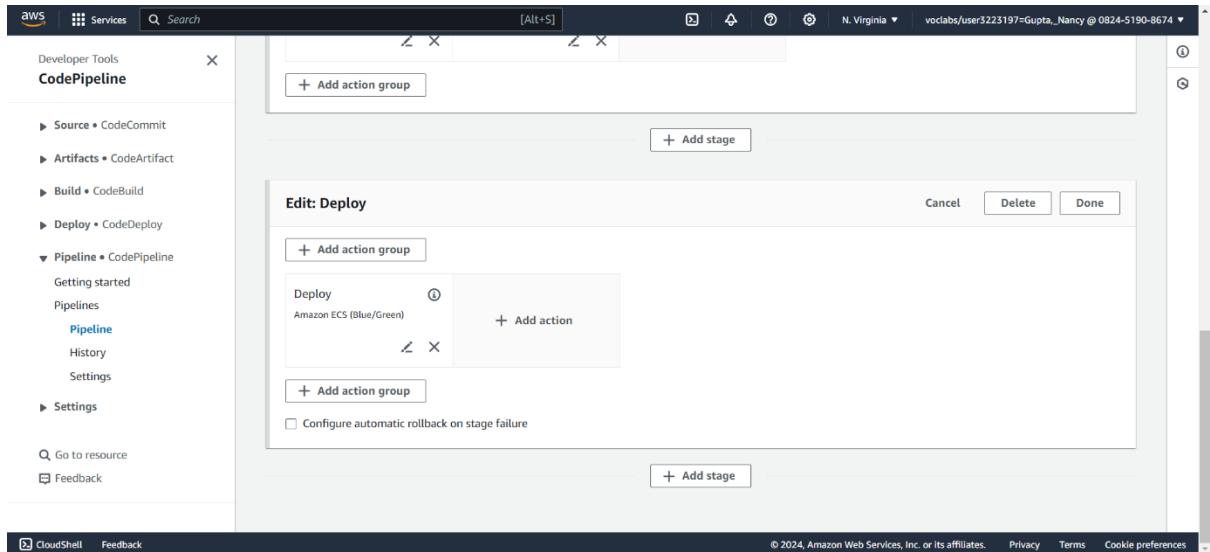
Deploy

Amazon ECS (Blue/Green)

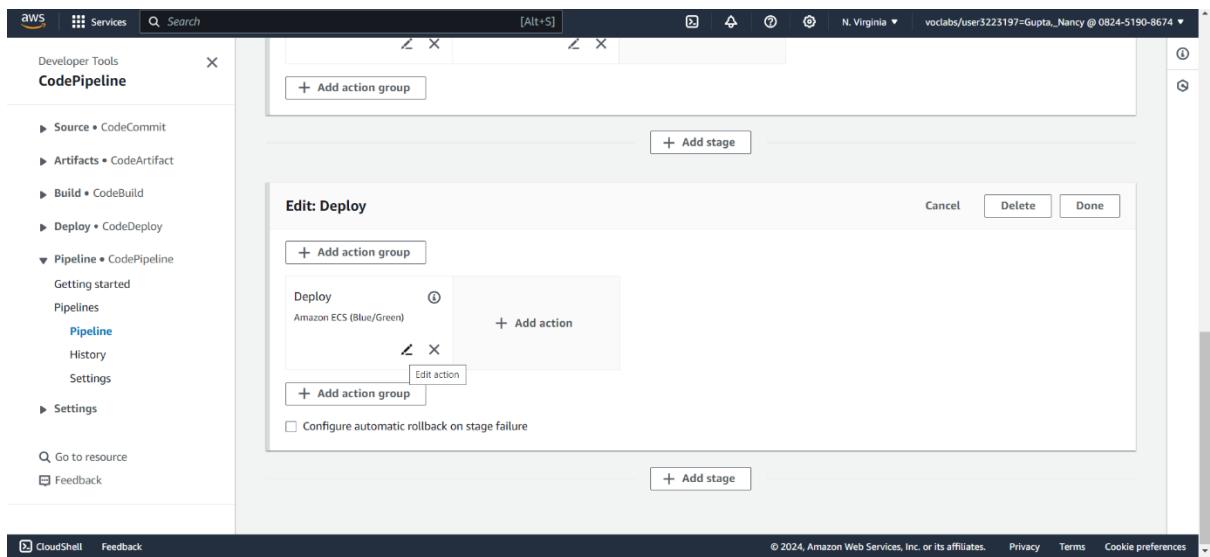
Configure automatic rollback on stage failure

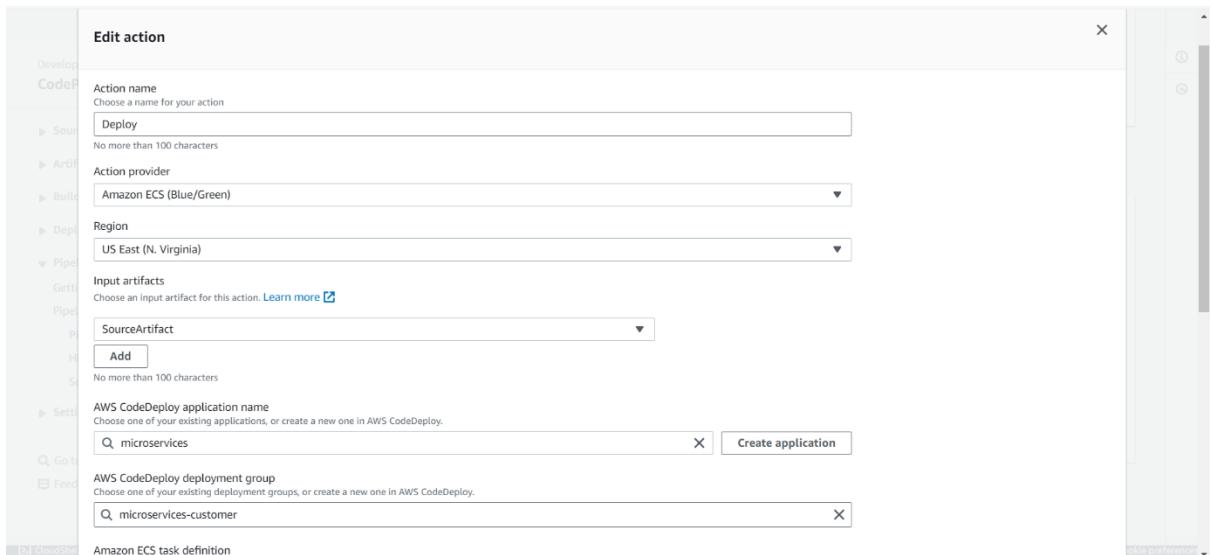
+ Add stage

- In the **Edit: Deploy** section, choose **Edit stage**, then add an input artifact as described below:



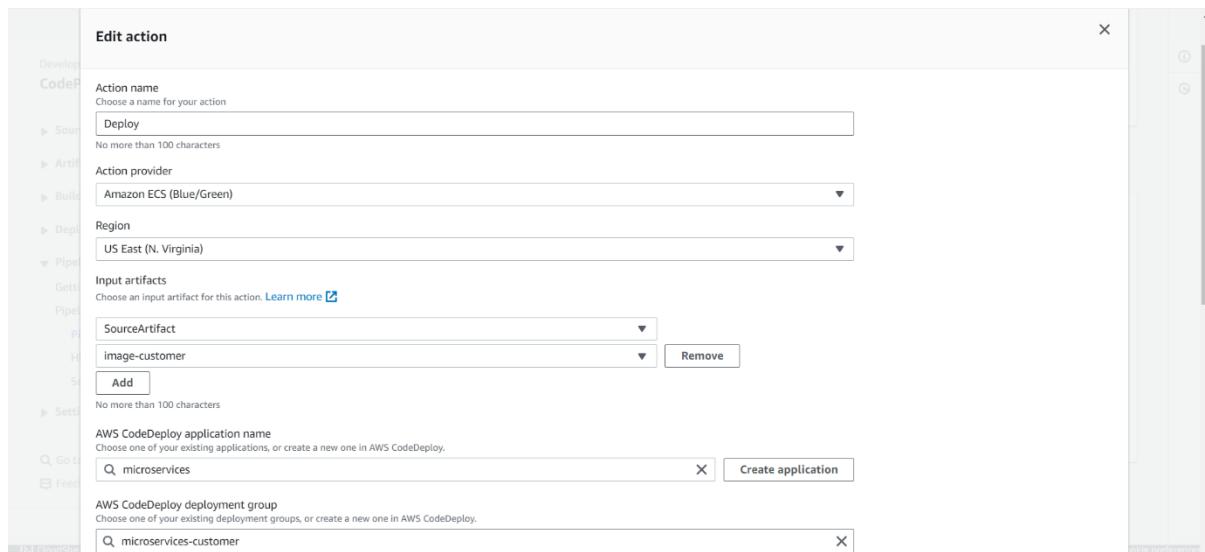
- On the **Deploy Amazon ECS (Blue/Green)** card, choose the edit (pencil) icon.





- Under **Input artifacts**, choose **Add** and then choose **image-customer**.

Note: You should now have *SourceArtifact* and *image-customer* as listed input artifacts.



- Under **Dynamically update task definition image**, for **Input artifact with image details**, choose **image-customer**.
- For **Placeholder text in the task definition**, enter **IMAGE1_NAME**

AWS CodeDeploy deployment group
Choose one of your existing deployment groups, or create a new one in AWS CodeDeploy.

SourceArtifact: microservices-customer

Amazon ECS task definition
Choose the input artifact where your Amazon ECS task definition file is stored. If other than the default file path, specify the path and filename of your task definition file.

SourceArtifact: taskdef-customer.json

AWS CodeDeploy AppSpec file
Choose the input artifact where your AWS CodeDeploy AppSpec file is stored. If other than the default file path, specify the path and filename of your AppSpec file.

SourceArtifact: appspec-customer.yaml

Dynamically update task definition image - optional
You can provide an input artifact and a placeholder name for the container definition image that will be used to dynamically update a task definition. You can specify multiple input artifacts and placeholders.

Input artifact with image details
image-customer

Placeholder text in the task definition
IMAGE1_NAME

Add Remove

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](#)

DeployVariables

Cancel Done

Analysis: Recall that in a previous phase, you entered the *IMAGE1_NAME* placeholder text in the *taskdef-customer.json* file before you pushed it to CodeCommit. In this current task, you configured the logic that will replace the placeholder text with the actual image name that the source phase of the CodePipeline returns.

Developer Tools Services Search [Alt+S]

CodePipeline

- Source • CodeCommit
- Artifacts • CodeArtifact
- Build • CodeBuild
- Deploy • CodeDeploy
- Pipeline • CodePipeline
 - Getting started
 - Pipelines
 - Pipeline**
 - History
 - Settings
- Settings
- Go to resource
- Feedback

+ Add action group

+ Add stage

Edit: Deploy

+ Add action group

Deploy + Add action

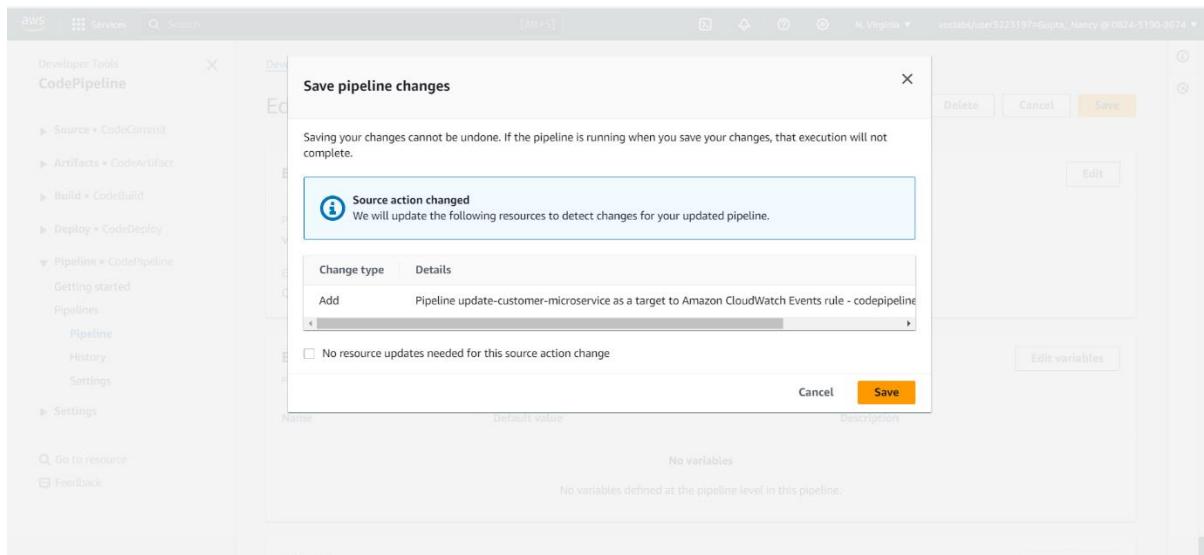
+ Add action group

Configure automatic rollback on stage failure

+ Add stage

Cancel Delete Done

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



Success
Pipeline was saved successfully.

Developer Tools > CodePipeline > Pipelines > update-customer-microservice

Pipeline type: V2 Execution mode: QUEUED

Source Succeeded
Pipeline execution ID: 4583d111-1a93-4373-b987-79087718f986

Source: AWS CodeCommit
Image: Amazon ECR
Status: Succeeded - 36 minutes ago
Execution ID: 4970d713
Details: No executions yet
View details

4970d713 Source: Updated task definition files

Disable transition

Source Succeeded
Pipeline execution ID: 4583d111-1a93-4373-b987-79087718f986

Source: AWS CodeCommit
Image: Amazon ECR
Status: Succeeded - 36 minutes ago
Execution ID: 4970d713
Details: No executions yet
View details

4970d713 Source: Updated task definition files

Deploy Failed
Pipeline execution ID: 4583d111-1a93-4373-b987-79087718f986

Deploy: Amazon ECS (Blue/Green)
Status: Failed - 36 minutes ago
Execution ID: 4970d713
Details: No executions yet
View details

4970d713 Source: Updated task definition files

Task 8.3: Test the CI/CD pipeline for the *customer* microservice

In this task, you will test that the CI/CD pipeline for the *customer* microservice functions as intended.

Important: If you are repeating this task, confirm that all target groups are still associated with the Application Load Balancer. The [Reassociate Target Groups with Load Balancer](#) section of the appendix provides more detail.

1. Launch a deployment of the *customer* microservice on Amazon ECS on Fargate.

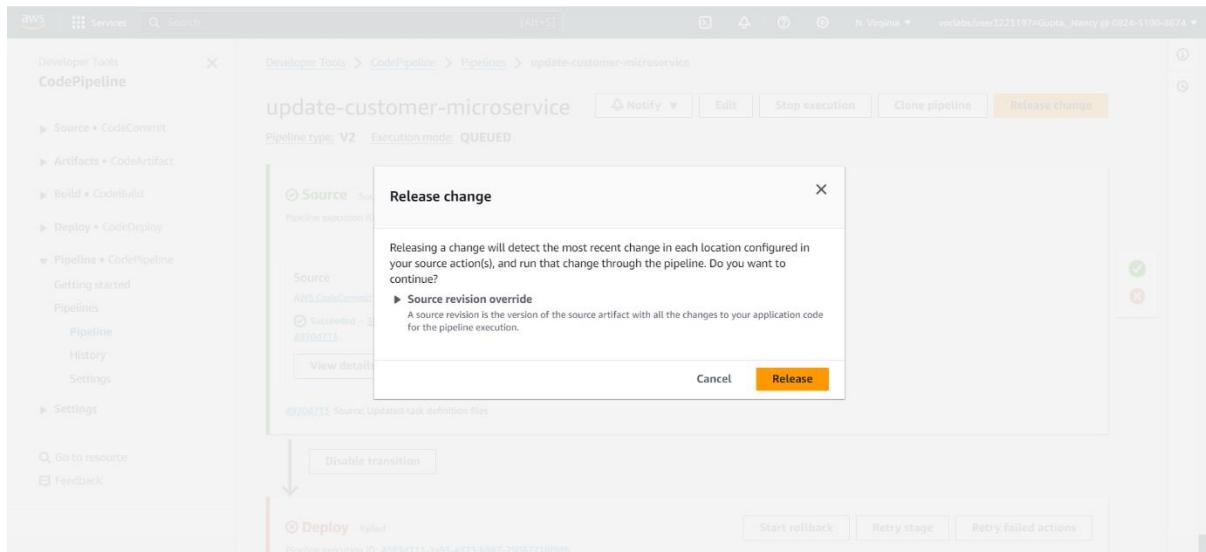
- o Navigate to the CodePipeline console.

The screenshot shows the AWS CodePipeline console. On the left, there's a sidebar with options like Source, Artifacts, Build, Deploy, Pipeline, and Settings. The Pipeline section is expanded, showing 'Getting started', 'Pipelines', and 'Settings'. The main area is titled 'Pipelines' and shows a table with one row. The row for 'update-customer-microservice' has a red circle with a question mark icon next to it, indicating a failure. The table columns include Name, Latest execution status, Latest source revisions, Latest execution started, and Most recent executions. Below the table, there's a note: 'Source - 4970d713: Updated task definition files 39 minutes ago' and three small red circular icons followed by 'View details'.

- o On the **Pipelines** page, choose the link for the pipeline that is named **update-customer-microservice**.

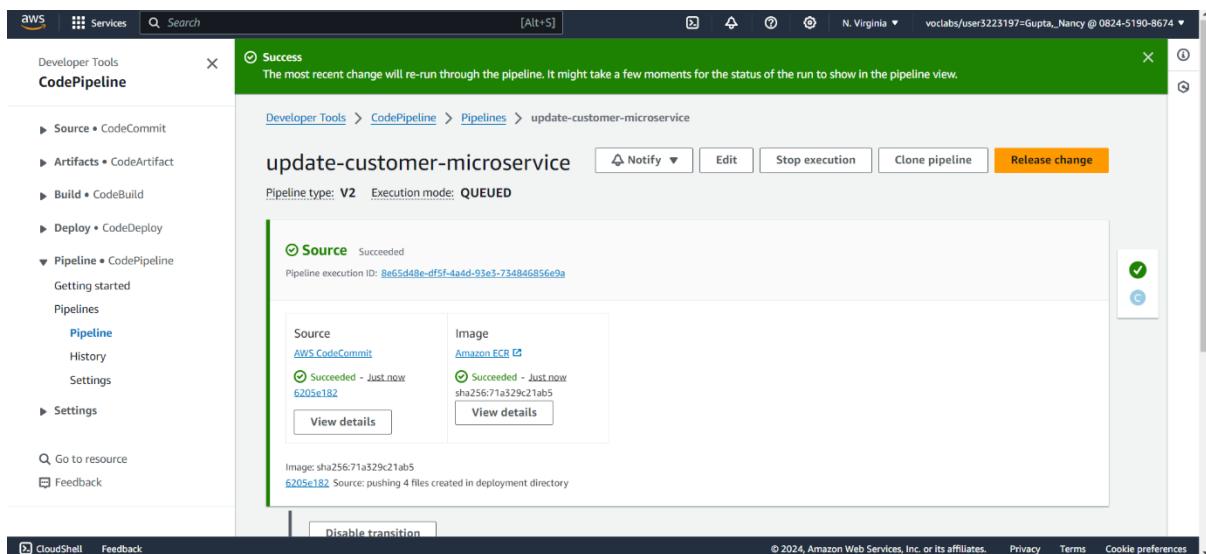
This screenshot shows the details of the 'update-customer-microservice' pipeline. The sidebar is identical to the previous screenshot. The main area shows the pipeline details for 'update-customer-microservice'. It highlights the 'Source' stage, which is listed as 'Succeeded' with a green checkmark icon. Below it, a note says 'Pipeline execution ID: 4583d111-1a93-4373-b987-79087718f986'. The 'Deploy' stage is shown in red with a red error icon, indicating a failure. A button labeled 'Release change' is highlighted with a yellow background. At the bottom, there are buttons for 'Start rollback', 'Retry stage', and 'Retry failed actions'.

- o To force a test of the current pipeline settings, choose **Release change**, and then choose **Release**.



Note: By invoking the pipeline, you created a new revision of the task definition.

Wait for the two *Source* tasks to show a status of *Succeeded - just now*.



- In the **Deploy** section, wait for a **Details** link to appear, and then click the link.

A CodeDeploy page opens in a new browser tab.

Succeeded - 1 minute ago
6205e182

Succeeded - 1 minute ago
sha256:71a329c21ab5
6205e182 Source: pushing 4 files created in deployment directory

Deploy In progress
Pipeline execution ID: Be65d48e-df5f-4a4d-93e3-734846856e9a

Deploy
Amazon ECS (Blue/Green)
In progress - 1 minute ago

Image: sha256:71a329c21ab5
6205e182 Source: pushing 4 files created in deployment directory

Action name: Deploy Status: In progress

Summary Input

Status
In progress

Action execution ID
b37950f8-d43d-4db0-915a-693804e87aba

Message
Deployment d-ZYQT3I8U5 created

Execution details
[View in CodeDeployToECS](#)

Done

Developer Tools > CodeDeploy > Deployments > d-ZYQT3I8U5

d-ZYQT3I8U5

Deployment status

- Step 1: Deploying replacement task set
Completed: Succeeded 100%
- Step 2: Test traffic route setup
Completed: Succeeded 100%
- Step 3: Rerouting production traffic to replacement task set
100% traffic shifted: In progress
- Step 4: Wait 5 minutes 0 seconds
Not started
- Step 5: Terminate original task set
0%

Traffic shifting progress

Original	Replacement
0%	100%
Original task set not serving traffic	Replacement task set serving traffic

2. Observe the progress in CodeDeploy.

The screenshot shows the AWS CodeDeploy console with a deployment named 'd-ZYQT3I8US'. The 'Deployment status' section lists five steps: Step 1 (Deploying replacement task set), Step 2 (Test traffic route setup), Step 3 (Rerouting production traffic to replacement task set), Step 4 (Wait 5 minutes 0 seconds), and Step 5 (Terminate original task set). Step 1, 2, and 3 are completed with a 'Succeeded' status. Step 4 is 'In progress' at 50%. Step 5 has a 0% completion bar. The 'Traffic shifting progress' section shows 'Original' at 0% and 'Replacement' at 100%, indicating the original task set is not serving traffic and the replacement task set is serving traffic.

3.

- Scroll to the bottom of the page, and notice the **Deployment lifecycle events** section.

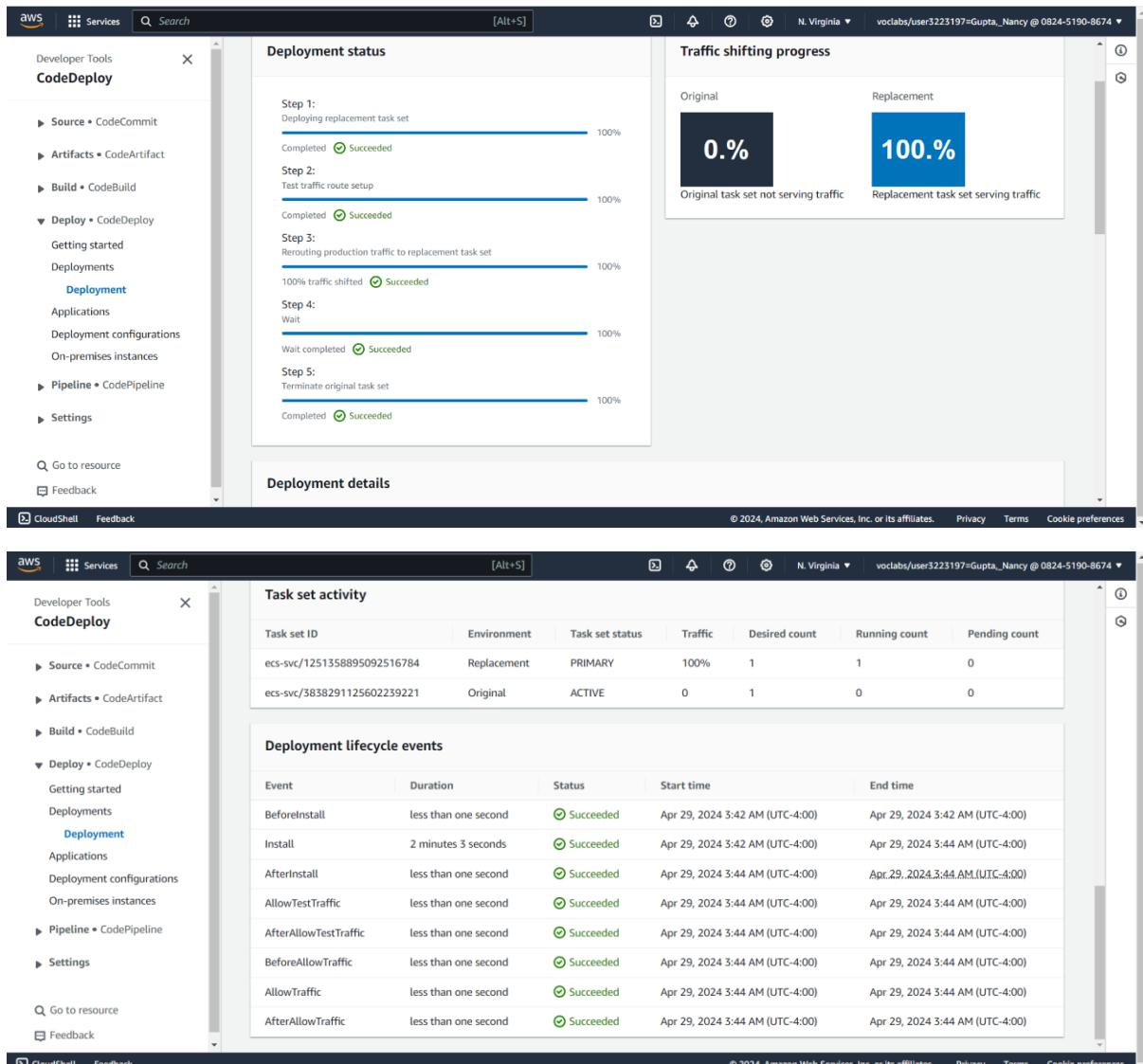
The screenshot shows the AWS CodeDeploy console with the 'Task set activity' table and the 'Deployment lifecycle events' table.

Task set ID	Environment	Task set status	Traffic	Desired count	Running count	Pending count
ecs-svc/1251358895092516784	Replacement	PRIMARY	100%	1	1	0
ecs-svc/3838291125602239221	Original	ACTIVE	0	1	0	0

Event	Duration	Status	Start time	End time
BeforeInstall	less than one second	Succeeded	Apr 29, 2024 3:42 AM (UTC-4:00)	Apr 29, 2024 3:42 AM (UTC-4:00)
Install	2 minutes 3 seconds	Succeeded	Apr 29, 2024 3:42 AM (UTC-4:00)	Apr 29, 2024 3:44 AM (UTC-4:00)
AfterInstall	less than one second	Succeeded	Apr 29, 2024 3:44 AM (UTC-4:00)	Apr 29, 2024 3:44 AM (UTC-4:00)
AllowTestTraffic	less than one second	Succeeded	Apr 29, 2024 3:44 AM (UTC-4:00)	Apr 29, 2024 3:44 AM (UTC-4:00)
AfterAllowTestTraffic	less than one second	Succeeded	Apr 29, 2024 3:44 AM (UTC-4:00)	Apr 29, 2024 3:44 AM (UTC-4:00)
BeforeAllowTraffic	less than one second	Succeeded	Apr 29, 2024 3:44 AM (UTC-4:00)	Apr 29, 2024 3:44 AM (UTC-4:00)
AllowTraffic	less than one second	Succeeded	Apr 29, 2024 3:44 AM (UTC-4:00)	Apr 29, 2024 3:44 AM (UTC-4:00)
AfterAllowTraffic	less than one second	Succeeded	Apr 29, 2024 3:44 AM (UTC-4:00)	Apr 29, 2024 3:44 AM (UTC-4:00)

Tip: If you see a "Primary task group must be behind listener" error, refer to the [Reassociate Target Groups with Load Balancer](#) section in the appendix.

Within a few minutes, if everything was configured correctly, all of the deployment lifecycle events should succeed. Don't wait for that to happen—move to the next step. Keep this page open.



4. Load the customer microservice in a browser tab and test it.

The screenshot shows the AWS EC2 Dashboard. The left sidebar includes links for EC2 Dashboard, EC2 Global View, Events, Console-to-Code Preview, Instances (selected), Images, and Elastic Block Store. The main content area is titled 'Load balancers (1)' and displays a table with one row for 'microservicesLB'. The table columns include Name, DNS name, State, VPC ID, Availability Zones, Type, and Date created. The 'microservicesLB' row has a 'DNS name' value of 'microservicesLB-1849491...'. A message at the bottom says '0 load balancers selected'.

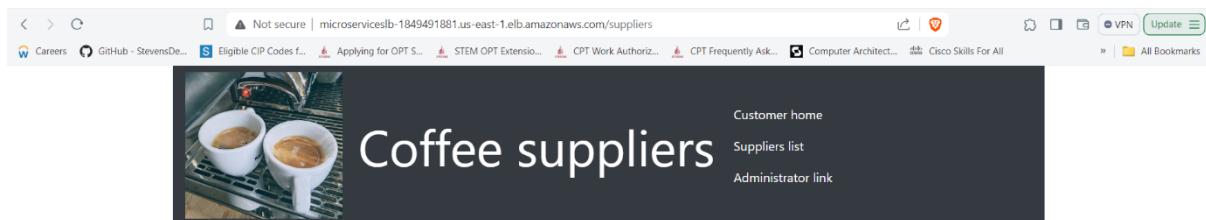
- Locate the **DNS name** value of the **microservicesLB** load balancer, and paste it into a new browser tab.

- The customer microservice loads. If it doesn't load, add :8080 to the end of the URL and try again.

Analysis: Recall that your load balancer has two listeners: one on port 80 and another on port 8080. Port 8080 is where the replacement task set will run for the first 5 minutes. Therefore, if you load the :80 URL within the first 5 minutes, the customer microservice page might not load, but you should already see the page at 8080. Then, after 5 minutes, you should see that the microservice is available at both ports.

- In the café web application, choose **List of suppliers** or **Suppliers list**.

The suppliers page loads. It should *not* have the edit or add supplier buttons because it is a customer page.



5. Observe the running tasks in the Amazon ECS console.

- Navigate to the Amazon ECS console.

- In the clusters list, choose the link for **microservices-serverlesscluster**.

On the **Services** tab, notice that the *customer-microservice* service appears. The **Deployments and tasks** status will change as the blue/green deployment advances through its lifecycle events.

The screenshot shows the AWS Elastic Container Service (ECS) Cluster Overview page. The cluster name is 'microservices-serverlesscluster'. The ARN is listed as 'arn:aws:ecs:us-east-1:082451908674:cluster/microservices-serverlesscluster'. The status is 'Active'. CloudWatch monitoring is set to 'Default'. There are no registered container instances. Under the 'Services' section, there is one service named 'Draining' with an ARN of 'arn:aws:ecs:us-east-1:082451908674:service/microservices-serverlesscluster/Draining'. It has 2 tasks, both of which are pending. There is 1 running task. The 'Services' tab is selected.

This screenshot is identical to the one above, but the 'Tasks' tab is selected. It shows the same cluster details and the 'Draining' service. The tasks table lists two tasks for the 'Draining' service, both of which are pending. There is 1 running task.

- Choose the **Tasks** tab.

Here you can see the actual tasks that are running. You might have more than one task running per service that you defined.

The screenshot shows the AWS CloudWatch Metrics Metrics Insights page. At the top, there's a search bar and navigation links for 'CloudWatch Metrics' and 'Metrics Insights'. Below the search bar, there's a section titled 'Metrics Insights' with a table showing metrics from various services like Amazon CloudWatch Metrics, AWS Lambda, and AWS Lambda Metrics. The table includes columns for 'Metric Name', 'Source', 'Unit', 'Value Type', and 'Last Value'. A large orange button labeled 'Create new insight' is prominently displayed.

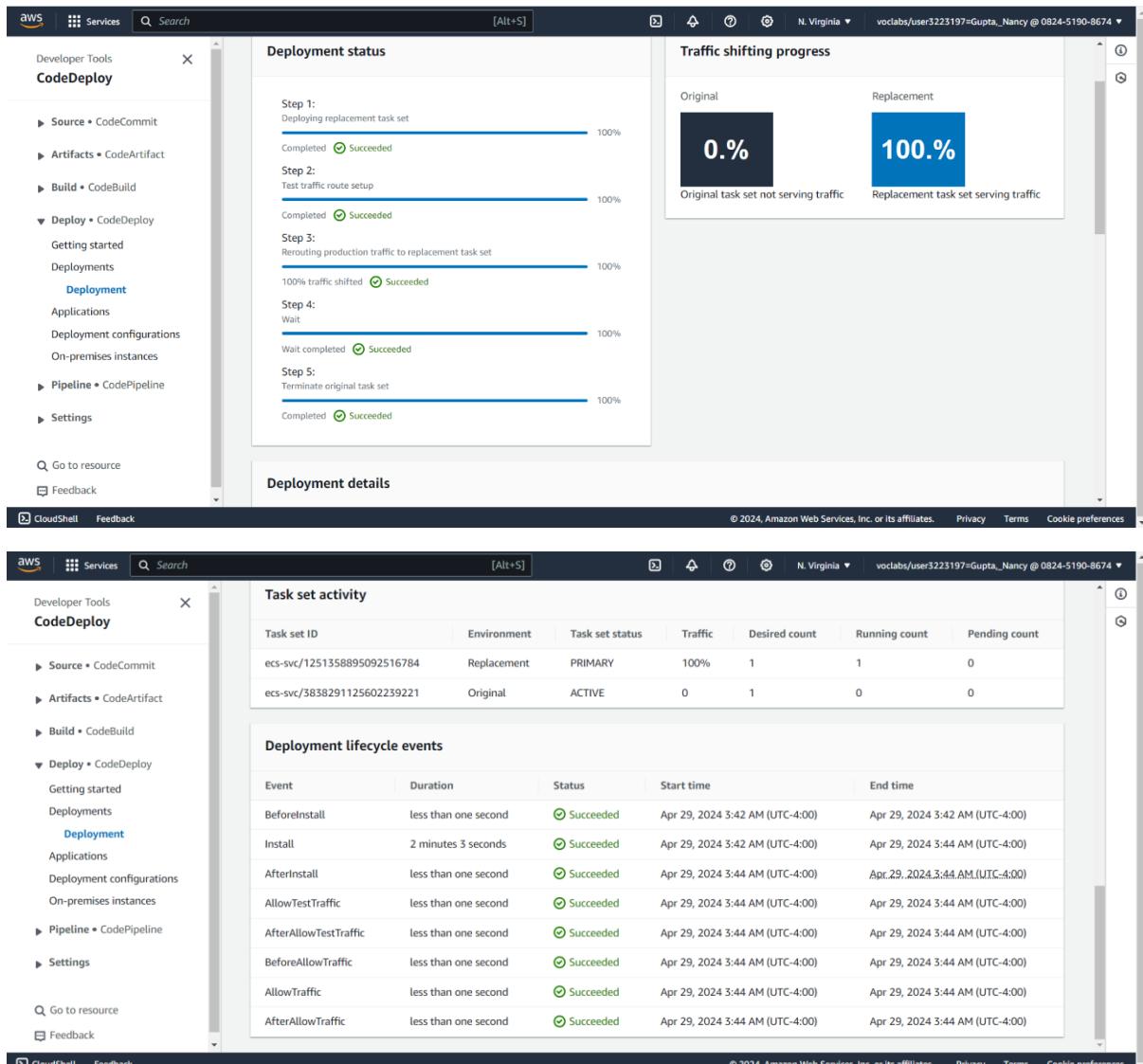
- Choose the link for one of the listed tasks. You might only have one.

Here you can see the actual container details and the configuration information, such as the IP addresses that are associated with the running container.

The screenshot shows the AWS CloudWatch Metrics Metrics Insights page. At the top, there's a search bar and navigation links for 'CloudWatch Metrics' and 'Metrics Insights'. Below the search bar, there's a section titled 'Metrics Insights' with a table showing metrics from various services like Amazon CloudWatch Metrics, AWS Lambda, and AWS Lambda Metrics. The table includes columns for 'Metric Name', 'Source', 'Unit', 'Value Type', and 'Last Value'. A large orange button labeled 'Create new insight' is prominently displayed.

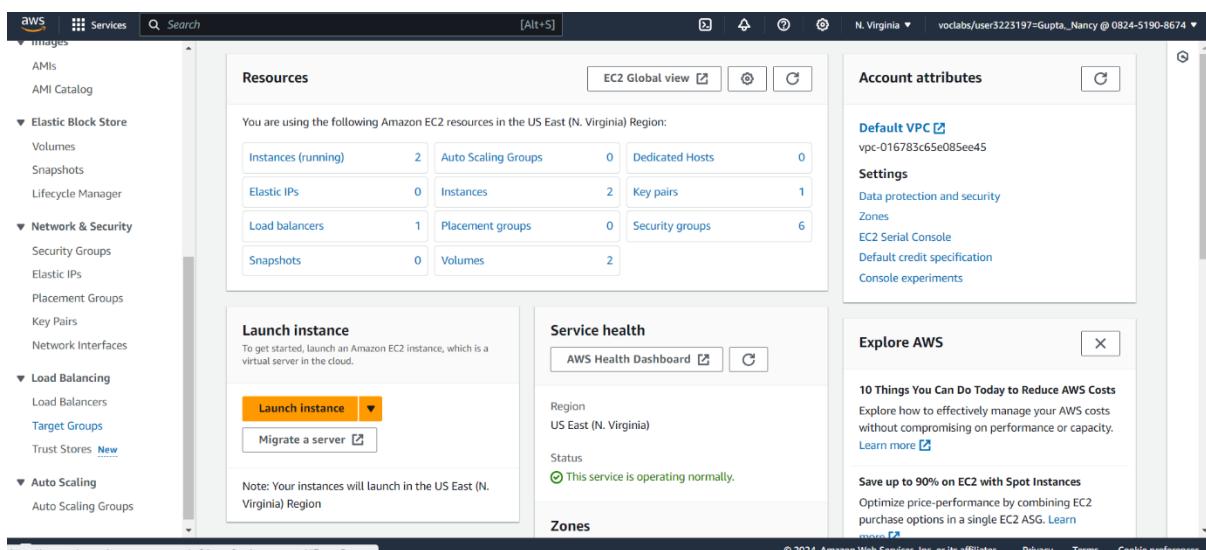
6. Return to the CodeDeploy page that is open in another browser tab.

You should now see that all five steps of the deployment succeeded and the replacement task set is now serving traffic.



7. Observe the load balancer and target group settings.

- In the Amazon EC2 console, choose **Target Groups**.



You might notice that the *customer-tg-two* target group is no longer associated with the load balancer. This is because CodeDeploy is managing the load balancer listener rules and might have determined that some of the target groups are no longer needed.

The screenshot shows the AWS EC2 Target Groups page. The left sidebar includes sections for AMIs, AMI Catalog, Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), Load Balancing (Load Balancers, Target Groups, Trust Stores), and Auto Scaling (Auto Scaling Groups). The main content area displays a table titled "Target groups (4) Info" with columns for Name, ARN, Port, Protocol, Target type, and Load balancer. The table lists four entries: "customer-tg-one" (arn:aws:elasticloadbalancing:us-east-1:082451908674:targetgroup/customer-tg-one), "customer-tg-two" (arn:aws:elasticloadbalancing:us-east-1:082451908674:targetgroup/customer-tg-two), "employee-tg-one" (arn:aws:elasticloadbalancing:us-east-1:082451908674:targetgroup/employee-tg-one), and "employee-tg-two" (arn:aws:elasticloadbalancing:us-east-1:082451908674:targetgroup/employee-tg-two). The "customer-tg-two" entry has a status of "None associated". A modal window titled "0 target groups selected" is open, stating "Select a target group above."

- o Observe the HTTP:80 listener rules.

The default rule has changed here. The default "if no other rule applies" rule previously pointed to *customer-tg-two*, but now it points to *customer-tg-one*. This is because CodeDeploy actively managed your Application Load Balancer.

The screenshot shows the AWS Application Load Balancer (ALB) configuration for the load balancer "microservicesLB". The left sidebar includes sections for AMIs, AMI Catalog, Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), Load Balancing (Load Balancers, Target Groups, Trust Stores), and Auto Scaling (Auto Scaling Groups). The main content area shows the "Details" tab for the load balancer, which includes fields for Load balancer type (Application), Status (Active), VPC (vpc-0d0d3f1f2eb589fdd), IP address type (IPv4), Scheme (Internet-facing), Hosted zone (Z35SXDOTRQ7X7K), Availability Zones (subnet-0c1ad2d13f3070aa7, subnet-089933df8ddacc089), Date created (April 29, 2024, 02:58 (UTC-04:00)), and DNS name (microservicesLB-1849491881.us-east-1.elb.amazonaws.com). Below this, the "Listeners and rules" tab is selected, showing two listeners: "HTTP:80" and "HTTPS:443". The "HTTP:80" listener is configured to use the "customer-tg-one" target group. The "Listeners and rules (2) Info" section provides a detailed description of how traffic is routed based on protocol and port.

The screenshot shows the AWS CloudFront console with the 'Listeners and rules' tab selected. At the top, it displays the Load balancer ARN: arn:aws:elasticloadbalancing:us-east-1:082451908674:loadbalancer/app/microservicesLB/27fed4d4baeb9f7a and the DNS name: microservicesLB-1849491881.us-east-1.elb.amazonaws.com (A Record). Below this, there are two listeners listed:

- HTTP:80**: Forward to target group (customer-tg-one, 100%), Group-level stickiness: Off.
- HTTP:8080**: Forward to target group (customer-tg-one, 100%), Group-level stickiness: Off.

The screenshot shows the AWS EC2 Load Balancers console. Under the 'Load balancers' section, it shows the details for the 'microservicesLB' load balancer. The 'HTTP:80' listener is selected. The details show:

- Protocol: Port: HTTP:80
- Load balancer: microservicesLB
- Default actions: Forward to target group (customer-tg-one, 100%), Group-level stickiness: Off.

The screenshot shows the AWS CloudFront console. Under the 'Listeners and rules' section, it shows the 'Listener rules' for the 'HTTP:80' listener. There are two rules listed:

- Path Pattern is /admin/***: Forward to target group (employee-tg-two, 100%), Group-level stickiness: Off.
- Default**: Forward to target group (customer-tg-one, 100%), Group-level stickiness: Off.

- Observe the HTTP:8080 listener rules.

The two rules still forward to the "one" target groups.

The screenshot shows the AWS CloudFront console. On the left, a sidebar lists services like AMIs, Elastic Block Store, Network & Security, Load Balancing, Auto Scaling, and others. The main area is titled "HTTP:8080 info" under "Load balancers > microservicesLB > HTTP:8080 listener".
Details:
Protocol: Port: HTTP:8080
Load balancer: [microservicesLB](#)
Default actions:
Forward to target group:

- [customer-tg-one](#): 1 (100%)

Listener ARN: arn:aws:elasticloadbalancing:us-east-1:082451908674:listener/app/microservicesLB/27fed4d4baeb9f7a/dfca4386c207d6c
Rules (selected) | Tags
Listener rules (2) Info
Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.

Name tag	Priority	Conditions (If)	Actions (Then)	ARN
-	1	Path Pattern is /admin/*	Forward to target group <ul style="list-style-type: none">employee-tg-one: 1 (100%)	ARN
Default	Last (default)	If no other rule applies	Forward to target group <ul style="list-style-type: none">customer-tg-one: 1 (100%)	ARN

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

This screenshot is identical to the one above, showing the AWS CloudFront console with the same listener configuration for "microservicesLB". The details, rules, and overall structure are the same.

Congratulations! You successfully deployed one of the two microservices to Amazon ECS on Fargate by using a CI/CD pipeline.

Task 8.4: Create a pipeline for the *employee* microservice

In this task, you will create the pipeline for the *employee* microservice.

Pipelines

Name	Latest execution status	Latest source revisions	Latest execution started	Most recent executions
update-customer-microservice	Succeeded	Source - 6205e182: pushing 4 files created in deployment directory Image - sha256:7:	24 minutes ago	View details

1. Create a pipeline for the *employee* microservice with the following specifications:

- Pipeline name: update-employee-microservice

Choose pipeline settings

Pipeline name: update-employee-microservice

Pipeline type: V2

Execution mode: Queued (Pipeline type V2 required)

- Role ARN: **PipelineRole**

Screenshot of the AWS CodePipeline console showing the configuration of a new pipeline.

Pipeline type: Pipeline type V2 is selected. A note states: "The pipeline type determines the pipeline structure and availability of parameters such as triggers. Pipeline type selection will impact features and pricing. [Which pipeline is right for me?](#)"

Execution mode: Execution mode V2 is selected. A note states: "Choose the execution mode for your pipeline. This determines how the pipeline is run." Options include Superseded, Queued (Pipeline type V2 required), and Parallel (Pipeline type V2 required).

Service role: Existing service role is selected. A note states: "Create a service role in your account." An ARN field contains: arn:aws:iam::082451908674:role/PipelineRole.

Variables: A note states: "You can add variables at the pipeline level. You can choose to assign the value when you start the pipeline. Choosing this option requires pipeline type V2. [Learn more](#)."

Advanced settings:

- Artifact store:** Custom location is selected. A note states: "Choose an existing S3 location from your account in the same region and account as your pipeline." Default location is also listed.
- Bucket:** Bucket name is codepipeline-us-east-1-788663725808.
- Encryption key:** Default AWS Managed Key is selected. A note states: "Use the default artifact store (Amazon S3 codepipeline-us-east-1-788663725808) designated in the same region and account as your pipeline." Customer Managed Key is also listed.

Buttons at the bottom: Cancel, Next.

▪ Source provider: AWS CodeCommit

- Repository name: deployment
- Branch name: dev

Step 1
Choose pipeline settings

Step 2
Add source stage

Step 3
Add build stage

Step 4
Add deploy stage

Step 5
Review

Add source stage Info

Step 2 of 5

Source

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

AWS CodeCommit

Repository name
Choose a repository that you have already created where you have pushed your source code.

Q deployment X

Branch name
Choose a branch of the repository

Q dev X

Change detection options
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.

Amazon CloudWatch Events (recommended)
Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs

AWS CodePipeline
Use AWS CodePipeline to check periodically for changes

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

Step 5 Review

Repository name
Choose a repository that you have already created where you have pushed your source code.

Q deployment X

Branch name
Choose a branch of the repository

Q dev X

Change detection options
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.

Amazon CloudWatch Events (recommended)
Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs

AWS CodePipeline
Use AWS CodePipeline to check periodically for changes

Output artifact format
Choose the output artifact format.

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

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

Cancel Previous Next

- Deploy provider: **Amazon ECS (Blue/Green)**
- AWS CodeDeploy application name: **microservices**
- AWS CodeDeploy deployment group: **microservices-employee**
- Amazon ECS task definition: **SourceArtifact**
 - Path: taskdef-employee.json
 - AWS CodeDeploy AppSpec file: **SourceArtifact**
 - Path: appspec-employee.yaml

Screenshot of the AWS CodePipeline 'Create new pipeline' wizard, Step 4: Add deploy stage.

The 'Deploy provider' dropdown is set to 'Amazon ECS (Blue/Green)'. The 'Region' dropdown is set to 'US East (N. Virginia)'. The 'AWS CodeDeploy application name' search bar contains 'microservices' and has a 'Create application' button. The 'AWS CodeDeploy deployment group' search bar contains 'microservices-employee'.

Screenshot of the AWS CodePipeline 'Create new pipeline' wizard, Step 4: Add deploy stage.

The 'Deploy provider' dropdown is set to 'Amazon ECS (Blue/Green)'. The 'Region' dropdown is set to 'US East (N. Virginia)'. The 'AWS CodeDeploy application name' search bar contains 'microservices' and has a 'Create application' button. The 'AWS CodeDeploy deployment group' search bar contains 'microservices-employee'.

Below these fields, there is a section for 'Amazon ECS task definition' with a 'SourceArtifact' dropdown set to 'taskdef-employee.json'.

Screenshot of the AWS CodePipeline 'Create new pipeline' wizard, Step 4: Add deploy stage.

The 'Deploy provider' dropdown is set to 'Amazon ECS (Blue/Green)'. The 'Region' dropdown is set to 'US East (N. Virginia)'. The 'AWS CodeDeploy application name' search bar contains 'microservices-employee'.

Below these fields, there is a section for 'Amazon ECS task definition' with a 'SourceArtifact' dropdown set to 'taskdef-employee.json'.

Below the task definition section, there is a section for 'AWS CodeDeploy AppSpec file' with a 'SourceArtifact' dropdown set to 'appspec-employee.yaml'.

At the bottom of the page, there is a 'Configure automatic rollback on stage failure' checkbox.

aws Services Search [Alt+S] N. Virginia v vocabs/user5223197=Gupta_Nancy @ 0824-5190-8674 ▾

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1 Choose pipeline settings Review Info Step 5 of 5

Step 2 Add source stage Step 3 Add build stage Step 4 Add deploy stage Step 5 Review

Step 1: Choose pipeline settings

Pipeline settings

Pipeline name: update-employee-microservice
Pipeline type: V2
Execution mode: QUEUED
Artifact location: codepipeline-us-east-1-788663725808
Service role name: arn:aws:iam::082451908674:role/PipelineRole

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

aws Services Search [Alt+S] N. Virginia v vocabs/user5223197=Gupta_Nancy @ 0824-5190-8674 ▾

Variables

Name	Default value	Description
No variables		

No variables defined at the pipeline level in this pipeline.

Step 2: Add source stage

Source action provider

Source action provider: AWS CodeCommit
RepositoryName: deployment
Default branch: dev
PollForSourceChanges: false
OutputArtifactFormat: CODE_ZIP

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

aws Services Search [Alt+S] N. Virginia v vocabs/user5223197=Gupta_Nancy @ 0824-5190-8674 ▾

PollForSourceChanges: false
OutputArtifactFormat: CODE_ZIP

Step 3: Add build stage

Build action provider

Build stage: No build

Step 4: Add deploy stage

Deploy action provider

Deploy action provider: Amazon ECS (Blue/Green)
ApplicationName: microservices
DeploymentGroupName:

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

aws Services Search [Alt+S] N. Virginia v vocabs/user3223197=Gupta,_Nancy @ 0824-5190-8674 ▾

☰ Deploy action provider

Deploy action provider
Amazon ECS (Blue/Green)
ApplicationName
microservices
DeploymentGroupName
microservices-employee
TaskDefinitionTemplateArtifact
SourceArtifact
TaskDefinitionTemplatePath
taskdef-employee.json
AppSpecTemplateArtifact
SourceArtifact
AppSpecTemplatePath
appspec-employee.yaml
Configure automatic rollback on stage failure
Disabled

Cancel Previous Create pipeline

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

aws Services Search [Alt+S] N. Virginia v vocabs/user3223197=Gupta,_Nancy @ 0824-5190-8674 ▾

Developer Tools CodePipeline

Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline History Settings
Settings

Go to resource Feedback

Success Congratulations! The pipeline update-employee-microservice has been created.

Create a notification rule for this pipeline

Developer Tools > CodePipeline > Pipelines > update-employee-microservice

update-employee-microservice Notify Edit Stop execution Clone pipeline Release change

Pipeline type: V2 Execution mode: QUEUED

Source In progress
Pipeline execution ID: 446003bb-c4af-4b9e-9f09-30315df9dd3

Source Didn't Run
No executions yet

Disable transition

Deploy Didn't Run Start rollback

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

aws Services Search [Alt+S] N. Virginia v vocabs/user3223197=Gupta,_Nancy @ 0824-5190-8674 ▾

Developer Tools CodePipeline

Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline History Settings
Settings

Go to resource Feedback

Success Congratulations! The pipeline update-employee-microservice has been created.

Create a notification rule for this pipeline

Developer Tools > CodePipeline > Pipelines > update-employee-microservice

update-employee-microservice Notify Edit Stop execution Clone pipeline Release change

Pipeline type: V2 Execution mode: QUEUED

Source Succeeded
Pipeline execution ID: 446003bb-c4af-4b9e-9f09-30315df9dd3

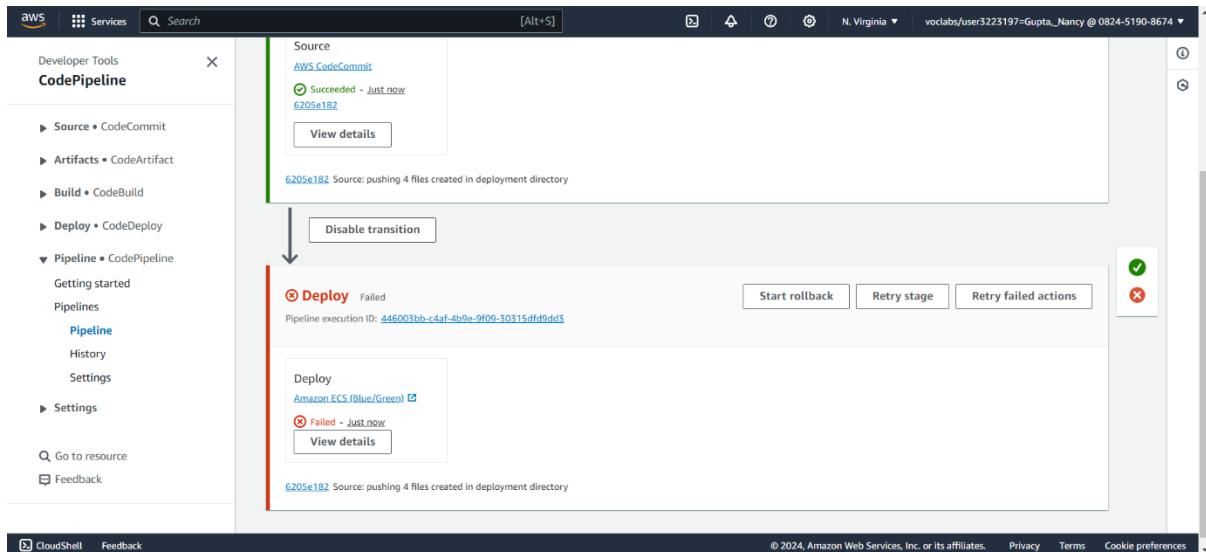
Source Succeeded - Just now
6205e182

View details

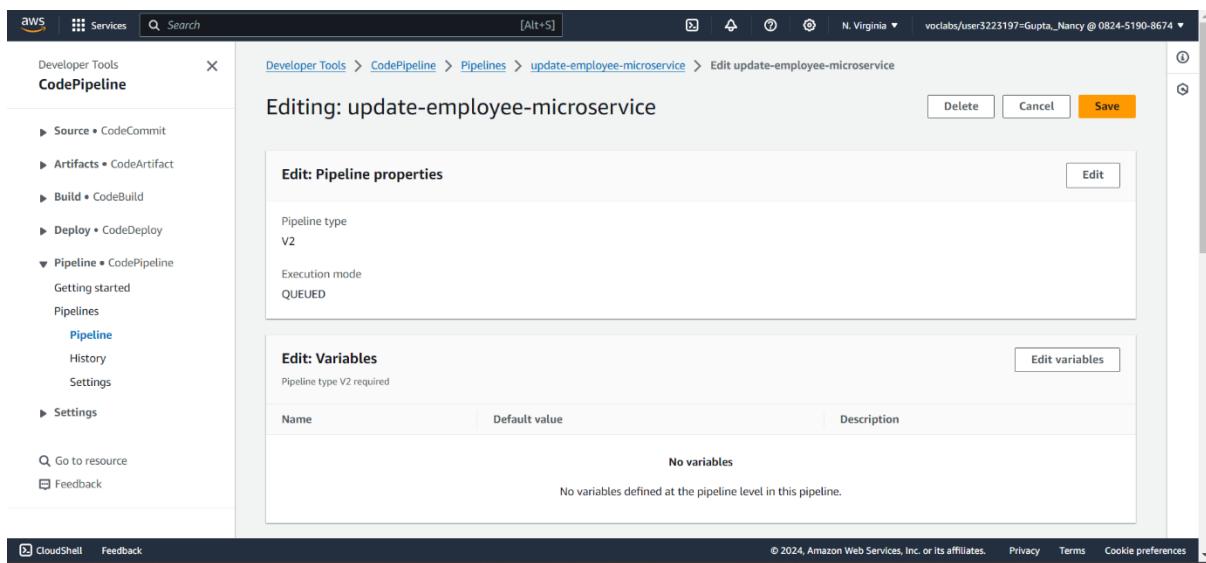
6205e182 Source: pushing 4 files created in deployment directory

Disable transition

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



2. Add another *source* to the employee microservice pipeline. Add an action with the following details:



The screenshot shows the AWS CodePipeline console interface. On the left, there's a navigation sidebar with options like Source, Artifacts, Build, Deploy, Pipeline, and Settings. The main area is divided into three sections: 'Edit: Triggers' (No triggers), 'Edit: Source' (Source: AWS CodeCommit), and 'Edit: Deploy' (Deploy: Amazon ECS (Blue/Green)). A note at the bottom says 'Configure automatic rollback on stage failure'. The top right shows account information: N. Virginia, vclabs/user3223197=Gupta_Nancy @ 0824-5190-8674.

This screenshot shows the 'Edit: Source' stage from the previous interface. It has a 'Source' section with 'AWS CodeCommit' selected. Below it is a 'Configure automatic rollback on stage failure' link. The top right has 'Edit stage' and 'Done' buttons. The bottom right shows copyright information: © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences.

- Action name: **Image**
- Action provider: **Amazon ECR**
- Repository name: **employee**
- Image tag: **latest**
- Output artifacts: **image-employee**

Edit action

Action name
Choose a name for your action

No more than 100 characters

Action provider
Amazon ECR

Repository name
Choose an Amazon ECR repository as the source location.

Image tag - optional
Choose the image tag that triggers your pipeline when a change occurs in the image repository.

If an image tag is not selected, defaults to latest

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](#)

Output artifacts
Choose a name for the output of this action.

No more than 100 characters

Developer Tools **CodePipeline**

Source • CodeCommit

Artifacts • CodeArtifact

Build • CodeBuild

Deploy • CodeDeploy

Pipeline • CodePipeline

Getting started

Pipelines

Pipeline

History

Settings

Settings

Go to resource

Feedback

Edit: Triggers

No triggers

This pipeline does not have any source actions that support triggers. Triggers are only supported for the CodeStarSourceConnection action type.

Edit: Source

+ Add action group

Source	AWS CodeCommit	Image	Amazon ECR
<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>

+ Add action

+ Add action group

+ Add stage

Edit: Deploy

Edit stage

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

3. Edit the Amazon ECS (Blue/Green) action in the deploy stage:

- Add another *input artifact* and choose **image-employee**.
- Under **Dynamically update task definition image**, for **Input artifact with image details**, choose **image-employee**.
- For **Placeholder text in the task definition**, enter **IMAGE1_NAME**

Screenshot of the AWS CodePipeline console showing the pipeline configuration interface.

The left sidebar shows the navigation menu:

- Developer Tools
- CodePipeline**
- Source • CodeCommit
- Artifacts • CodeArtifact
- Build • CodeBuild
- Deploy • CodeDeploy
- Pipeline • CodePipeline
- Getting started
- Pipelines
- Pipeline**
- History
- Settings
- Settings

The main area displays the pipeline stages:

- Source**: AWS CodeCommit
- Image**: Amazon ECR
- + Add action**

Edit: Deploy

- Deploy**: Amazon ECS (Blue/Green)
- Configure automatic rollback on stage failure
- + Add stage**

+ Add action group

+ Add stage

Screenshot of the AWS CodePipeline console showing the pipeline configuration interface.

The left sidebar shows the navigation menu:

- Developer Tools
- CodePipeline**
- Source • CodeCommit
- Artifacts • CodeArtifact
- Build • CodeBuild
- Deploy • CodeDeploy
- Pipeline • CodePipeline
- Getting started
- Pipelines
- Pipeline**
- History
- Settings
- Settings

The main area displays the pipeline stages:

- Source**: AWS CodeCommit
- Image**: Amazon ECR
- + Add action**

Edit: Deploy

- Deploy**: Amazon ECS (Blue/Green)
- + Add action**
- + Add action group**
- Configure automatic rollback on stage failure
- + Add stage**

+ Add action group

+ Add stage

Screenshot of the AWS CodePipeline console showing the edit action configuration interface.

The left sidebar shows the navigation menu:

- Developer Tools
- CodePipeline**
- Source • CodeCommit
- Artifacts • CodeArtifact
- Build • CodeBuild
- Deploy • CodeDeploy
- Pipeline • CodePipeline
- Getting started
- Pipelines
- Pipeline**
- History
- Settings
- Settings

The main area displays the edit action configuration:

Edit action

Action name: Deploy

Action provider: Amazon ECS (Blue/Green)

Region: US East (N. Virginia)

Input artifacts:

- SourceArtifact: image-employee
- Add
- No more than 100 characters

AWS CodeDeploy application name: microservices

AWS CodeDeploy deployment group

AWS CodeDeploy deployment group
Choose one of your existing deployment groups, or create a new one in AWS CodeDeploy.

X

Amazon ECS task definition
Choose the input artifact where your Amazon ECS task definition file is stored. If other than the default file path, specify the path and filename of your task definition file.

SourceArtifact taskdef-employee.json

The default path is taskdef.json.

AWS CodeDeploy AppSpec file
Choose the input artifact where your AWS CodeDeploy AppSpec file is stored. If other than the default file path, specify the path and filename of your AppSpec file.

SourceArtifact appspec-employee.yaml

Dynamically update task definition image - *optional*
You can provide an input artifact and a placeholder name for the container definition image that will be used to dynamically update a task definition. You can specify multiple input artifacts and placeholders.

Input artifact with image details

Placeholder text in the task definition
 Remove

Add

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](#)

Cancel Done

aws | Services | Search [Alt+S] | vocabs/user5223197=Gupta_Nancy @ 0824-5190-9674

Developer Tools > CodePipeline > Pipelines > update-employee-microservice > Edit update-employee-microservice

Delete Cancel Save

CodePipeline

- ▶ Source • CodeCommit
- ▶ Artifacts • CodeArtifact
- ▶ Build • CodeBuild
- ▶ Deploy • CodeDeploy
- ▶ Pipeline • CodePipeline
 - Getting started
 - Pipelines
 - Pipeline**
 - History
 - Settings
 - ▶ Settings
- Q Go to resource
- Feedback

Edit: update-employee-microservice

Edit: Pipeline properties

Pipeline type
V2

Execution mode
QUEUED

Edit: Variables

Pipeline type V2 required

Name	Default value	Description
No variables		
No variables defined at the pipeline level in this pipeline.		

aws | Services | Search [Alt+S] | vocabs/user5223197=Gupta_Nancy @ 0824-5190-9674

Developer Tools > CodePipeline > Pipelines > update-employee-microservice > Edit update-employee-microservice

Delete Cancel Save

CodePipeline

- ▶ Source • CodeCommit
- ▶ Artifacts • CodeArtifact
- ▶ Build • CodeBuild
- ▶ Deploy • CodeDeploy
- ▶ Pipeline • CodePipeline
 - Getting started
 - Pipelines
 - Pipeline**
 - History
 - Settings
 - ▶ Settings
- Q Go to resource
- Feedback

Save pipeline changes

Saving your changes cannot be undone. If the pipeline is running when you save your changes, that execution will not complete.

Source action changed
We will update the following resources to detect changes for your updated pipeline.

Change type	Details
Add	Pipeline update-employee-microservice as a target to Amazon CloudWatch Events rule - codepipeline

No resource updates needed for this source action change

Cancel Save

Edit variables

No variables

No variables defined at the pipeline level in this pipeline.

The screenshot shows the AWS CodePipeline console. A success message at the top states "Pipeline was saved successfully." The pipeline name is "update-employee-microservice". The pipeline type is V2 and the execution mode is QUEUED. The Source stage is listed as Succeeded, with a Pipeline execution ID of 446003bb-c4af-4b9e-9f09-30315df9dd5. The Deploy stage is shown as Failed. The pipeline has a history of one execution, which succeeded 4 minutes ago. The pipeline status is overall SUCCEEDED.

Task 8.5: Test the CI/CD pipeline for the *employee* microservice

In this task, you will test the pipeline that you just defined for the *employee* microservice.

Important: If you are repeating this task, confirm that all target groups are still associated with the Application Load Balancer. The [Reassociate Target Groups with Load Balancer](#) section of the appendix provides detail if needed.

1. Launch a deployment of the *employee* microservice on Amazon ECS on Fargate.

The screenshot shows the AWS CodePipeline console. The pipeline run is now Failed. The Deploy stage is highlighted in red and labeled "Failed". The pipeline status is overall FAILED.

- o Use the *release change* feature to force a test of the pipeline.

The screenshots illustrate the workflow for releasing changes and deploying them using AWS CodePipeline.

Screenshot 1: Release change

This screenshot shows the "Release change" dialog box. It contains a message asking if you want to release the most recent change. Below this, it shows a "Source" section with "AWS CodeCommit" and a "Succeeded" status. A "View details" button is available. At the bottom right are "Cancel" and "Release" buttons. A note at the bottom states: "6205e182 Source: pushing 4 files created in deployment directory".

Screenshot 2: Pipeline execution success

This screenshot shows the pipeline execution status as "Success". It displays the "Source" step (AWS CodeCommit) and the "Image" step (Amazon ECR). Both steps show a "Succeeded" status with a timestamp of "Just now". A "View details" button is present for each. A note at the bottom states: "6205e182 Source: pushing 4 files created in deployment directory Image: sha256:6005a8ab686c".

Screenshot 3: Deployment in progress

This screenshot shows the "Deploy" step in progress. It lists "Amazon ECS (Blue/Green)" as the provider. A status message indicates "In progress - Just now". A "View details" button is available. A note at the bottom states: "6205e182 Source: pushing 4 files created in deployment directory Image: sha256:6005a8ab686c".

- Follow the progress in CodeDeploy. Within a few minutes, if everything was configured correctly, all of the *Deployment lifecycle events* should succeed.

SOURCE: AWS CodeCommit IMAGE: Amazon ECS

Succeeded - just now Succeeded - just now
09/28/2024 09:58:07 UTC 09:58:07 UTC

Action execution details

Action name: Deploy Status: In progress

Summary Input

Status: In progress Last updated: Just now

Action execution ID: f681eb54-091c-46fd-9946-b1967d86ef34

Message: Deployment d-NMB93F9U5 created

Execution details: View in CodeDeployToECS

Done

Developer Tools > CodeDeploy > Deployments > d-NMB93F9U5

d-NMB93F9U5

Deployment status

- Step 1: Deploying replacement task set (In progress, 50%)
- Step 2: Test traffic route setup (Not started, 0%)
- Step 3: Rerouting production traffic to replacement task set (Not started, 0%)
- Step 4: Wait 5 minutes 0 seconds (Not started, 0%)
- Step 5: Terminate original task set (Not started, 0%)

Traffic shifting progress

Original	Replacement
100%	0%
Original task set serving traffic	Replacement task set not serving traffic

Developer Tools > CodeDeploy > Deployments > d-NMB93F9U5

d-NMB93F9U5

Deployment status

- Step 1: Deploying replacement task set (Completed, Succeeded, 100%)
- Step 2: Test traffic route setup (Completed, Succeeded, 100%)
- Step 3: Rerouting production traffic to replacement task set (100% traffic shifted, Succeeded)
- Step 4: Wait (Wait completed, Succeeded)
- Step 5: Terminate original task set (Completed, Succeeded)

Traffic shifting progress

Original	Replacement
0%	100%
Original task set not serving traffic	Replacement task set serving traffic

Deployment details

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

- Deployment details:**
 - Application: microservices
 - Deployment ID: d-NMB93F9US
 - Status: Succeeded
 - Deployment configuration: CodeDeployDefault.ECSAllAtOnce
 - Deployment group: microservices-employee
 - Initiated by: User action
 - Deployment description: -
- Revision details:**
 - Revision location: f78a0e5f055cc7870ec47b300e0b3a9901ead809da4
 - Revision created: 7 minutes ago
 - Revision description: Application revision registered by Deployment ID: d-NMB93F9US
- Task set activity:**

Task set ID	Environment	Task set status	Traffic	Desired count	Running count	Pending count
ecs-svc/0996791440688041300	Replacement	PRIMARY	100%	1	1	0

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

- Task set activity:**

Task set ID	Environment	Task set status	Traffic	Desired count	Running count	Pending count
ecs-svc/0996791440688041300	Replacement	PRIMARY	100%	1	1	0
ecs-svc/016457500502814039	Original	ACTIVE	0	1	0	0
- Deployment lifecycle events:**

Event	Duration	Status	Start time	End time
BeforeInstall	less than one second	Succeeded	Apr 29, 2024 4:25 AM (UTC-4:00)	Apr 29, 2024 4:25 AM (UTC-4:00)
Install	2 minutes 3 seconds	Succeeded	Apr 29, 2024 4:25 AM (UTC-4:00)	Apr 29, 2024 4:27 AM (UTC-4:00)
AfterInstall	less than one second	Succeeded	Apr 29, 2024 4:27 AM (UTC-4:00)	Apr 29, 2024 4:27 AM (UTC-4:00)
AllowTestTraffic	less than one second	Succeeded	Apr 29, 2024 4:27 AM (UTC-4:00)	Apr 29, 2024 4:27 AM (UTC-4:00)
AfterAllowTestTraffic	less than one second	Succeeded	Apr 29, 2024 4:27 AM (UTC-4:00)	Apr 29, 2024 4:27 AM (UTC-4:00)
BeforeAllowTraffic	less than one second	Succeeded	Apr 29, 2024 4:27 AM (UTC-4:00)	Apr 29, 2024 4:27 AM (UTC-4:00)
AllowTraffic	less than one second	Succeeded	Apr 29, 2024 4:27 AM (UTC-4:00)	Apr 29, 2024 4:27 AM (UTC-4:00)
AfterAllowTraffic	less than one second	Succeeded	Apr 29, 2024 4:27 AM (UTC-4:00)	Apr 29, 2024 4:27 AM (UTC-4:00)

2. Load the *employee* microservice in a browser tab.

- In the browser tab where the customer microservice is running, choose **Administrator link**.

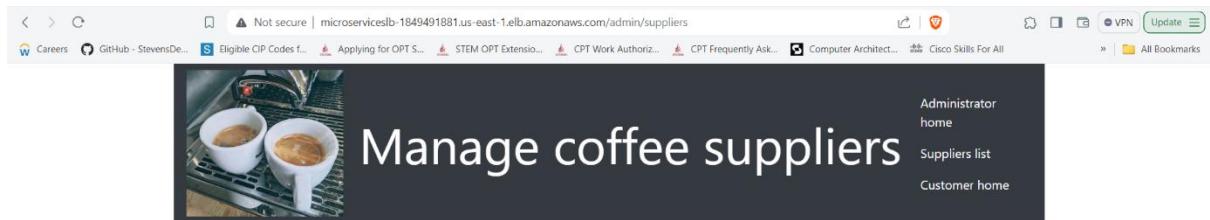
You are directed to a page in the format <http://microserviceslb-UNIQUE-NUMBER.us-east-1.elb.amazonaws.com/admin/suppliers>.

The employee microservice loads. If it doesn't load, try adding :8080 just after amazonaws.com in the URL.



All suppliers

Name	Address	City	State	Email	Phone
Nancy_Supplier2	5 castle point terrace	Hoboken	New Jersey	ngupta20@stevens.edu	987456321



All suppliers

Name	Address	City	State	Email	Phone
Nancy_Supplier2	5 castle point terrace	Hoboken	New Jersey	ngupta20@stevens.edu	987456321

[Add a new supplier](#)

- Choose **List of suppliers** or **Suppliers list**.



All suppliers

Name	Address	City	State	Email	Phone
Nancy_Supplier2	5 castle point terrace	Hoboken	New Jersey	ngupta20@stevens.edu	987456321

The suppliers page should load. This version of the page should *not* have the edit or add supplier buttons. All links in the café web application should now work because you have now deployed both microservices.

3. Observe the running tasks in the Amazon ECS console.

The **Deployments and tasks** status will change as the blue/green deployment advances through its lifecycle events.

The screenshots illustrate the progression of a blue/green deployment across three stages:

- Stage 1 (Initial State):** The "Clusters" page shows a single cluster named "microservices-serverlesscluster". It has 2 services, 0 pending tasks, 0 running tasks, and 0 EC2 instances. CloudWatch monitoring is set to "Default".
- Stage 2 (Deployment Progress):** The "Cluster overview" page for "microservices-serverlesscluster" shows the cluster status as "Active". Under "Services", one service is listed as "Draining" with 2 active tasks. Under "Tasks", there is 1 pending task and 2 running tasks. The "Services" table shows 2 services: "employee-microservice" and "customer-microservice", both active and running 1 task each.
- Stage 3 (Deployment Complete):** The "Cluster overview" page shows the cluster status as "Active". Under "Services", the "Draining" service now has 0 active tasks. Under "Tasks", there are 2 pending tasks and 2 running tasks. The "Services" table shows 2 services: "employee-microservice" and "customer-microservice", both active and running 1 task each.