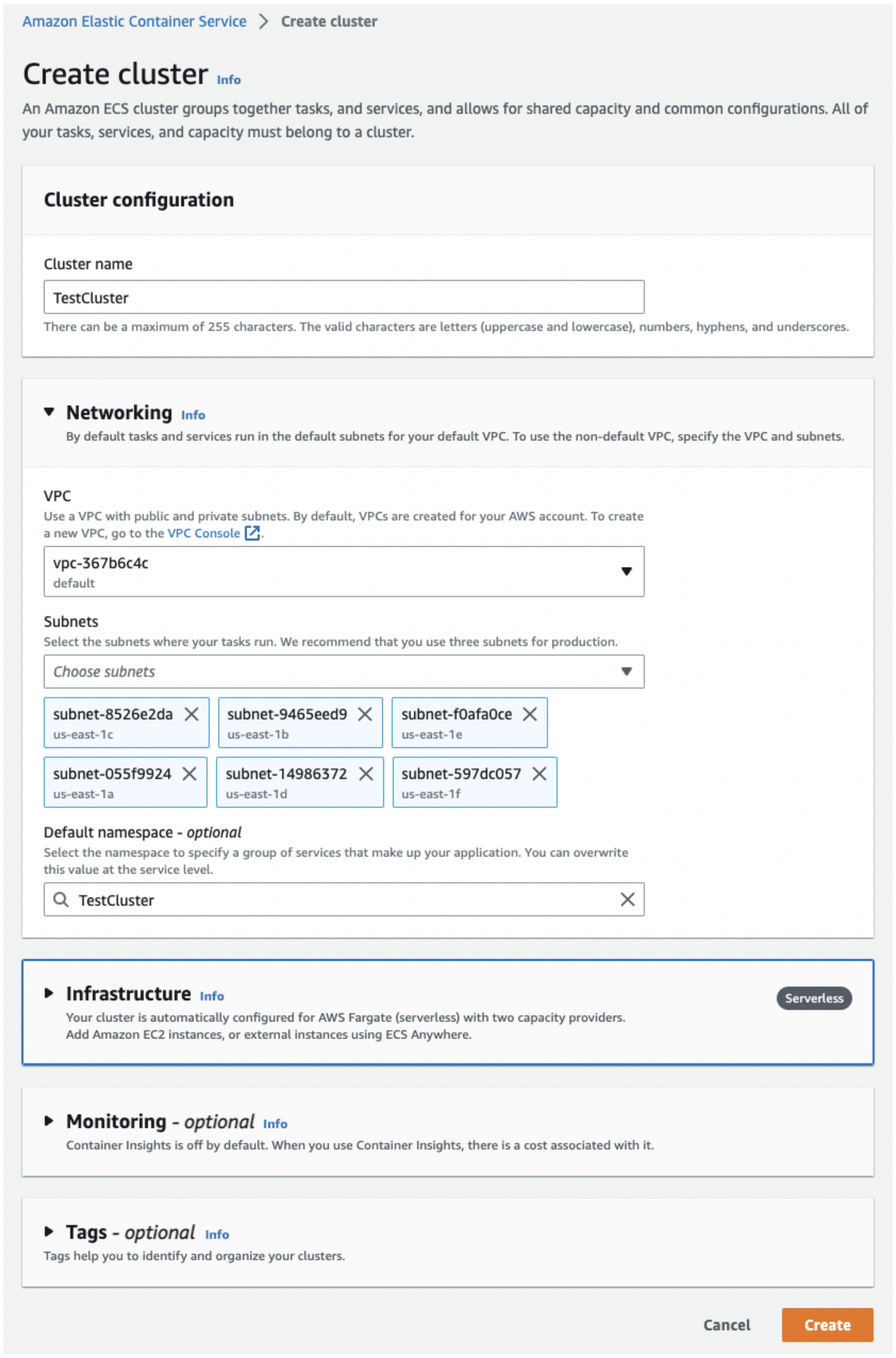
Auto-deploy your container to Fargate

Pre-requisites:

- Should have completed the steps mentioned in **Setup CI/CD for a GitHub repository with Codebuild + Codepipeline** tutorial:
    - Have a ECR repository
    - Completed the AWS Code build setup

Creating a ECS cluster for the Fargate:

- Open the console at https://console.aws.amazon.com/ecs/v2.
- From the navigation bar, select the Region to use.
- In the navigation pane, choose **Clusters**.
- On the **Clusters** page, choose **Create cluster**.
- Under **Cluster configuration**, for **Cluster name**, enter a unique name.
    - The name can contain up to 255 letters (uppercase and lowercase), numbers, and hyphens.
- Choose **Create**.


Amazon Elastic Container Service > Create cluster

Create cluster Info

An Amazon ECS cluster groups together tasks, and services, and allows for shared capacity and common configurations. All of your tasks, services, and capacity must belong to a cluster.

Cluster configuration

Cluster name: TestCluster

There can be a maximum of 255 characters. The valid characters are letters (uppercase and lowercase), numbers, hyphens, and underscores.

Networking Info

By default tasks and services run in the default subnets for your default VPC. To use the non-default VPC, specify the VPC and subnets.

VPC

Use a VPC with public and private subnets. By default, VPCs are created for your AWS account. To create a new VPC, go to the VPC Console.

vpc-367b6c4c
default

Subnets

Select the subnets where your tasks run. We recommend that you use three subnets for production.

Choose subnets

subnet-8526e2da us-east-1c
subnet-9465eed9 us-east-1b
subnet-f0afa0ce us-east-1e
subnet-055f9924 us-east-1a
subnet-14986372 us-east-1d
subnet-597dc057 us-east-1f

Default namespace - *optional*

Select the namespace to specify a group of services that make up your application. You can overwrite this value at the service level.

TestCluster

Infrastructure Info — Serverless

Your cluster is automatically configured for AWS Fargate (serverless) with two capacity providers. Add Amazon EC2 instances, or external instances using ECS Anywhere.

Monitoring - *optional* Info

Container Insights is off by default. When you use Container Insights, there is a cost associated with it.

Tags - *optional* Info

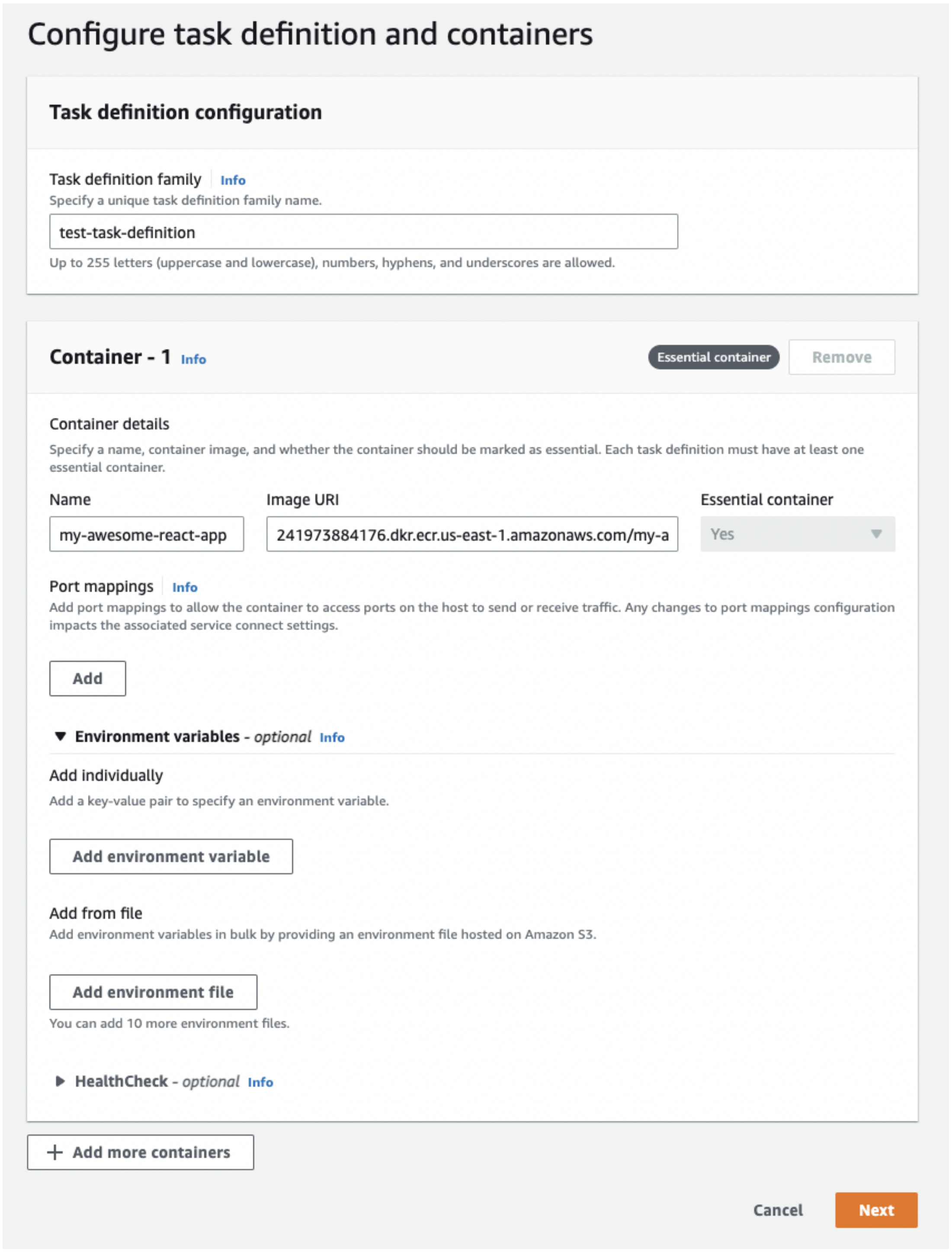
Tags help you to identify and organize your clusters.

Cancel Create

Create a Task Definition:

- Open the console at https://console.aws.amazon.com/ecs/v2.
- In the navigation pane, choose **Task definitions**

- Choose **Create new task definition**, **Create new task definition**.
- For **Task definition family**, specify a unique name for the task definition.
- For each container to define in your task definition, complete the following steps.
  - For **Name**, enter a name for the container.
    - NOTE: Note this name to be used in later steps.
  - For **Image URI**, enter the image to use to start a container. This will be the URI for your ECR repository.
  - Under **Port mappings**, for **Container port** (eg: 8080)and **Protocol** (eg: HTTP), choose the port mapping to use for the container


Configure task definition and containers

Task definition configuration

Task definition family Info
Specify a unique task definition family name.
test-task-definition
Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.

Container - 1 Info
Essential container
Remove

Container details
Specify a name, container image, and whether the container should be marked as essential. Each task definition must have at least one essential container.

Name	Image URI	Essential container
my-awesome-react-app	241973884176.dkr.ecr.us-east-1.amazonaws.com/my-a	Yes

Port mappings Info
Add port mappings to allow the container to access ports on the host to send or receive traffic. Any changes to port mappings configuration impacts the associated service connect settings.
Add

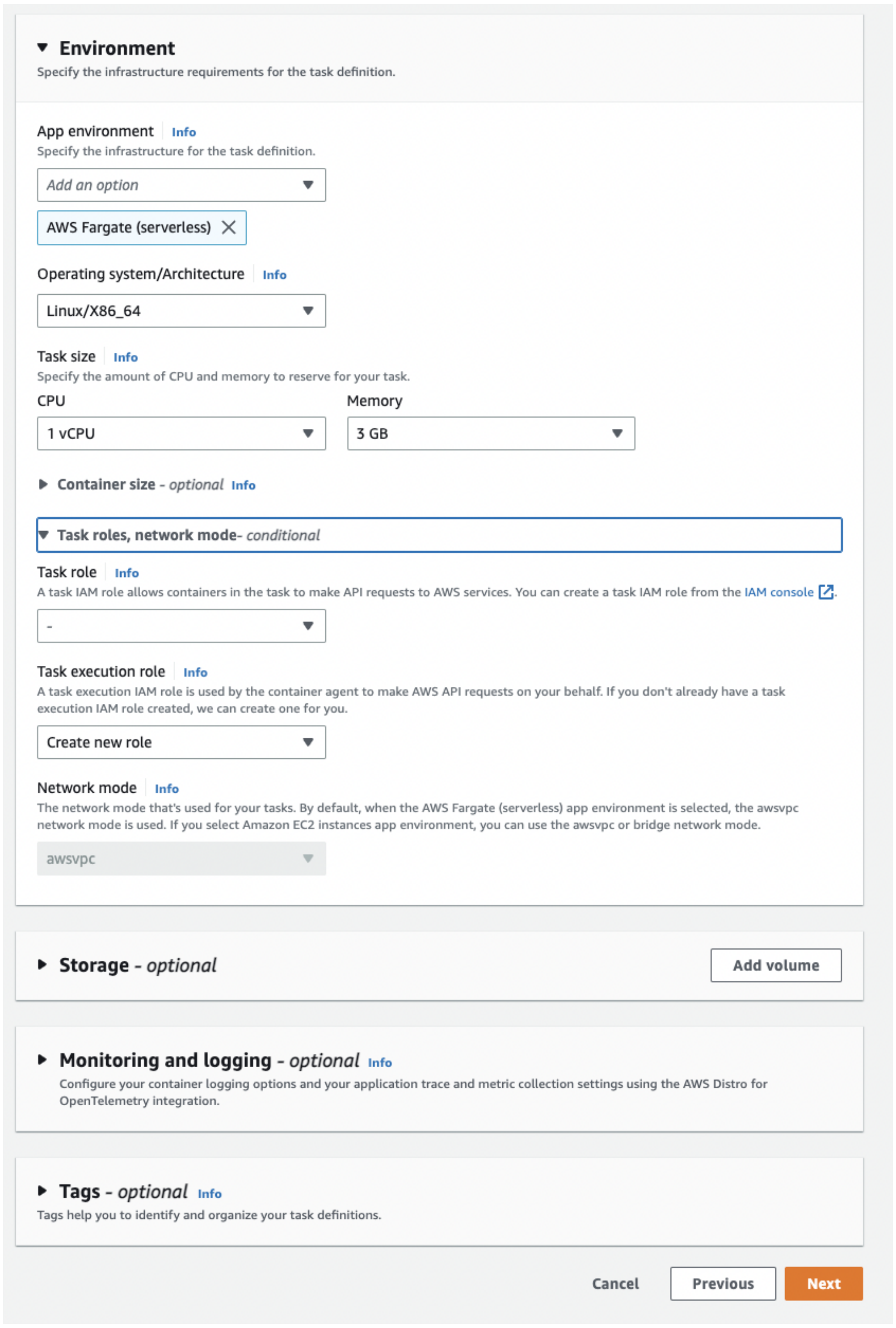
▼ Environment variables - *optional* Info

Add individually
Add a key-value pair to specify an environment variable.
Add environment variable

Add from file
Add environment variables in bulk by providing an environment file hosted on Amazon S3.
Add environment file
You can add 10 more environment files.

▶ HealthCheck - *optional* Info

  - Choose **Next**
- For **App environment**, choose **AWS Fargate (serverless)**



Environment
Specify the infrastructure requirements for the task definition.

App environment Info
Specify the infrastructure for the task definition.
Add an option
AWS Fargate (serverless)

Operating system/Architecture Info
Linux/X86_64

Task size Info
Specify the amount of CPU and memory to reserve for your task.
CPU: 1 vCPU
Memory: 3 GB

Container size - *optional* Info

Task roles, network mode- *conditional*

Task role Info
A task IAM role allows containers in the task to make API requests to AWS services. You can create a task IAM role from the IAM console.
-

Task execution role Info
A task execution IAM role is used by the container agent to make AWS API requests on your behalf. If you don't already have a task execution IAM role created, we can create one for you.
Create new role

Network mode Info
The network mode that's used for your tasks. By default, when the AWS Fargate (serverless) app environment is selected, the awsvpc network mode is used. If you select Amazon EC2 instances app environment, you can use the awsvpc or bridge network mode.
awsvpc

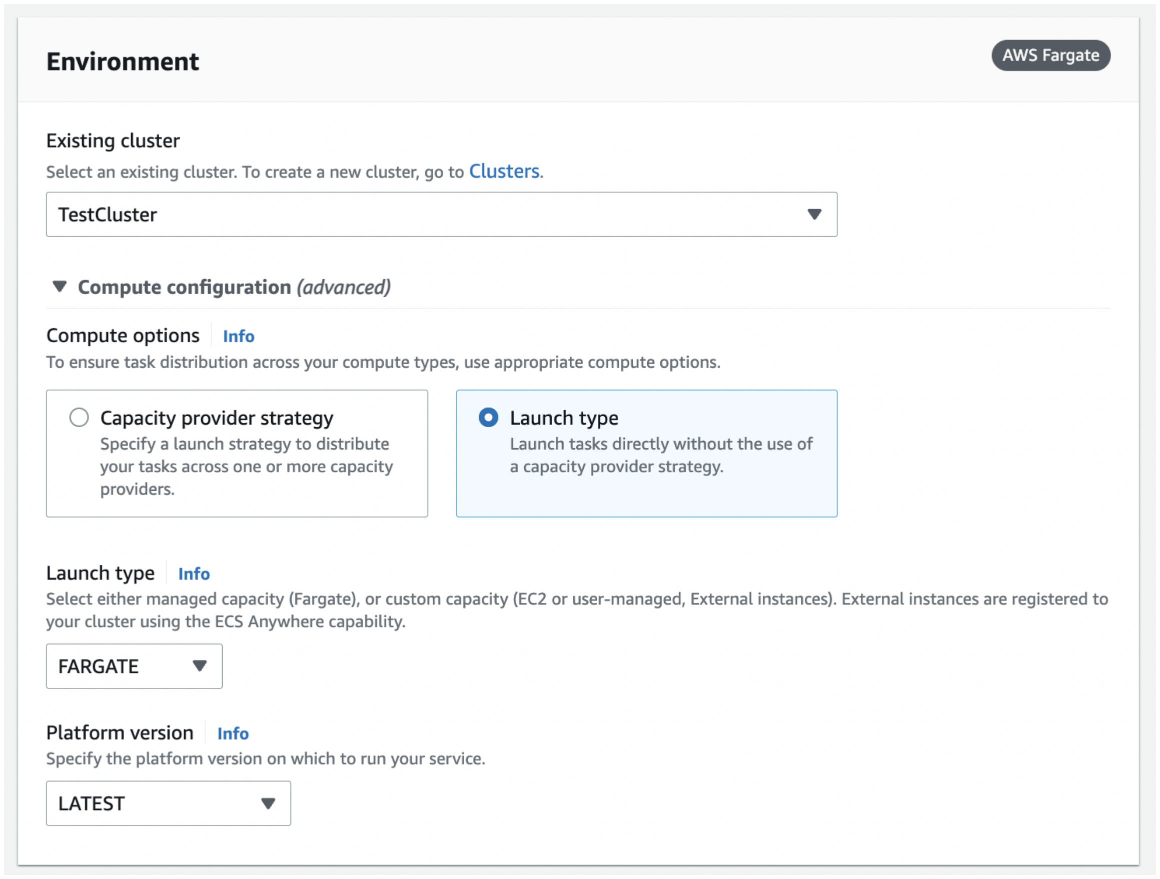
Storage - *optional*

Monitoring and logging - *optional* Info
Configure your container logging options and your application trace and metric collection settings using the AWS Distro for OpenTelemetry integration.

Tags - *optional* Info
Tags help you to identify and organize your task definitions.

- Choose **Next** to review the task definition.
- On the **Review and create** page, choose **Create** to register the task definition.
- In the navigation pane, choose **Task Definitions** and choose the task definition you just created.
- Choose **Deploy** → **Create Service**

- Choose the Cluster you created for **Existing Cluster**



Environment

AWS Fargate

Existing cluster

Select an existing cluster. To create a new cluster, go to Clusters.

TestCluster

▼ Compute configuration *(advanced)*

Compute options Info

To ensure task distribution across your compute types, use appropriate compute options.

○ Capacity provider strategy
Specify a launch strategy to distribute your tasks across one or more capacity providers.

● Launch type
Launch tasks directly without the use of a capacity provider strategy.

Launch type Info

Select either managed capacity (Fargate), or custom capacity (EC2 or user-managed, External instances). External instances are registered to your cluster using the ECS Anywhere capability.

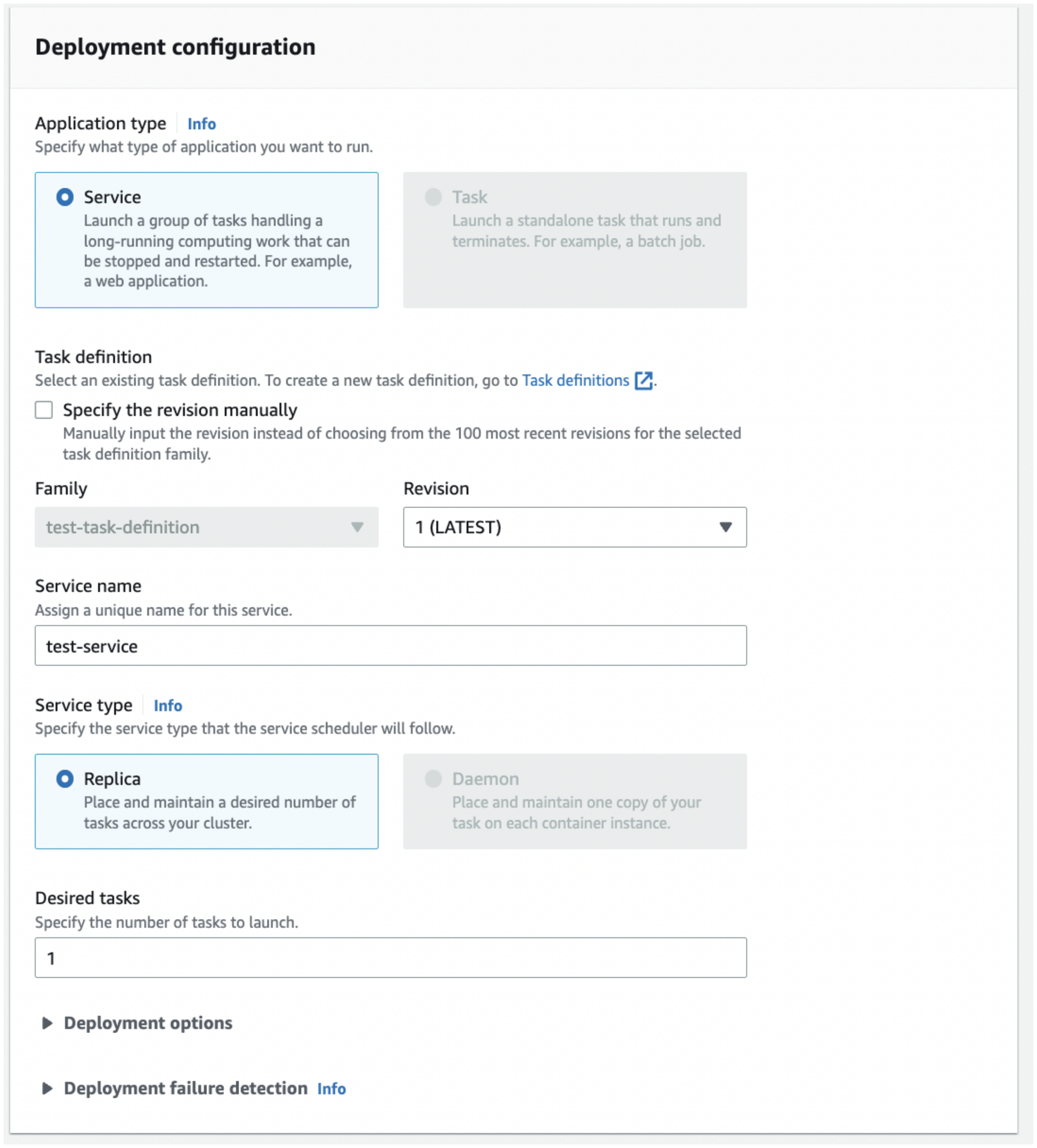
FARGATE

Platform version Info

Specify the platform version on which to run your service.

LATEST

- For **Compute options,** choose **Launch Type**
- **Service name**: Enter a unique name for your service (eg: test-service)


Deployment configuration

Application type Info
Specify what type of application you want to run.

[✓ Service] Launch a group of tasks handling a long-running computing work that can be stopped and restarted. For example, a web application.

Task: Launch a standalone task that runs and terminates. For example, a batch job.

Task definition
Select an existing task definition. To create a new task definition, go to Task definitions.

☐ Specify the revision manually
Manually input the revision instead of choosing from the 100 most recent revisions for the selected task definition family.

Family	test-task-definition
Revision	1 (LATEST)

Service name
Assign a unique name for this service.
test-service

Service type Info
Specify the service type that the service scheduler will follow.

[✓ Replica] Place and maintain a desired number of tasks across your cluster.

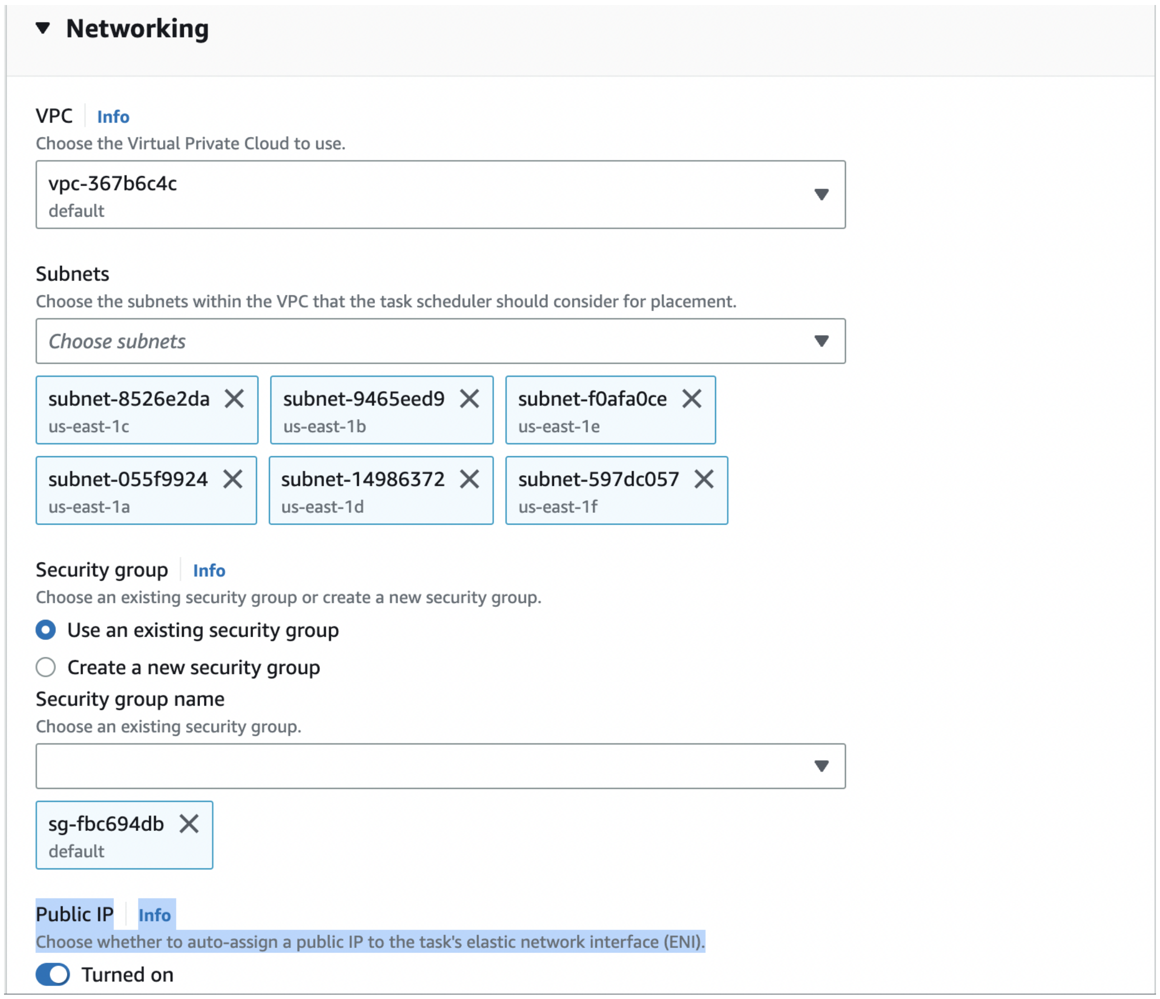
Daemon: Place and maintain one copy of your task on each container instance.

Desired tasks
Specify the number of tasks to launch.
1

▶ Deployment options

▶ Deployment failure detection Info

- IMPORTANT: Open the **Networking** section, turn on **Public IP** setting


▼ Networking

VPC Info
Choose the Virtual Private Cloud to use.
vpc-367b6c4c
default

Subnets
Choose the subnets within the VPC that the task scheduler should consider for placement.
Choose subnets

subnet-8526e2da us-east-1c
subnet-9465eed9 us-east-1b
subnet-f0afa0ce us-east-1e
subnet-055f9924 us-east-1a
subnet-14986372 us-east-1d
subnet-597dc057 us-east-1f

Security group Info
Choose an existing security group or create a new security group.
[✓ Use an existing security group]
Create a new security group
Security group name
Choose an existing security group.
sg-fbc694db
default

Public IP Info
Choose whether to auto-assign a public IP to the task's elastic network interface (ENI).
Turned on

- Choose **Create**

Create/Update the buildspec file:

- Create a file named "buildspec.yml" in the root of your GitHub repository if not already present
- Paste the following code into the buildspec file, which builds the Docker image and pushes it to the ECR repository:

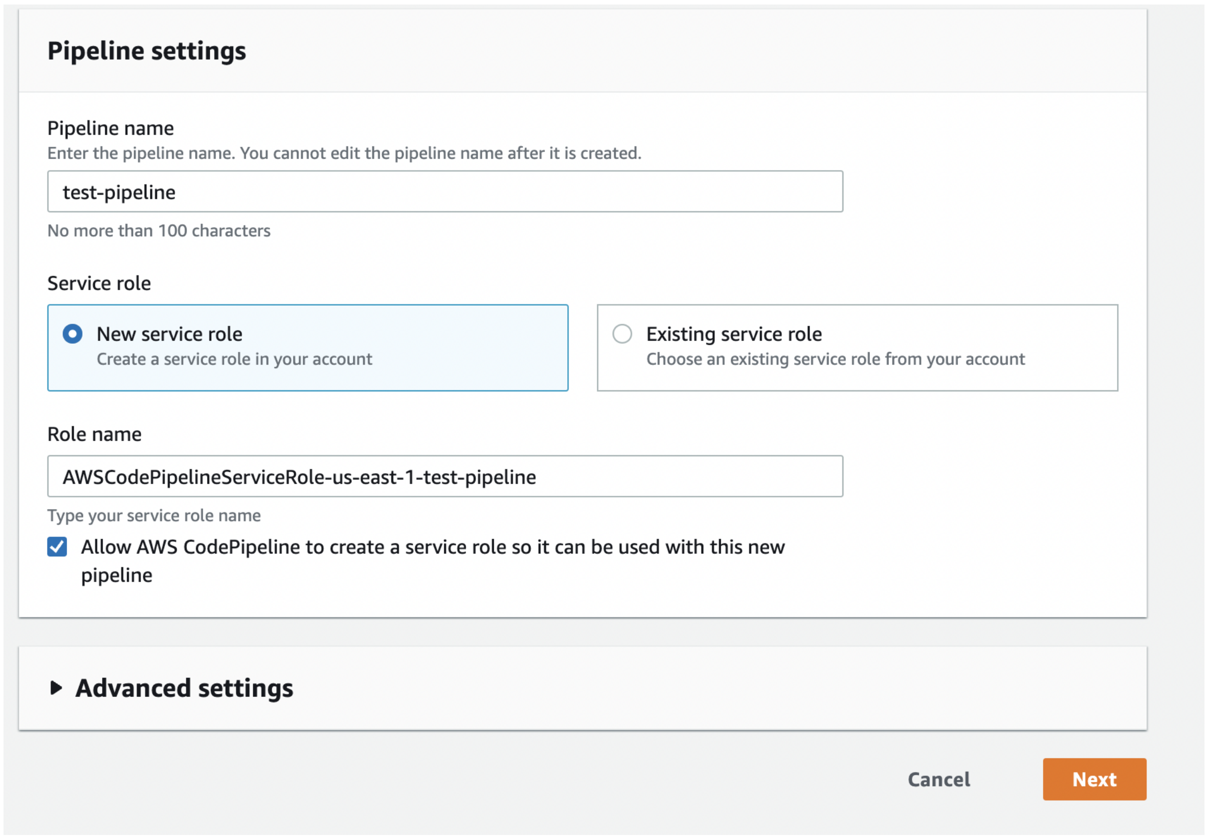
```
version: 0.2
phases:
  install:
    commands:
      - echo install step...
  pre_build:
    commands:
      - echo logging in to AWS ECR...
      - $(aws ecr get-login --no-include-email --region us-east-1)
  build:
    commands:
      - echo build Docker image on `date`
      - cd src
      - docker build -t <image name>:latest .
      - docker tag <image name>:latest <ECR repository URI>:latest
  post_build:
    commands:
      - echo build Docker image complete `date`
      - echo push latest Docker images to ECR...
      - docker push <ECR repository URI>:latest
      - echo Writing image definitions file...
      - printf '[{"name":"<container name>","imageUri":"<ECR repository URI>:latest"}]' > imagedefinitions.json
artifacts:
  files: imagedefinitions.json
```

- Replace <image name> and <ECR repository URI> with the actual values.
- Replace <container name> with the name you specified in task definition in previous step
- Run `git add buildspec.yml` , `git commit` and `git push` to commit the file

💡 NOTE: The updated `buildspec.yml` writes a file called `imagedefinitions.json` in the build root that has your Amazon ECS service's container name and the image and tag. The deployment stage of your CD pipeline uses this information to create a new revision of your service's task definition, and then it updates the service to use the new task definition.
The `imagedefinitions.json` file is required for the ECS job worker.

Create a AWS CodePipeline pipeline:

- Click here to open the AWS CodePipeline console.
- On the Welcome page, click **Create pipeline**. If this is your first time using AWS CodePipeline, an introductory page appears instead of Welcome. Click **Get Started**.
- Enter the name for your pipeline, Choose **New service role**, and in **Role Name**, enter the name for your new service role. Click **Next**


Pipeline settings

Pipeline name

Enter the pipeline name. You cannot edit the pipeline name after it is created.

test-pipeline

No more than 100 characters

Service role

● New service role
Create a service role in your account

○ Existing service role
Choose an existing service role from your account

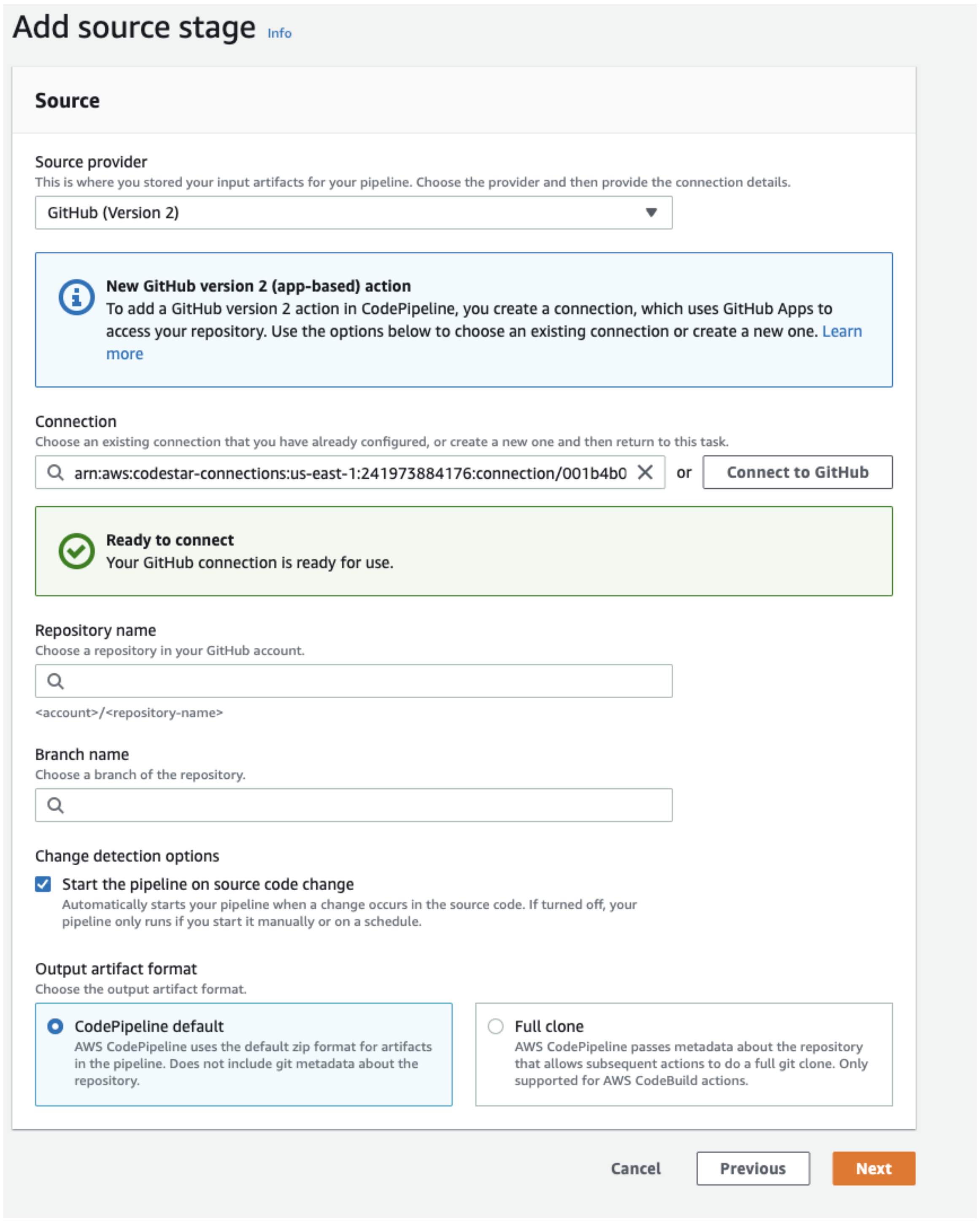
Role name

AWSCodePipelineServiceRole-us-east-1-test-pipeline

Type your service role name

☑ Allow AWS CodePipeline to create a service role so it can be used with this new pipeline

- On the **Add source stage** page, for the **Source provider,** choose **GitHub (v2) ,** Click on **Connect To Github** and follow the instructions
- Set the **repository** and **branch name** and set **Output artifact format** to **CodePipeline default**



Add source stage Info

Source

Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.
GitHub (Version 2)

New GitHub version 2 (app-based) action
To add a GitHub version 2 action in CodePipeline, you create a connection, which uses GitHub Apps to access your repository. Use the options below to choose an existing connection or create a new one. Learn more

Connection
Choose an existing connection that you have already configured, or create a new one and then return to this task.
arn:aws:codestar-connections:us-east-1:241973884176:connection/001b4b0 or Connect to GitHub

Ready to connect
Your GitHub connection is ready for use.

Repository name
Choose a repository in your GitHub account.
<account>/<repository-name>

Branch name
Choose a branch of the repository.

Change detection options
☑ Start the pipeline on source code change
Automatically starts your pipeline when a change occurs in the source code. If turned off, your pipeline only runs if you start it manually or on a schedule.

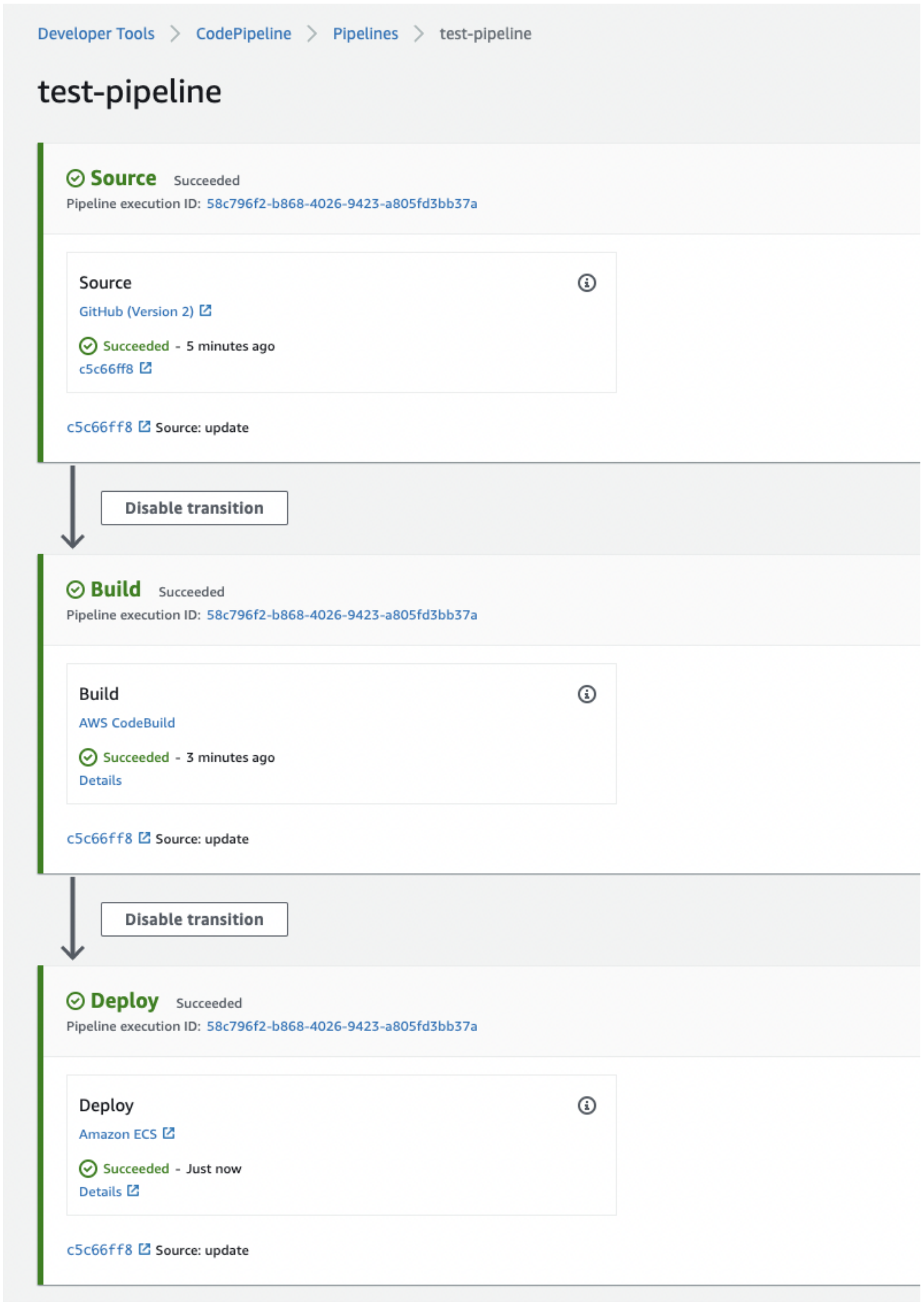
Output artifact format
Choose the output artifact format.
● CodePipeline default
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include git metadata about the repository.
○ Full clone
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full git clone. Only supported for AWS CodeBuild actions.

- For **the build stage**, select **AWS CodeBuild** and choose the CodeBuild project created during last exercise
- For the **Add deploy stage:**
  - For **Deploy action provider** choose Amazon ECS
  - For **Cluster name,** Choose the cluster you created in previous step

    - For **Service name,** Choose the service you created in create task definition step
- Go to the bottom of the review page and click **Create Pipeline**

Test your Pipeline:

- Make a code change to your configured source repository, commit, and push the change.
- Open the CodePipeline console at https://console.aws.amazon.com/codepipeline/.
- Choose your pipeline from the list.
- Watch the pipeline progress through its stages. Your pipeline should complete and your Amazon ECS service runs the Docker image that was created from your code change.


Developer Tools > CodePipeline > Pipelines > test-pipeline

test-pipeline

Source Succeeded
Pipeline execution ID: 58c796f2-b868-4026-9423-a805fd3bb37a

Source
GitHub (Version 2)
Succeeded - 5 minutes ago
c5c66ff8

c5c66ff8 Source: update

Disable transition

Build Succeeded
Pipeline execution ID: 58c796f2-b868-4026-9423-a805fd3bb37a

Build
AWS CodeBuild
Succeeded - 3 minutes ago
Details

c5c66ff8 Source: update

Disable transition

Deploy Succeeded
Pipeline execution ID: 58c796f2-b868-4026-9423-a805fd3bb37a

Deploy
Amazon ECS
Succeeded - Just now
Details

c5c66ff8 Source: update