

AWS CodePipeline

Instruction
By @Chihiro-001

Scenario

- A development team manages their application in their GitHub repository. Whenever they release their application, they do manual build and deployment checks. The release frequency is expected to increase, and so the team wants to reduce manual operations and tedious tasks.
- The team wants to automate the release pipelines for fast and reliable application and infrastructure updates. They need build, test, and deploy phases for every commit pushed.
- The team wants to use a container service as a deployment destination for consistency across environments on the user side.

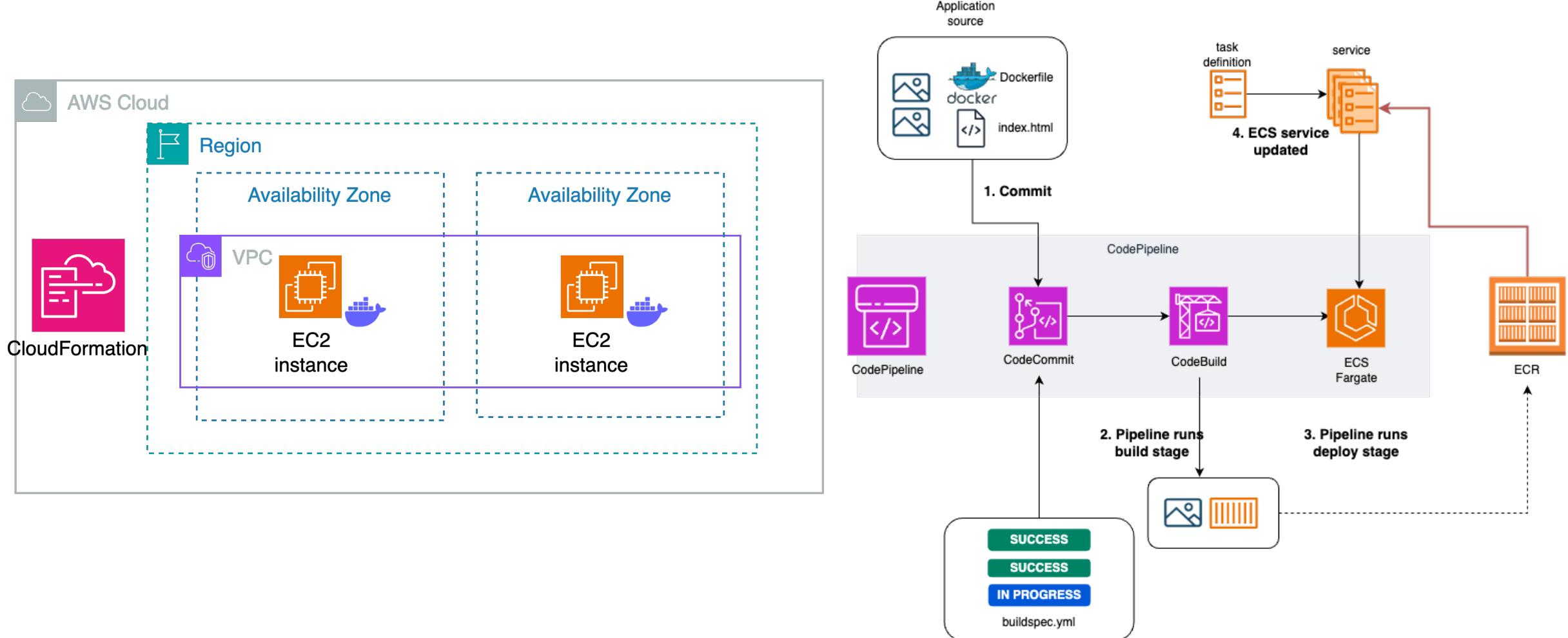
2. Design

- Specification (Solution) for Requirements:

No	Requirement	Specification (Solution)
1	Automatically build an application's environment that fulfills the following conditions: ① set up a network infrastructure ② set EC2 instances with docker installed in each public subnet, using nginx as the web server.	Use CloudFormation to automatically build the environment. ① create a YAML file to set up the network infrastructure such as VPC, Security Group, etc ② use the same YAML file as the above one to set up EC2 instances.
2	Automatically build and deploy the application	Use CodePipeline with CodeBuild and CodeDeploy. (Use CodeCommit as a source repository) With a build process, use ECR. This is where their docker image is stored and managed
3	Use container service(s) for deployment destination. It should be serverless.	Use ECS Fargate

2. Design

Diagram: CodePipeline + ECR + ECS Fargate with CloudFormation



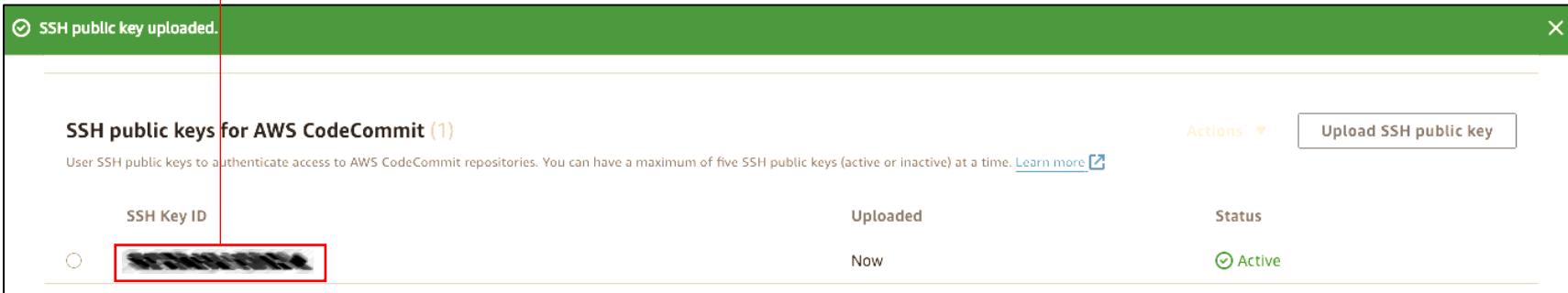
3. Implementation

Configure CodeCommit with SSH public key:

1. On your terminal, generate a SSH public key
2. On AWS Console, go to the IAM console, upload the SSH key to Security Credential under the AWS CodeCommit section
3. Add the SSH Key ID to the config file from your terminal

```
# vi .ssh/config  
  
Host git-codecommit.*.amazonaws.com  
User <YOUR_SSH_KEY_ID>  
IdentityFile ~/.ssh/codecommit
```

```
#run the following command to generate SSH key (This  
demo is done on Mac)  
cd .ssh  
ssh-keygen -t rsa -b 4096 #name the key 'codecommit'  
cat ~/.ssh/codecommit.pub  
vi config # add the configuration as left  
chmod 600 config
```



The screenshot shows the AWS IAM SSH Public Keys page. A green success message at the top says "SSH public key uploaded." Below it, a table lists the uploaded SSH key. The key has a status of "Active". A red box highlights the "SSH Key ID" column, which contains a long, obscured string of characters.

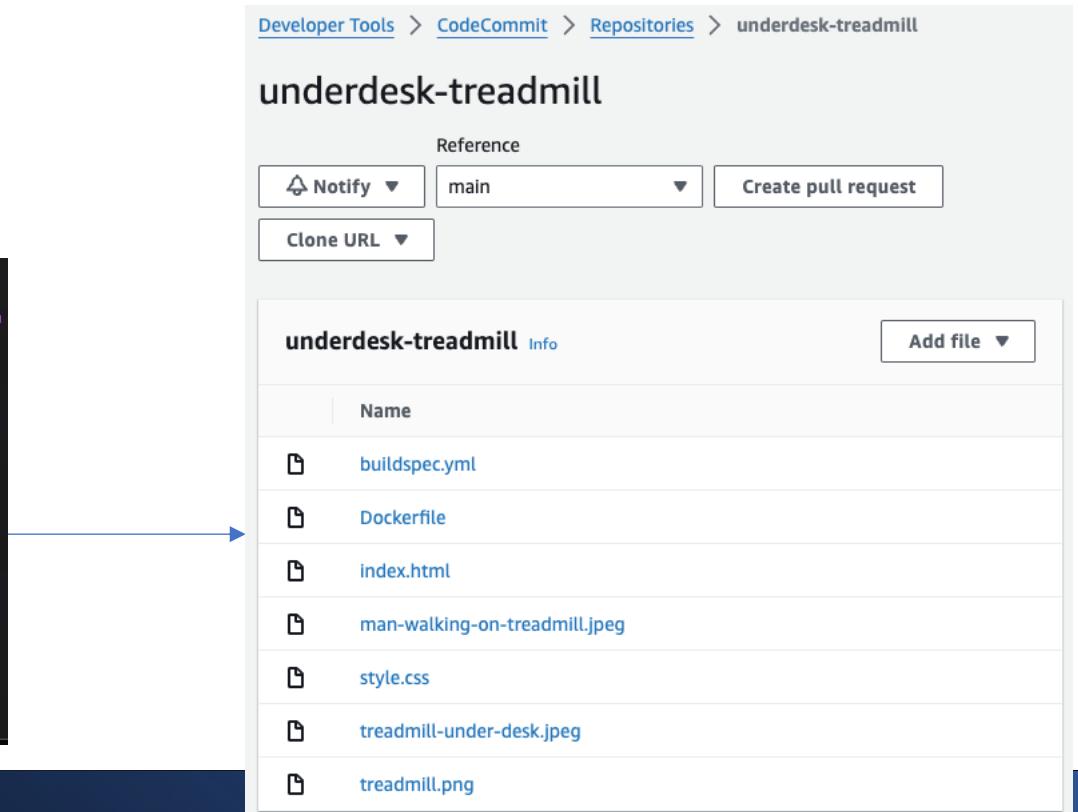
SSH Key ID	Uploaded	Status
[REDACTED]	Now	Active

3. Implementation

Setup the source code:

1. On the AWS CodeCommit console, create a CodeCommit repository
2. On your terminal, clone the repository and copy the necessary files (below) to CodeCommit
 1. Dockerfile
 2. index.html
 3. style.css
 4. *.jpeg
 5. treadmill.png
 6. buildspec.yml
3. Git push

```
chihiro@Chihiros-Air ~ % cd repos/underdesk-treadmill
chihiro@Chihiros-Air % git add -A
chihiro@Chihiros-Air % git commit -m 'first commit'
[main (root-commit) 7bc3284] first commit
 7 files changed, 100 insertions(+)
 create mode 100644 Dockerfile
 create mode 100644 buildspec.yml
 create mode 100644 index.html
 create mode 100644 man-walking-on-treadmill.jpeg
 create mode 100644 style.css
 create mode 100644 treadmill-under-desk.jpeg
 create mode 100644 treadmill.png
chihiro@Chihiros-Air % git push
```



3. Implementation

Setup Elastic Container Registry (ECR):

Configure the Elastic Container Registry to manage docker images.

1. Create a private repository

Amazon ECR > Private registry > Repositories > Create repository

Create repository

General settings

Visibility settings [Info](#)
Choose the visibility setting for the repository.

Private
Access is managed by IAM and repository policy permissions.

Public
Publicly visible and accessible for image pulls.

Repository name
Provide a concise name. A developer should be able to identify the repository contents by the name.
252520751240.dkr.ecr.us-east-1.amazonaws.com/ under-desk-treadmill
20 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

Tag immutability [Info](#)
Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.

Disabled

[Once a repository is created, the visibility setting of the repository can't be changed.](#)

Image scan settings and encryption settings are default setting. (both are set to be 'disabled')

Repositories (1)

C View push commands Delete Actions ▾ [Create repository](#)

Filter status

Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type
under-desk-treadmill	dkr.ecr.ap-northeast-1.amazonaws.com/under-desk-treadmill	May 02, 2024, 15:43:16 (UTC+09)	Disabled	Manual	AES-256

3. Implementation

Setup CodeBuild: (Continue 1/3)

Configure a CodeBuild Project to take the contents of the CodeCommit repository, build a docker image, and store it in the ECR.

The screenshot shows the AWS CodeBuild setup interface across three main sections:

- Source**:
 - Source 1 - Primary**:
 - Source provider: AWS CodeCommit
 - Repository: under-desk-treadmill
 - Reference type: Branch (selected)
 - Branch: main
 - Source version: refs/heads/main (3e5049f1 add images)
 - Additional configuration: Git clone depth, Git submodules
 - Add source button (marked with a blue box labeled 1)
- Environment**:
 - Provisioning model: On-demand (selected)
 - Environment image: Managed image (selected)
 - Compute: EC2 (selected)
 - Operating system: Amazon Linux
 - Runtime(s): Standard
 - Image: aws/codebuild/amazonlinux2-x86_64-standard:5.0
- Additional configuration**:
 - Image version: Always use the latest image for this runtime version
 - Use GPU-enhanced compute: Unchecked
 - Service role:
 - New service role (selected): Create a service role in your account
 - Existing service role: Choose an existing service role from your account
 - Role name: codebuild-under-desk-treadmill-service-role
 - Timeout:
 - Default timeout is 1 hour
 - Hours: 1
 - Minutes: 0
 - Queued timeout:
 - Default time in build queue is 8 hours
 - Hours: 8
 - Minutes: 0
 - Privileged:
 - Enable this flag if you want to build Docker images or want your builds to get elevated privileges (checkbox checked)
 - Certificate:
 - Do not install any certificate (selected)
 - Install certificate from your S3 bucket
 - VPC: Select a VPC that your AWS CodeBuild project will access.

A blue callout bubble points to the "Privileged" section in the Additional configuration step, containing the text: "For docker image".

3. Implementation

Setup CodeBuild: (Continue 2/3)

Configure a CodeBuild Project to take the contents of the CodeCommit repository, build a docker image, and store it in the ECR.

Compute

3 GB memory, 2 vCPUs
 7 GB memory, 4 vCPUs
 15 GB memory, 8 vCPUs
 70 GB memory, 36 vCPUs
 145 GB memory, 72 vCPUs

Environment variables

Name	Value	Type	Remove
AWS_DEFAULT_REGION	ap-northeast-1	Plaintext	Remove
AWS_ACCOUNT_ID	[REDACTED]	Plaintext	Remove
IMAGE_TAG	latest	Plaintext	Remove
IMAGE_REPO_NAME	under-desk-treadmill	Plaintext	Remove

Add environment variable

Create parameter

The environment variables will be used in buildspec.yaml

Buildspec

Build specifications

Insert build commands
Store build commands as build project configuration
 Use a buildspec file
Store build commands in a YAML-formatted buildspec file

Buildspec name - optional
By default, CodeBuild looks for a file named buildspec.yml in the source code root directory. If your buildspec file uses a different name or location, enter its path from the source root here (for example, buildspec-two.yml or configuration/buildspec.yml).

buildspec.yml

Batch configuration
You can run a group of builds as a single execution. Batch configuration is also available in advanced option when starting build.

Define batch configuration - optional
You can also define or override batch configuration when starting a build batch.

Artifacts

Add artifact

Artifact 1 - Primary

Type
No artifacts

You might choose no artifacts if you are running tests or pushing a Docker image to Amazon ECR.

Additional configuration
Cache, encryption key

3. Implementation

Setup CodeBuild: (Continue 3/3)

Configure a CodeBuild Project to take the contents of the CodeCommit repository, build a docker image, and store it in the ECR.

Logs

CloudWatch

CloudWatch logs - optional
Checking this option will upload build output logs to CloudWatch.

Group name - optional
`aws/codebuild/underdesk-treadmill`

The group name of the logs in CloudWatch Logs. The log group name will be `/aws/codebuild/<project-name>` by default.

Stream name prefix - optional
`udtreadmill`

The prefix of the stream name of the CloudWatch Logs.

S3

S3 logs - optional
Checking this option will upload build output logs to S3.

[Cancel](#) [Create build project](#)

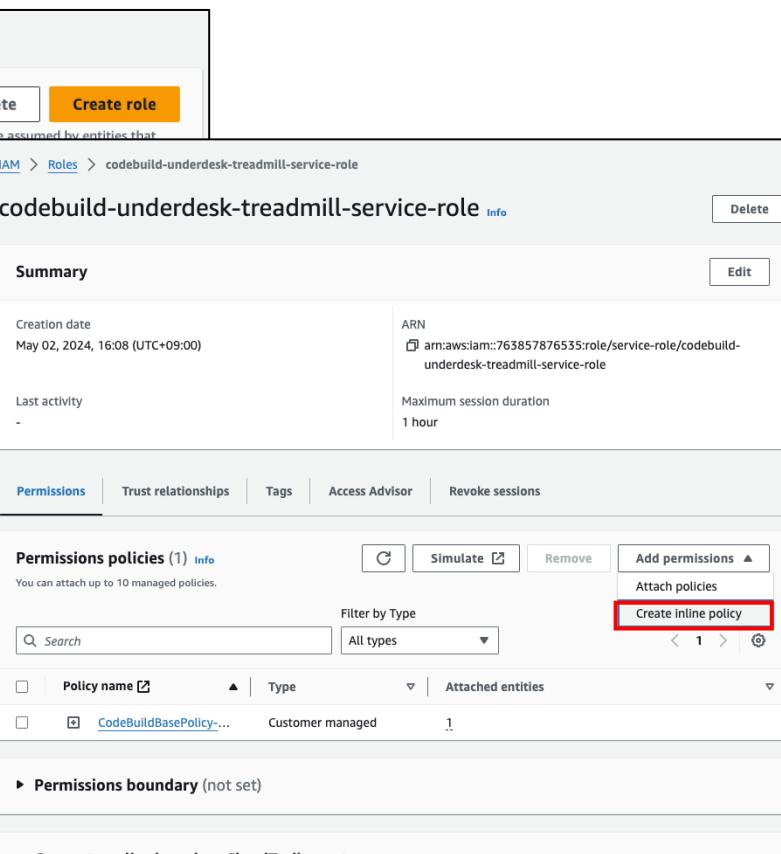
3. Implementation

Configure IAM Role for CodeBuild security and permissions: (Continue 1/2)

The build project will use IAM Role which is automatically created by CodeBuild. For the Build Project to access ECR to store a docker image, update the role permissions with ALLOWs for ECR.



A screenshot of the AWS IAM Roles page. A search bar at the top has 'underdesk' typed into it. Below the search bar, there is a table with two rows. The first row has a checkbox next to 'Role name' and the value 'codebuild-underdesk-treadmill-service-role'. The second row has a checkbox next to 'AWS Service:' and the value 'codebuild-underdesk-treadmill-service-role'. A red arrow points from the 'codebuild-underdesk-treadmill-service-role' entry in the table to the 'Permissions' tab of the detailed view on the right.



A screenshot of the detailed view for the 'codebuild-underdesk-treadmill-service-role'. The 'Permissions' tab is selected. At the top of the 'Permissions' section, there is a button labeled 'Create inline policy' with a red box around it. Below this, there is a table showing one attached policy: 'CodeBuildBasePolicy-...' (Customer managed). The table includes columns for 'Policy name', 'Type', and 'Attached entities'.

Create a new inline policy in JSON. Delete the skelton JSON and replace with this.
Name the new policy 'codebuild-ecr'

Policy editor

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Action": [  
6         "ecr:BatchCheckLayerAvailability",  
7         "ecr:CompleteLayerUpload",  
8         "ecr:GetAuthorizationToken",  
9         "ecr:InitiateLayerUpload",  
10        "ecr:PutImage",  
11        "ecr:UploadLayerPart"  
12      ],  
13      "Resource": "*",  
14      "Effect": "Allow"  
15    }  
16  ]  
17 }
```

How to test CodeBuild will be presented in the latter slides.

3. Implementation

Configure IAM Role for CodeBuild security and permissions: (Continue 2/2)

The build project will use IAM Role which is automatically created by CodeBuild. For the Build Project to access ECR to store a docker image, update the role permissions with ALLOWs for ECR.

The screenshot shows the 'codebuild-underdesk-treadmill-service-role' IAM role configuration page. The 'Summary' tab is selected, displaying details like Creation date (May 02, 2024, 16:08 (UTC+09:00)), ARN (arn:aws:iam::763857876535:role/service-role/codebuild-underdesk-treadmill-service-role), Last activity (None), and Maximum session duration (1 hour). Below the summary, the 'Permissions' tab is active, showing 'Permissions policies (2)'. It lists two policies: 'codebuild-ecr' (Customer inline, 0 attachments) and 'CodeBuildBasePolicy-underdesk-treadmill-ap-northeast-1' (Customer managed, 1 attachment). Other tabs include 'Trust relationships', 'Tags', 'Access Advisor', and 'Revoke sessions'.

Policy name	Type	Attached entities
codebuild-ecr	Customer inline	0
CodeBuildBasePolicy-underdesk-treadmill-ap-northeast-1	Customer managed	1

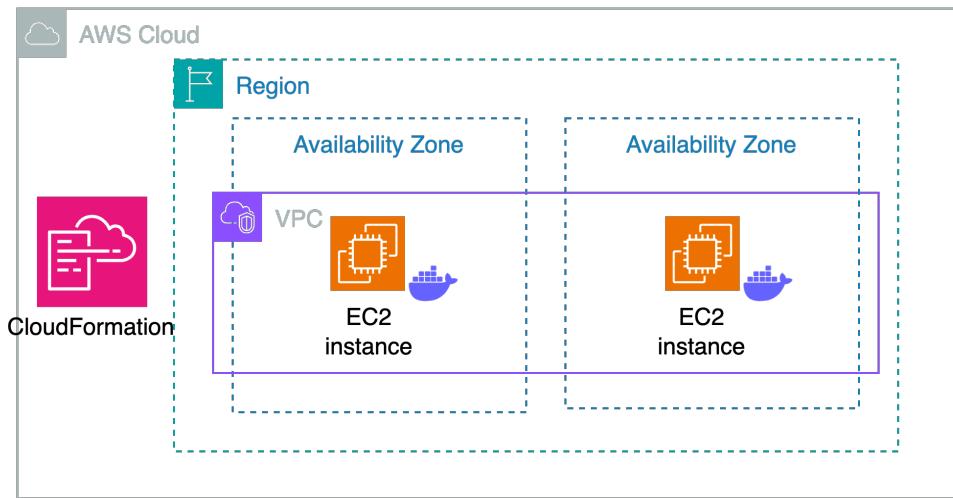
The IAM Role for CodeBuild should look like this.

3. Implementation

Use CloudFormation to automatically build an application's environment that fulfills the following conditions:

- ① set up a network infrastructure
- ② set EC2 instances with docker installed in each public subnet, using nginx as the web server.

Use `ec2docker.yaml` to build the stack. This template will launch EC2 instances with Docker installed like below:



The image displays two screenshots of the AWS CloudFormation and Docker services. The left screenshot shows the CloudFormation console with the 'Stacks' section open, displaying a single stack named 'Docker' in the 'CREATE_COMPLETE' state. The right screenshot shows the Docker service's 'Events' page, which lists 86 events from March 17, 2024, including the creation of EC2 instances and Route Tables.

Timestamp	Logical ID	Status	Detailed status	Status reason
2024-03-17 11:49:12 UTC+0900	Docker	CREATE_COMPLETE	-	-
2024-03-17 11:49:11 UTC+0900	PublicEC2	CREATE_COMPLETE	-	-
2024-03-17 11:48:40 UTC+0900	PublicEC2	CREATE_IN_PROGRESS	-	Resource creation initiated
2024-03-17 11:48:39 UTC+0900	PublicEC2	CREATE_IN_PROGRESS	-	-
2024-03-17 11:48:38 UTC+0900	SessionManagerInstanceProfile	CREATE_COMPLETE	-	-
2024-03-17 11:46:36 UTC+0900	RouteTableWebDefaultIPv4	CREATE_COMPLETE	-	-
2024-03-17 11:46:35 UTC+0900	RouteTableWebDefaultIPv4	CREATE_IN_PROGRESS	-	Resource creation initiated
2024-03-17 11:46:34 UTC+0900	RouteTableWebDefaultIPv4	CREATE_IN_PROGRESS	-	-

How to test the docker images will be presented in the latter slides.

3. Implementation

Setup CodeDeploy: Configure automated deployment of the application to ECS Fargate.

Before setting up CodeDeploy, set up an application load balancer (ALB). Docker will consume AWS resources. Using ALB is a key to load off the stress from them. (Continue)

Create Application Load Balancer Info

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

► How Application Load Balancers work

Basic configuration

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

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

Scheme Info
Scheme can't be changed after the load balancer is created.

Internet-facing
An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. [Learn more](#)

Internal
An internal load balancer routes requests from clients to targets using private IP addresses.

IP address type Info
Select the type of IP addresses that your subnets use.

IPv4
Includes only IPv4 addresses.

Dualstack
Includes IPv4 and IPv6 addresses.

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

VPC Info
Select the virtual private cloud (VPC) for your targets or you can [create a new VPC](#). Only VPCs with an internet gateway are enabled for selection. The selected VPC can't be changed after the load balancer is created. To confirm the VPC for your targets, view your [target groups](#).

vpc-02f69f1651c3891f
IPv4 VPC CIDR: 172.31.0.0/16

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

ap-northeast-1a (apne1-az4)
Subnet

IPv4 address
Assigned by AWS

ap-northeast-1c (apne1-az1)
Subnet

IPv4 address
Assigned by AWS

ap-northeast-1d (apne1-az2)
Subnet

IPv4 address
Assigned by AWS

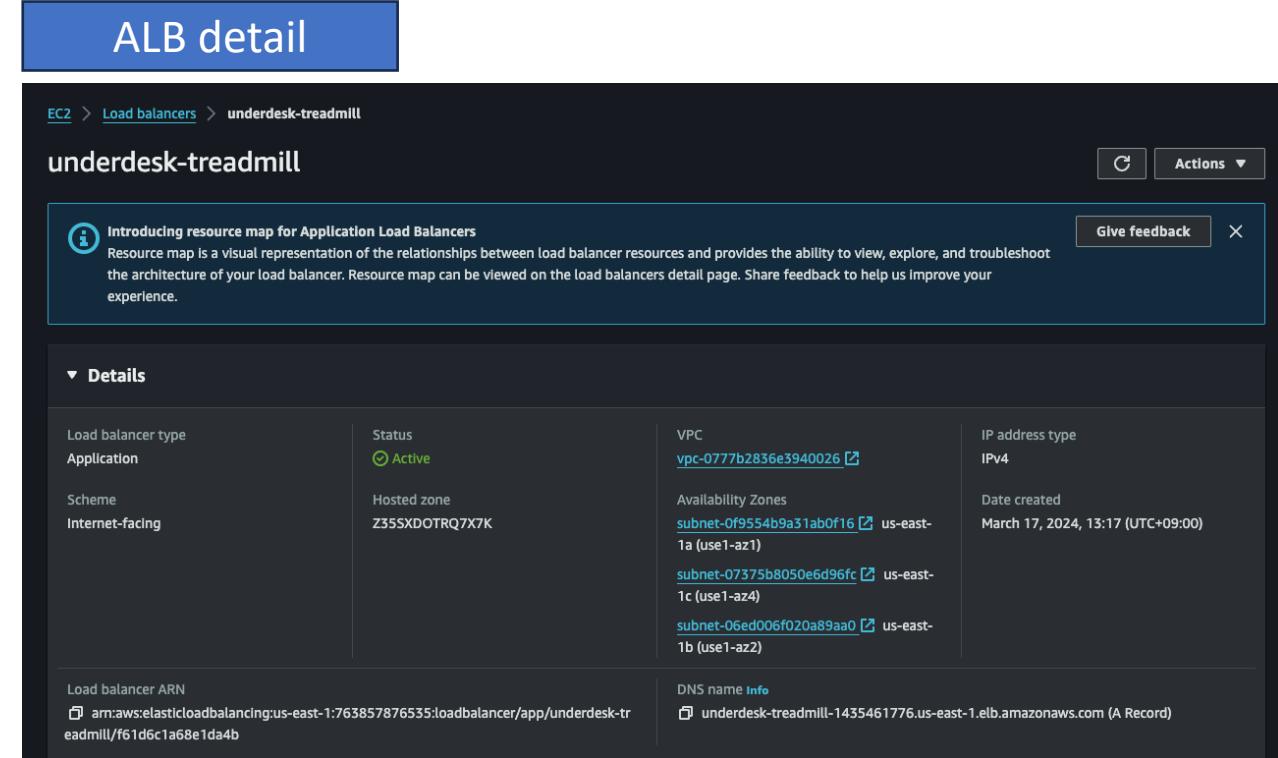
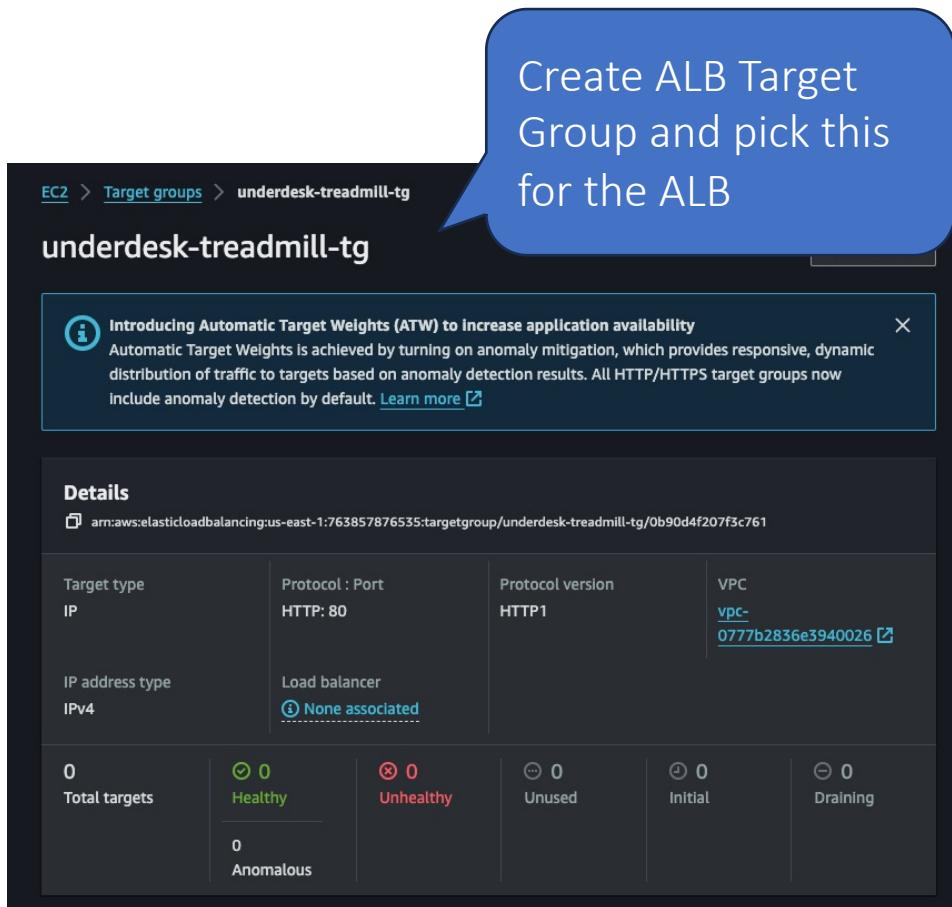
Choose default VPC

3. Implementation

Setup CodeDeploy:

Configure automated deployment of the application to ECS Fargate.

Set up an application load balancer (ALB). Docker will consume AWS resources. Using ALB is a key to load off the stress from them.



3. Implementation

Setup ECS Fargate:

1. Create the ECS Cluster
2. Create Task and Container Definitions

Amazon Elastic Container Service > Task definitions > udt-demo > Revision 1 > Create service

Create Info

Environment AWS Fargate

Existing cluster

▼ 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.

Platform version Info
Specify the platform version on which to run your service.

Create new task definition Info

Task definition configuration

Task definition family Info
Specify a unique task definition family name.

Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.

► Infrastructure requirements
Specify the infrastructure requirements for the task definition.

▼ Container - 1 Info Essential container

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 <input type="text" value="underdesk-treadmill"/>	Image URI <input type="text" value="763857876535.dkr.ecr.ap-northeast-1.amazonaws.com/underdesk-treadmill"/>	Essential container <input checked="" type="checkbox" value="Yes"/>
---	--	---

Private registry Info
Store credentials in Secrets Manager, and then use the credentials to reference images in private registries.

Private registry authentication

Port mappings Info
Add port mappings to allow the container to access ports on the host to send or receive traffic. For port name, a default will be assigned if left blank.

Container port	Protocol	Port name	App protocol
<input type="text" value="80"/>	<input type="text" value="TCP"/>	<input type="text" value="container-port-pro"/>	<input type="text" value="HTTP"/>
<input type="button" value="Remove"/>			

3. Implementation

Setup ECS Fargate:

1. Create a service from Container Definitions

Deployment options

Deployment type: Rolling update Blue/green deployment (powered by AWS CodeDeploy)

Min running tasks %: values in %

Max running tasks %: values in %

Deployment failure detection: Turned on

Networking

VPC: default

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

Choose subnets: Clear current selection

subnet-0f283c0fa252e51d9 X ap-northeast-1a 172.31.32.0/20
subnet-07a3dcdb442e812b3 X ap-northeast-1c 172.31.0.0/20
subnet-0ae312cab7c0d5d99 X ap-northeast-1d 172.31.16.0/20

Security group: Choose an existing security group or create a new security group.

Choose security groups:

sg-037c805cbc4ddf39c X underdesk-treadmill-sg-2
sg-0235b8c04c156913f X default | -

Public IP: Turned on

Load balancing - optional

Configure load balancing using Amazon Elastic Load Balancing to distribute traffic evenly across the healthy tasks in your service.

Load balancer type: Info

Configure a load balancer to distribute incoming traffic across the tasks running in your service.

Container: The container and port to load balance the incoming traffic to

underdesk-treadmill 80:80

Host port:Container port

Application Load Balancer

Specify whether to create a new load balancer or choose an existing one.

Create a new load balancer

Use an existing load balancer

Load balancer

Select the load balancer you wish to use to distribute incoming traffic across the tasks running in your service.

underdesk-treadmill-2

Health check grace period: seconds

Listener: Listener

Specify the port and protocol that the load balancer will listen for connection requests on.

Create new listener

Use an existing listener: 80:HTTP

Listener rules for 80:HTTP (1)

Traffic received by the listener is routed according to its rules. Rules are evaluated in priority order, from the lowest value to the highest value. The default rule is evaluated last.

Evaluation order: Rule path: Target group:

default / underdesk-treadmill-latest3

Target group: Info

Specify whether to create a new target group or choose an existing one that the load balancer will use to route requests to the tasks in your service.

Create new target group

Use an existing target group: underdesk-treadmill-latest3

Target group name: underdesk-treadmill-latest3

Health check path: /

Health check protocol: HTTP

3. Implementation

Create CodePipeline:

1. Create a pipeline
2. Add source stage
3. Add build stage
4. Add deploy stage

The screenshot shows the AWS CodePipeline creation interface across four main sections: Pipeline settings, Source, Deploy - optional, and Build - optional.

Pipeline settings:

- Pipeline name:** underdesk-treadmill
- Pipeline type:** V2 (selected)
- Execution mode:** Queued (Pipeline type V2 required) (selected)
- Service role:** New service role (selected)
- Role name:** AWSCodePipelineServiceRole-ap-northeast-1-underdesk-treadmill
- Advanced settings:**
 - Artifact store:** Default location (selected)
 - Encryption key:** Default AWS Managed Key (selected)

Source:

- Source provider:** AWS CodeCommit
- Repository name:** under-desk-treadmill
- Branch name:** main
- Change detection options:** AWS CodePipeline (selected)
- Output artifact format:** CodePipeline default (selected)

Deploy - optional:

- Deploy provider:** Amazon ECS
- Region:** Asia Pacific (Tokyo)
- Cluster name:** underdesk-treadmill
- Service name:** udt-service
- Image definitions file - optional:** imagedefinitions.json
- Deployment timeout - optional:** (empty field)
- Configure automatic rollback on stage failure

Build - optional:

- Build provider:** AWS CodeBuild
- Region:** Asia Pacific (Tokyo)
- Project name:** underdesk-treadmill
- Environment variables - optional:** (empty field)
- Build type:** Single build (selected)

That's it for implementation.

4. Test Results:

Test build:

1. Go to CodeBuild Console, select “underdesk-treadmill-build” and click “Start build”

The screenshot shows the AWS CodeBuild console interface. At the top, the navigation path is: Developer Tools > CodeBuild > Build projects > underdesk-treadmill-build > underdesk-treadmill-build:42081594-3e60-4205-8382-f7b55ce8ca6c. Below the path, the build ID is displayed: underdesk-treadmill-build:42081594-3e60-4205-8382-f7b55ce8ca6c. There are two buttons: "Stop build" (gray) and "Retry build" (orange). The main section is titled "Build status". It contains the following details:

Status	Initiator	Build ARN
Succeeded	chanty-practice	arn:aws:codebuild:ap-northeast-1:763 [REDACTED]:build/underdesk-treadmill-build: d:42081594-3e60-4205-8382-f7b55ce8ca 6c
Resolved source version	Start time	End time
df9d839616c487da6e52fab7d9ba2e31c52 Of9ba	May 5, 2024 3:23 PM (UTC+9:00)	May 5, 2024 3:23 PM (UTC+9:00)
Build number		
3		

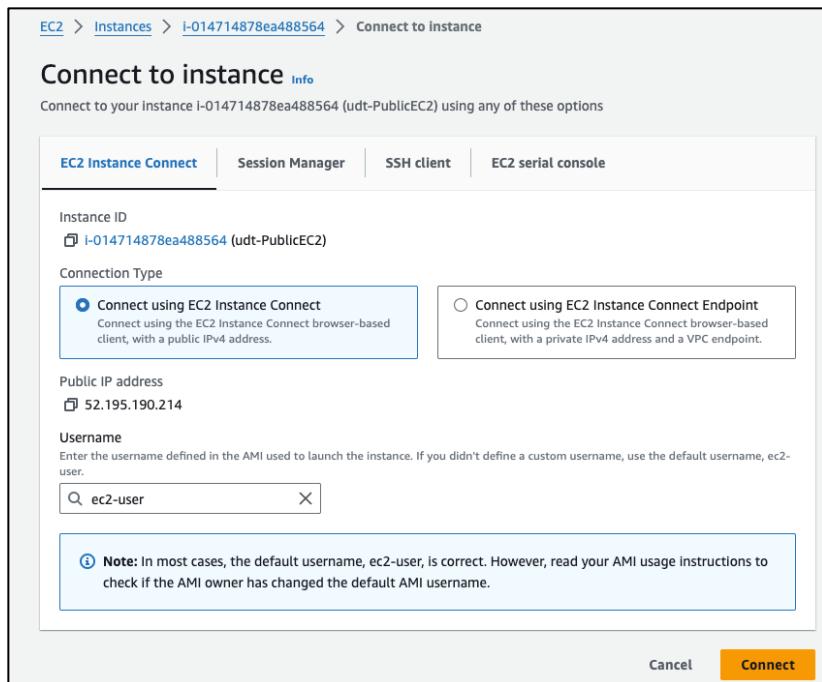
Below the status section, there are tabs: Build logs (selected), Phase details, Reports, Environment variables, Build details, and Resource utilization. The "Build logs" tab shows the last 210 lines of the build log. A message says "Showing the last 210 lines of the build log. [View entire log](#)". A "Tail logs" button is available. The log content starts with:

```
1 [Container] 2024/05/05 06:23:17.742318 Running on CodeBuild On-demand
2 [Container] 2024/05/05 06:23:17.742462 Waiting for agent ping
3 [Container] 2024/05/05 06:23:17.943003 Waiting for DOWNLOAD SOURCE
```

4. Test Results:

Test your Docker image: (Continue 1/3)

1. Go to EC2 Instance Console and click the EC2 instance created by the CloudFormation
2. Use EC2 Instance Connect to login to your instance



run the following commands:

```
aws ecr get-login-password --region ap-northeast-1 | docker login --  
username AWS --password-stdin <YOUR-AWS-ACCOUNT-ID>.dkr.ecr.ap-  
northeast-1.amazonaws.com
```

```
[ec2-user@ip-10-16-48-197 ~]$ aws ecr get-login-password --region ap-northeast-1 | docker login --username AWS --password-stdin  
ap-northeast-1.amazonaws.com  
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.  
Configure a credential helper to remove this warning. See  
https://docs.docker.com/engine/reference/commandline/login/#credentials-store  
Login Succeeded  
[ec2-user@ip-10-16-48-197 ~]$
```

4. Test Results:

Test your Docker image: (Continue 2/3)

```
# run the following commands:
```

```
docker pull <YOUR-AWS-ACCOUNT-ID>.dkr.ecr.ap-northeast-1.amazonaws.com/under-desk-treadmill
```

```
docker images # copy the image ID
```

```
Docker run -p 80:80 <IMAGE_ID>
```

```
[ec2-user@ip-10-16-52-20 ~]$ docker pull .....dkr.ecr.ap-northeast-1.amazonaws.com/underdesk-treadmill
Using default tag: latest
latest: Pulling from underdesk-treadmill
b0a0cf830b12: Pull complete
8ddb1e6cdf34: Pull complete
5252b206aac2: Pull complete
988b92d96970: Pull complete
7102627a7a6e: Pull complete
93295add984d: Pull complete
ebde0aald1aa: Pull complete
3fd9ff9e3799: Pull complete
3b508517b633: Pull complete
a9ec7b0141fd: Pull complete
cd89e971cb4b: Pull complete
Digest: sha256:df5073f1d69fe6ca4a8f5e0e1203ea9d1d2658d04de21eb12714c
Status: Downloaded newer image for .....dkr.ecr.ap-northeast-1.amazonaws.com/underdesk-treadmill
```

```
[ec2-user@ip-10-16-52-20 ~]$ docker run -p 80:80 f267e7eb7b85
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/05/05 06:58:01 [notice] 1#1: using the "epoll" event method
2024/05/05 06:58:01 [notice] 1#1: nginx/1.25.5
2024/05/05 06:58:01 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/05/05 06:58:01 [notice] 1#1: OS: Linux 4.14.336-257.568.amzn2.x86_64
2024/05/05 06:58:01 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 32768:65536
2024/05/05 06:58:01 [notice] 1#1: start worker processes
2024/05/05 06:58:01 [notice] 1#1: start worker process 29
```

4. Test Results:

Test your Docker image: (Continue 3/3)

Instances (1/1) [Info](#)

[Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Find Instance by attribute or tag (case-sensitive) [All states](#)

Name	Instance ID	Instance state	Instance type	Status check
<input checked="" type="checkbox"/> udt-PublicEC2	i-0fc6b9ab8e321e0b3	Running	t2.micro	2/2 healthy

i-0fc6b9ab8e321e0b3 (udt-PublicEC2)

[Details](#) [Status and alarms](#)

Instance summary [Info](#)

Instance ID i-0fc6b9ab8e321e0b3 (udt-PublicEC2)	Public IPv4 address 13.231.105.37 open address	Private IPv4 addresses 10.16.52.20
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-13-231-105-37.ap-northeast-1.compute.amazonaws.com open address
Hostname type IP name: ip-10-16-52-20.ap-northeast-1.compute.internal	Private IP DNS name (IPv4 only) ip-10-16-52-20.ap-northeast-1.compute.internal	

1. Copy the public IPv4
2. Type “<http://<IPv4>>” (It isn’t https)

Not Secure 13.231.105.37 [Relaunch to update](#)

[All Bookmarks](#)

Under-Desk Treadmill

Stay Active While Working



An under-desk treadmill allows you to stay active and burn calories while working on your tasks.

Space Saving Design



4. Test Results:

After creating the task definition and service for the ECS, test your Application Load Balancer.

EC2 > Load balancers > underdesk-treadmill-alb

underdesk-treadmill-alb

Details

Load balancer type Application	Status Active	VPC vpc-02f69f1651c3891f2 [edit]	IP address type IPv4
Scheme Internet-facing	Hosted zone Z14GRHDCWA56QT	Availability Zones subnet-0f283c0fa252e51d9 [edit] ap-northeast-1a (apne1-az4)	Date created May 5, 2024, 16:31 (UTC+09:00)

1. Copy the DNS name
2. Type “http://<DNS_name>” (It isn’t https)

DNS name copied

Load balancer ARN
[arn:aws:elasticloadbalancing:ap-northeast-1:
\[REDACTED\]:loadbalancer/app/underdesk-trea
dmill-alb/38df23ab5fcf55a](#)

[underdesk-treadmill-alb-1363880882.ap-no
rtheast-1.elb.amazonaws.com \(A Record\)](#)

Not Secure underdesk-treadmill-alb-1363880882.ap-northeast-1.elb.amazonaws.com Relaunch to update

Under-Desk Treadmill

Stay Active While Working



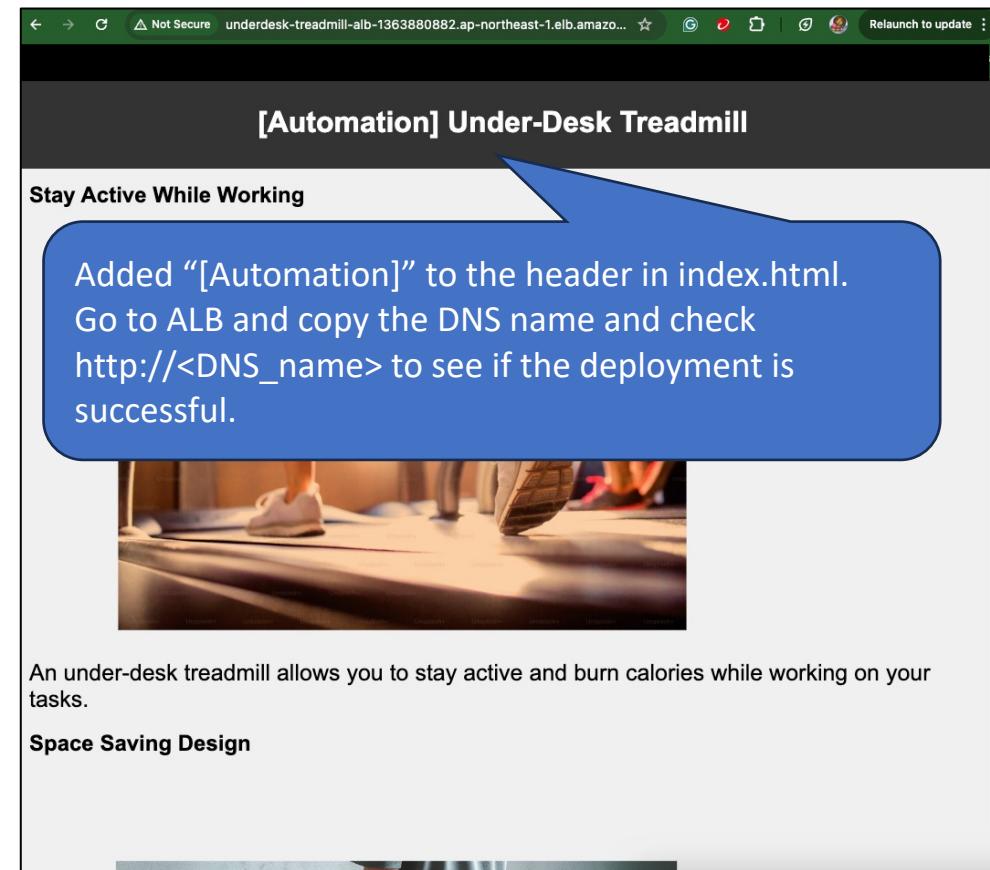
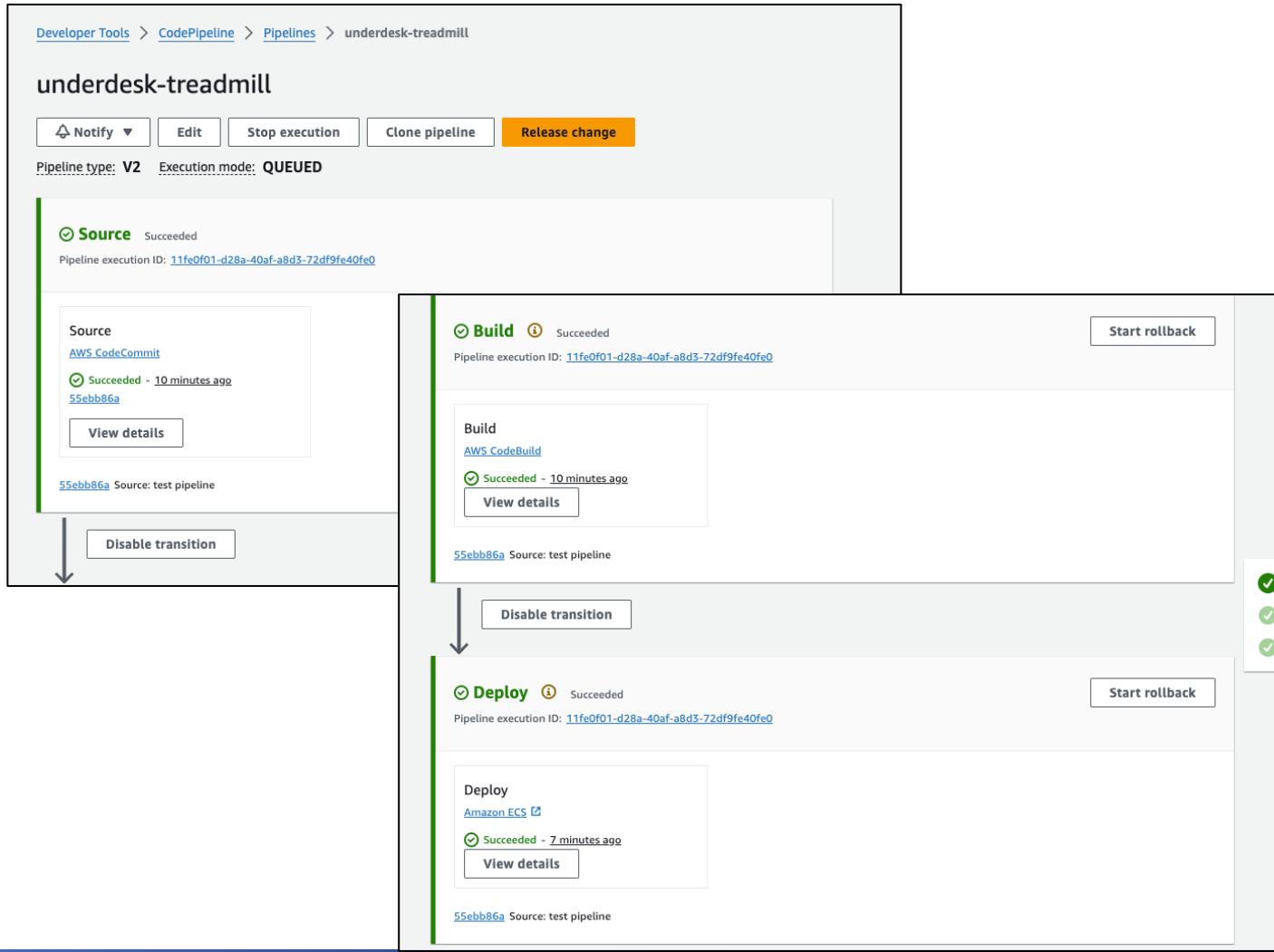
An under-desk treadmill allows you to stay active and burn calories while working on your tasks.

Space Saving Design



4. Test Results:

After creating the CodePipeline with Deploy stage, add one or two modifications to index.html to test the pipeline.



Fin.