

ECS Fargate Nginx Server Deployment with ALB And EFS using CodePipeline - Terraform (Task 8)



Zaeem Attique Ashar

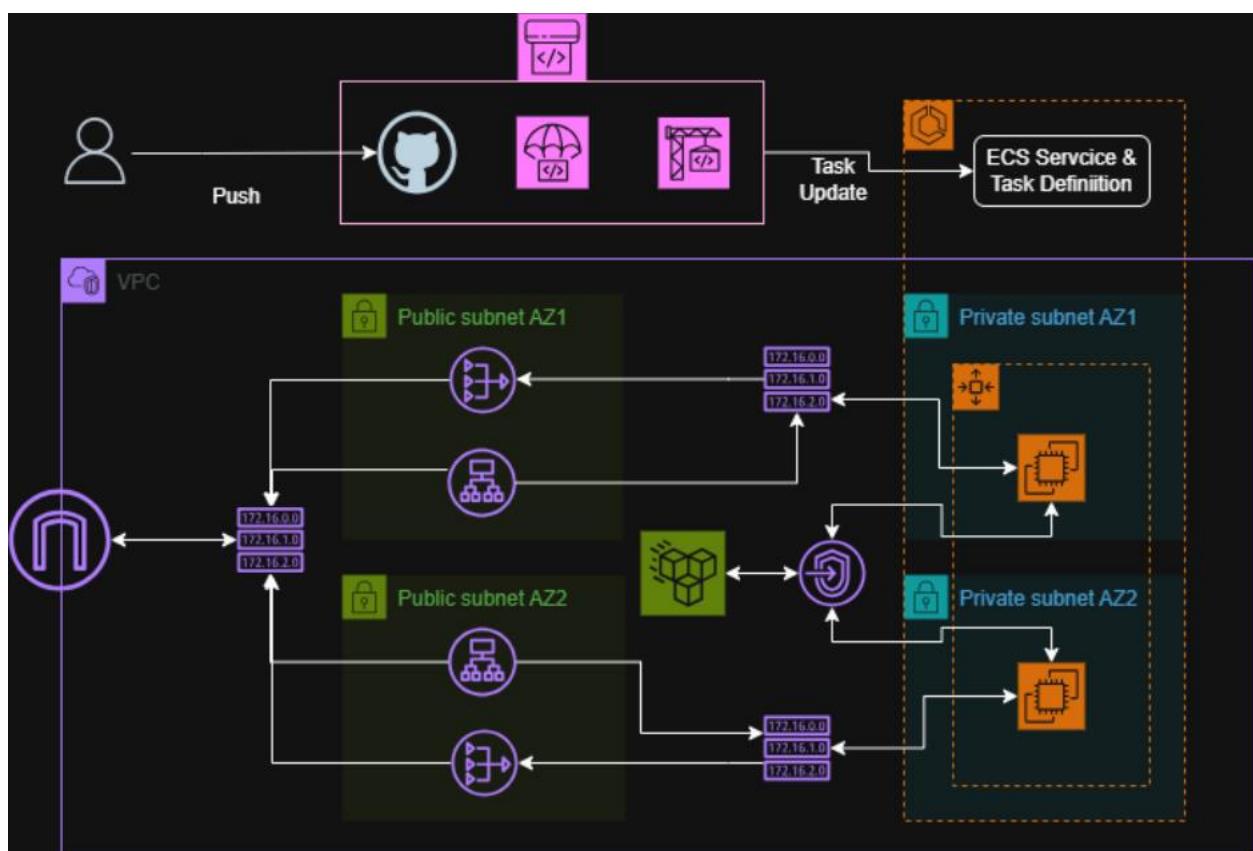
Cloud Intern

Task Description:

This task will be a guide on setting up a Node Application server with a highly available architecture on AWS ECS Cluster Spread over multiple availability zones. An internet facing Load Balancer will be set up in front of the ECS cluster to listen to traffic and balance it upon tasks. AWS EFS will be used for persistence in shared storage to prevent data loss. The application will automatically be deployed using the AWS CodePipeline. The complete infrastructure will be coded in Terraform modules.

Task8.1: Create basic networking infrastructure	4
Task8.2: Create EFS File System and Mount Points	4
Task8.3: Build Docker Image and push to ECR.....	6
Task8.4: Create an ECS + EC2 Cluster and dependencies	6
Task8.5: Creating the CodePipeline	11
Task8.6: Verification and Testing	12
Task8.7: Problems Faced.....	13

Architecture Diagram:



Task8.1: Create basic networking infrastructure

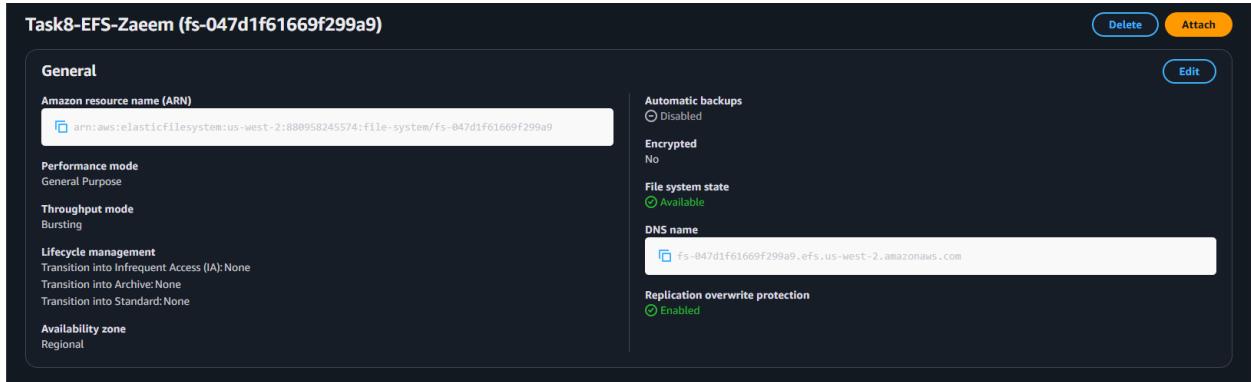
- Create and configure a VPC
 - CIDR Block: 10.0.0.0/16
- Create and configure Subnets
 - Public Subnet A (us-west-2a), CIDR: 10.0.1.0/24
 - Private Subnet A (us-west-2a), CIDR: 10.0.2.0/24
 - Public Subnet B (us-west-2b), CIDR: 10.0.3.0/24
 - Private Subnet A (us-west-2a), CIDR: 10.0.4.0/24
- Create and configure NAT Gateways
 - NAT Gateway A in Public Subnet A
 - NAT Gateway B in Public Subnet B
- Create and configure Internet Gateway
 - Create and attach to the project's VPC
- Create and configure Route Tables
 - Public Route Table, Outbound rule: 0.0.0.0/0 -> IGW, attach to Public SN A&B
 - Private Route Table A, Outbound Rule: 0.0.0.0/0 -> NGW attach to Private SN A
 - Private Route Table B, Outbound Rule: 0.0.0.0/0 -> NGW attach to Private SN B



Task8.2: Create EFS File System and Mount Points

- Create File System:
 - Name: Task8-EFS-Zaeem
 - Encryption: Yes
- Create Access Points:
 - File System: Task8-EFS-Zaeem

- o Name: Task8-EFS-AP-Zaeem
 - o Root Directory: /
 - o POSIX UID 1000, GID 1000
 - o Owner UID 1000, Owner GID: 1000
 - o AP Permissions: 0755
- Create Mount Points:
 - o AZ us-west-2a, Subnet Private, SG Task8-EFS-SG-Zaeem
 - o AZ us-west-2b, Subnet Private, SG Task8-EFS-SG-Zaeem



Task8.3: Build Docker Image and push to ECR

- Build Docker Image:
 - o Got to source code directory.
 - o Create DockerFile.yaml and write instructions
 - o Use command to build image: `docker image -t nodejs:latest -f DockerFile.yaml`.
 - o Use to tag image: `docker tag nodejs:latest <ecr uri>/zaeem/Task8:latest`
 - o Use command to push to ECR: `docker push <ecr uri>/zaeem/Task8:latest`

```
*PS C:\Users\zaeem\Documents\Innovation Lab - Cloudelligent\Task7\node-js-sample-master> aws ecr get-login-password --region us-west-2 | docker login --username AWS --password-stdin 504649076991.dkr.ecr.us-west-2.amazonaws.com
Login Succeeded
*PS C:\Users\zaeem\Documents\Innovation Lab - Cloudelligent\Task7\node-js-sample-master> docker build -t zaeem/task7 -f .\Dockerfile.yaml .
[+] Building 1.9s (10/10) FINISHED
--> [internal] load build definition from Dockerfile.yaml
--> [internal] transfering dockerfile: 199B
--> [internal] load metadata for docker.io/library/node:18-alpine
--> [internal] load .dockerignore
--> transfering context: 2B
--> [1/5] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca99d9e
--> [2/5] COPY package.json .
--> [3/5] COPY package-lock.json .
--> [4/5] RUN npm install --production
--> [5/5] COPY .
--> exporting image
--> exporting layers
--> exporting manifest sha256:5f4261775cbe94c003335a607953064cf9668818537e9fb2293d0bfffed2c
--> exporting config sha256:8bae0dd551541294c0b508caef453c328c7bd2c2b0d27897e16e30c7779dace
--> exporting attestation manifest sha256:1fd7b12ccde085e20d9195ab3d13e51fc300c5e6d1113dac6eefb92ae836a6
--> exporting manifest list sha256:730e95da8cf21f547b178ffa99695da51324667e6b5b17c8d81335f306942ed8
--> naming to doc
--> unpacking to docker-desktop/dashboard/build/desktop-linux/desktop-linux/8m8lchhrz9cw6bzr4jbbegofm (ctrl + click)
View build details: docker-desktop//dashboard/build/desktop-linux/desktop-linux/8m8lchhrz9cw6bzr4jbbegofm
*PS C:\Users\zaeem\Documents\Innovation Lab - Cloudelligent\Task7\node-js-sample-master> docker tag zaeem/task7:latest 504649076991.dkr.ecr.us-west-2.amazonaws.com/zaeem/task7:latest
*PS C:\Users\zaeem\Documents\Innovation Lab - Cloudelligent\Task7\node-js-sample-master> docker push 504649076991.dkr.ecr.us-west-2.amazonaws.com/zaeem/task7:latest
504649076991: Pushed
a2a0fd24a6cff: Pushed
1e54a893ce5: Pushed
25ff2da03641: Pushed
96805a348ee2: Pushed
cd71de3405: Pushed
7908e4cf363: Pushed
8c02218931c7: Pushed
f1823114bc9: Pushed
latest: digest: sha256:730e95da8cf21f547b178ffa99695da51324667e6b5b17c8d81335f306942ed8 size: 856
*PS C:\Users\zaeem\Documents\Innovation Lab - Cloudelligent\Task7\node-js-sample-master>
```

Task8.4: Create an ECS + EC2 Cluster and dependencies

- Create and Configure a Launch Template
 - o Name: Task8-EC2-LT-Zaeem
 - o Container instance AMI: Amazon Linux 2023
 - o Instance Type: t3.micro
 - o SSH Key pair: Task8-EC2
 - o Subnet: Do not include
 - o Availability Zone: Do not include
 - o Security Group: Task8-EC2-SG-Zaeem
 - o Storage Volume: 8GiB Default EBS volume

Task8-Launch-Template-Zaeem-20251202141903412000000005 (lt-03dadf894321fd96f)

Actions ▾ Delete template

Launch template details

Launch template ID lt-03dadf894321fd96f	Launch template name Task8-Launch-Template-Zaeem-20251202141903412000000005	Default version 1	Owner arn:aws:iam::880958245574:user/Zaeem
--	--	----------------------	---

Details Versions Template tags

Launch template version details

Version 1 (Default)	Description -	Date created 2025-12-02T14:19:03.000Z	Created by arn:aws:iam::880958245574:user/Zaeem
Instance details		Storage	Resource tags
AMI ID ami-07b09ad3acfff075f	Instance type t3.micro	Availability Zone -	Availability Zone Id -
Key pair name -	Security groups -	Security group IDs sg-07c3ab8569e31bb5d	

- Create and Configure Application Load Balancer
 - Name: Task8-ALB-Zaeem
 - Scheme: Internet Facing
 - Load Balancer IP: IPv4
 - VPC: Task8-VPC-Zaeem
 - AZ and Subnets: AZ1 Public SN & AZ2 Public SN
 - Security Groups: Task8-ALB-SG-Zaeem
 - Listener Protocol HTTP, Port 80
 - Routing Action: Forward to TG
 - Register Targets: None

The screenshot shows the AWS Elastic Load Balancing (ALB) console for the load balancer **Task8-ALB-A-Zaeem**. The **Details** section provides an overview of the load balancer's configuration, including its type (Application), status (Active), scheme (Internet-facing), VPC (vpc-08e12832b2d654c42), availability zones (us-west-2b (usw2-az1) and us-west-2a (usw2-az2)), and ARN (arn:aws:elasticloadbalancing:us-west-2:880958245574:loadbalancer/app/Task8-ALB-A-Zaeem/3330680984198999). The **Listeners and rules** tab is selected, displaying one listener rule for port 5000 (HTTP) that forwards traffic to the **Task8-ALB-Target-Group-Zaeem** target group. The rule has a 100% weight and no conditions or SSL termination.

- Create and Configure Auto Scaling Group
 - Name: Task8-ASG-Zaeem
 - Launch Template: Task8-EC2-LT-Zaeem
 - VPC: Task8-VPC-Zaeem
 - AZ: Private SN A & Private SN B
 - Load Balancer: Attach to existing
 - Attach to an Existing Target Group
 - Group Size: Desired capacity 2, min 1, max 3
 - No scaling policies

The screenshot shows the AWS ECS Capacity Provider console for the capacity provider **Task8-ECS-Capacity-Provider-Zaeem**. The **Capacity provider overview** section displays the provider's name (Task8-ECS-Capacity-Provider-Zaeem), scaling type (EC2 Auto Scaling), update status (-), and status (Active). The **Capacity provider ARN** is also listed.

- Create and Configure ECS Cluster
 - Name: Task8-ECS-Cluster-Zaeem
 - Infrastructure: Fargate and self-managed instances
 - Select Task8-ASG-Zaeem Auto Scaling Group

The screenshot shows the AWS CloudWatch Metrics interface for the 'Task8-ECS-Cluster-Zaeem' cluster. At the top, there's a summary section with ARN (arn:aws:ecs:us-west-2:880958245:574:cluster/Task8-ECS-Cluster-Zaeem), Status (Active), CloudWatch monitoring (Default), and Registered container instances (2). Below this, there are two tabs: 'Services' and 'Tasks'. Under 'Services', it shows one service named 'Task8-ECS-Service-Zaeem' with ARN arn:aws:ecs:us-west-2:880958245:574:task-definition/Task8-ECS-Service-Zaeem, Status Active, and Type EC2. Under 'Tasks', it shows 1 Active task, 0 Pending tasks, and 2 Running tasks.

Cluster overview

ARN arn:aws:ecs:us-west-2:880958245:574:cluster/Task8-ECS-Cluster-Zaeem	Status Active	CloudWatch monitoring Default	Registered container instances 2
--	------------------	----------------------------------	-------------------------------------

Services

Draining	Active 1	Pending	Running 2
----------	-------------	---------	--------------

Tasks

Services | **Tasks** | **Infrastructure** | **Metrics** | **Scheduled tasks** | **Configuration** | **Event history** | **Tags**

Services (1) Info

Filter services by value	Filter launch type Any launch type	Filter scheduling strategy Any scheduling strategy	Filter resource management type Any resource management type	Manage tags	Update	Delete service	Create
Service name Task8-ECS-Service-Zaeem	ARN arn:aws:ecs:us-west-2:880958245:574:task-definition/Task8-ECS-Service-Zaeem	Status Active	Type REPLICA	EC2	Task definition Task8-ECS-Service-Zaeem	2/2 Tasks	Deployments and tasks

- Create a Task Definition for the ECS Cluster
 - o Name: Task8-NodeJS-Zaeem
 - o Infrastructure Requirements: Amazon EC2 Instances
 - o OS&Arch: Linux/x64, Network Mode: bridge
 - o vCPU: 1, Memory: 3GB
 - o Select Task role and Task Execution Roles
 - o Container 1 (essential): Name NodeJS-App, ECR Image URI, Port Mapping 3000 to 80 for accessing the nodejs application
 - o Storage: Configure at task definition, Vol Type: EFS, Enter EFS ID, Enter AP ID

Last updated December 2, 2025, 20:51 (UTC+5:00) [C](#) Deploy Actions Create new revision ▾

Overview Info

ARN arn:aws:ecs:us-west-2:88095824 5574:task-definition/Task8-ECS-Task-Definition-Zaeem:17	Status ACTIVE	Time created December 2, 2025, 20:34 (UTC+5:00)	App environment EC2
Task role Task8-ecs-task-role-Zaeem	Task execution role Task8-ecs-task-execution-role-Zaeem	Operating system/Architecture -	Network mode bridge
Fault injection -			

Containers JSON Task placement Volumes (1) Requires attributes Tags

Task size

Task CPU 256 units (0.25 vCPU)	Task memory 256 MiB (0.25 GB)
Task CPU maximum allocation for containers	Task memory maximum allocation for container memory reservation
CPU (unit) 0 20 40 60 80 100 120 140 160 180 200 220 240	Memory (MiB) 0 20 40 60 80 100 120 140 160 180 200 220 240
■ NodeJS-App ■ Shared task CPU	■ NodeJS-App ■ Shared task memory

- Create ECS Service to run Task Definition:
 - o Family: Task8-NodeJS-Zaeem
 - o Svc Name: Task8-NodeJS-Zaeem-Service
 - o Compute option: Capacity Provider Strategy, custom: base 2, weight 1
 - o Scheduling Strategy: Replica
 - o Desired Task: 2
 - o Load Balancer: Task8-VPC-Zaeem, Type: ALB, Container: NodeJS-App, Load Balancer: Task8-ALB-Zaeem

Task8-ECS-Service-Zaeem [Info](#)

Last updated December 2, 2025, 20:50 (UTC+5:00) [Delete service](#) [Update service](#) ▾

Service overview [Info](#)

Status Active	Tasks (2 Desired) 0 Pending 2 Running	Task definition: revision Task8-ECS-Task-Definition-Zaeem:17	Deployment status Success
-------------------------------	---	--	---

Tasks | Logs | Deployments | Events | **Configuration and networking** | Service auto scaling | Event history | Tags

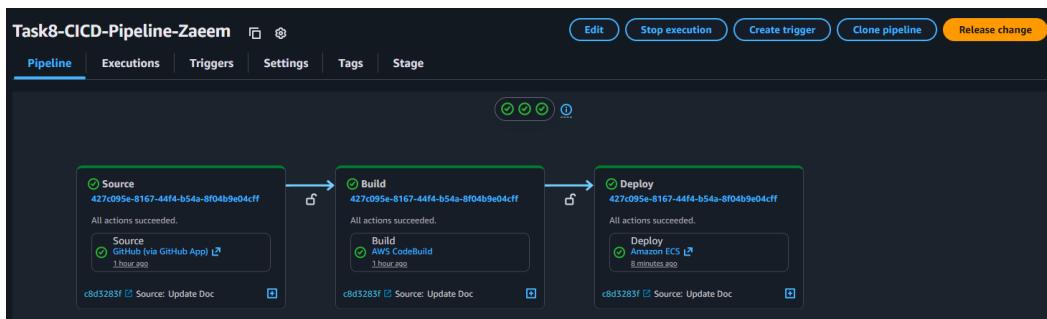
Service configuration [Info](#)

Service ARN arn:aws:ecs:us-west-2:880958245574:service/Task8-ECS-Cluster-Zaeem/Task8-ECS-Service-Zaeem	Task definition: revision Task8-ECS-Task-Definition-Zaeem:17	Launch type EC2	Scheduling strategy REPLICA
Created by arn:aws:iam::880958245574:user/Zaeem	Amazon ECS managed tags Turned off	Propagate tags from None	Availability Zone rebalancing Turned on
CloudFormation stack -	ECS Exec Info Turned off	▶ Task placement strategy and constraints	

Task8.5: Creating the CodePipeline

- Basic Pipeline Configuration:
 - Use the Build Pipeline button to create a new pipeline.
 - Category: Custom Pipeline
 - Pipeline Name: Task8-Pipeline-Zaeem
 - Execution mode: Queued
 - Service Role: Create new
- Source Stage:
 - Source Provider: GitHub (OAuth via app)
 - Connect to GitHub and allow the pipeline connection
- Build Stage:
 - Build Provider: Other (AWS CodeBuild)
 - Create a new project
 - Set Environment Variables:
 - REGION="us-west-2"
 - ACCOUNT_ID="504649076991"
 - REPO_NAME="zaeem/Task8"
 - IMAGE_TAG="latest"
 - Build Type: Single Build
 - Region: United States Oregon

- o Input Artifacts: SourceArtifact
- o Place the buildspec.yml file in the root of repo
- Deploy Stage (skipping the test stage):
 - o Deploy Provider: Amazon ECS
 - o Region: United States (Ohio)
 - o Input Artifacts: BuildArtifact
 - o Place the buildspec.yml file in the root of repo
 - o Cluster Name: Task8-ECS-Cluster-Zaeem
 - o Service Name: Task8-NodeJS-Zaeem-Service
 - o Deployment Time Out: 15 min
 - o Create Pipeline



Task8.6: Verification and Testing

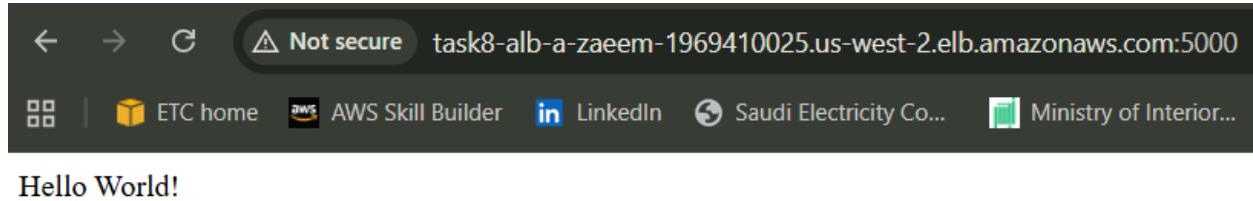
- Any commits to the GitHub code repository will trigger the pipeline



- The tasks deployed can be seen under the ECS Service

The screenshot shows the AWS Elastic Container Service (ECS) console. It displays the "Task8-ECS-Cluster-Zaeem" cluster overview and the "Task8-Service-Zaeem" service details. The cluster has 2 registered container instances and 1 active task. The service has 1 active task and 2 running tasks. The service details show 1 task in progress, with ARN: arn:aws:ecs:us-west-2:880958245574:cluster/Task8-ECS-Cluster-Zaeem and Status: Active.

- The webpage of the NodeJS application can be accessed at the DNS of the ALB



Task8.7: Problems Faced

- The resources of the containers in the task definition should be less than those set in the Launch Template for the instances otherwise the containers will not be placed and will be stuck in the pending state.
- We must specify artifacts output in the buildspec.yml file to make sure that they are saved in the S3 bucket and passed onto the Deploy Stage otherwise we get an S3 permission error which is not related to permissions at all.
- The pipeline role must have the correct policies attached to carry out all the stages.
- We need to attach the proper permissions to be able to pass the Exec and Task roles to the deploy stage of the codepipeline, as well as the s3 permissions to read the artifact files.
- The name of the container in the appspec.yml file should match exactly the name of the container in the task definition otherwise the project is not deployed.