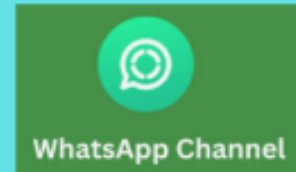
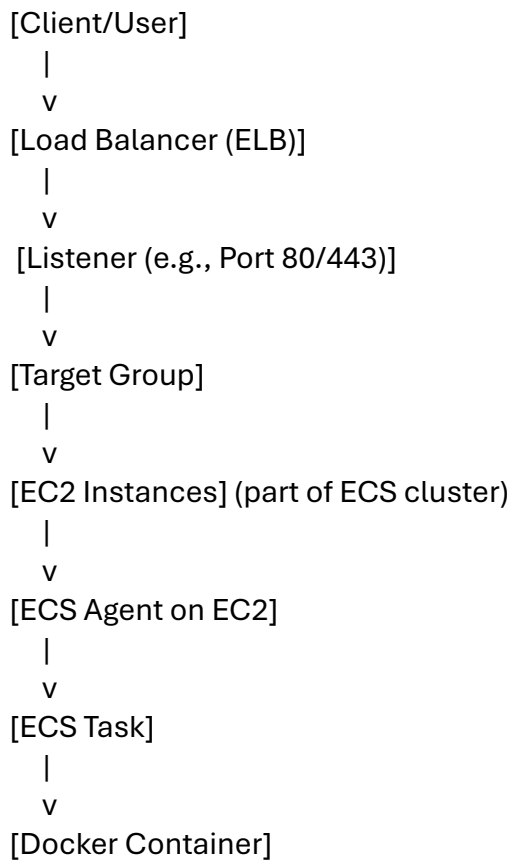


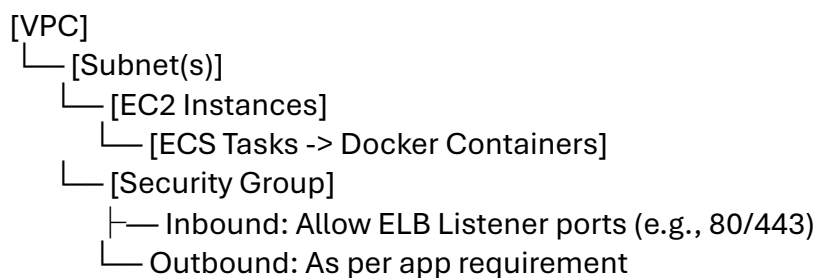
@devopschallengehub



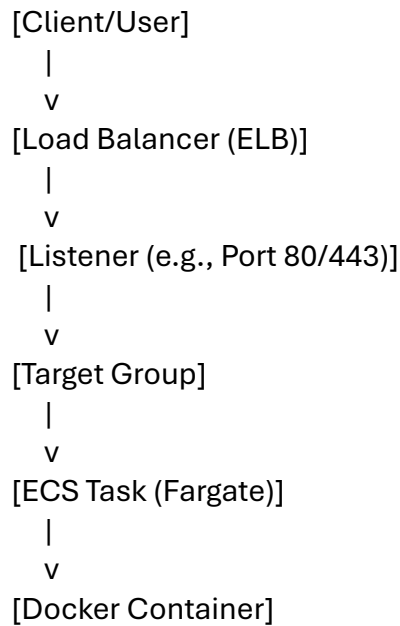
1. ECS with EC2 Launch Type



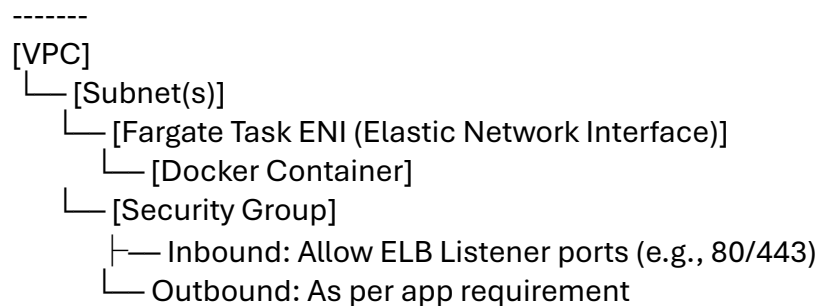
Security & Placement Context:



2. ECS with Fargate Launch Type



Security & Placement Context:



Comparison Key Points


Component	EC2 Launch Type	Fargate Launch Type
Host	You manage EC2 instances	AWS manages infrastructure
Scaling	Manual/Auto Scaling EC2	Task-based scaling
Network Mode	Shared or bridge via EC2	ENI per task
ECS Agent	Runs on EC2	Not needed (managed by AWS)
Use Case	More control, advanced customization	Fully serverless, less ops overhead

1. What are the different network modes available in ECS?

ECS supports **4 network modes** (in Task Definitions):

Network Mode	Description
bridge	Default for EC2. Containers share the host's network but use NAT.
host	Containers share the host's network directly (no port mapping).
awsvpc	Each task gets its own ENI (Elastic Network Interface) with private IP—required for Fargate.

Network Mode	Description
none	No networking. Used for advanced/isolated scenarios.

 **Note:** For **Fargate**, only **awsvpc** is supported.

What is Task Networking?

When you run a container (or group of containers called a task) in ECS, it needs a **network connection** so it can:

- Talk to the internet
- Communicate with other services (like databases or APIs)
- Receive traffic (like web requests)

So, you need to configure **how ECS gives your task an IP address**, allows traffic in/out, and connects it to the right network.

◆ For Fargate (serverless — you don't manage servers):

You must use awsvpc network mode

Think of this as giving **each task its own private IP**, like a mini virtual machine.

 Steps ECS does for you:

1. You pick a **VPC subnet** (usually private, with a NAT Gateway to access the internet).
2. You attach **security groups** to the task (like firewalls to allow/block traffic).
3. ECS gives your task a **private IP address** in the selected subnet.
4. No need to manage EC2 instances — AWS handles it for you.


Why this is good:

Each task is isolated, secure, and behaves like its own mini-server — easy to control and monitor.

◆ For EC2 (you manage EC2 instances):


You have **3 options** for network mode:

Mode	What It Means	Notes
bridge	Task shares EC2's network, but with port mapping	You must map ports manually (e.g., container port 80 → host port 8080)
host	Task uses the same network as EC2 — no port mapping needed	Fast, but less secure (no isolation between tasks)
awsvpc	Same as Fargate — each task gets its own IP address	Best option for isolation/security

 What you need to do:

- Attach **security groups to EC2 instance** (unless using awsvpc)
- Do **port mapping** if using bridge mode
- Choose network mode in task definition

✅ Best Practice Recommendation:

 **Use awsvpc network mode** (on both Fargate and EC2 if possible)

✓ Gives each task its **own IP**, better security

✓ Easier to use with service discovery, load balancers, and monitoring

Analogy:

- 🏠 bridge mode = All containers live in the same house (EC2), sharing one door. You have to label each room (port mapping) to deliver things.
 - 🏠 **awsvpc mode = Each task lives in its own house with its own address and security gate (private IP + security group).**
-

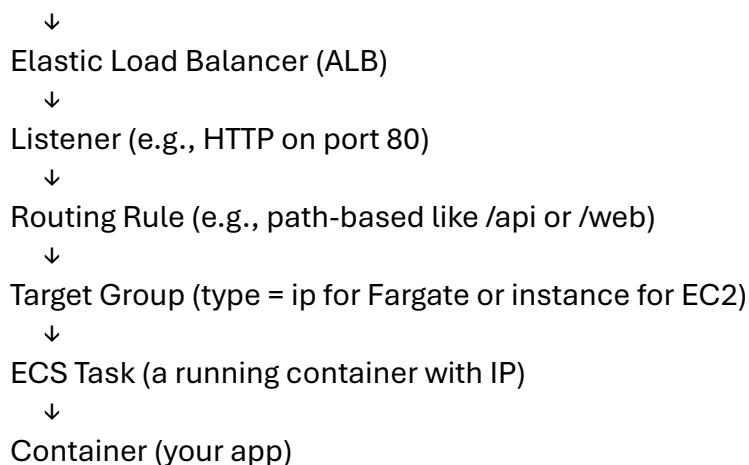
2. How do you integrate ECS with Application Load Balancer (ALB)?

Goal:

We want to **balance traffic** (like website visitors or API requests) across multiple ECS tasks (containers) using a **Load Balancer**. This gives:

- High availability
- Automatic health checks
- Support for updates with zero downtime

Client Request



What is an Application Load Balancer (ALB)?

It's like a smart traffic cop. It listens to requests (like from a browser) and sends them to the right ECS task based on **rules** (e.g., /api, /web).

🔧 Steps to Integrate ECS with ALB (Simple Explanation + Code)

✅ Step 1: Create an ALB

You can do this in the AWS Console or via **Terraform / CLI**.

Here's a CLI snippet:

bash

```
aws elbv2 create-load-balancer \  
  --name my-ecs-alb \  
  --subnets subnet-aaa subnet-bbb \  
  --security-groups sg-xyz \  
  --scheme internet-facing \  

```

--type application

💡 You must choose **at least two subnets** in different Availability Zones.

✅ Step 2: Create a Target Group

The target group is where the ALB sends traffic.

For **Fargate**, target type must be ip because Fargate tasks get private IPs.

bash

```
aws elbv2 create-target-group \  
  --name ecs-target-group \  
  --protocol HTTP \  
  --port 80 \  
  --vpc-id vpc-xxxx \  
  --target-type ip
```

✅ Step 3: Define Listener + Listener Rules

This tells the ALB:

“When someone visits this URL path, send the request to ECS tasks in the target group.”

```
aws elbv2 create-listener \  
  --load-balancer-arn arn:alb/my-ecs-alb \  
  --protocol HTTP \  
  --port 80 \  
  --default-actions Type=forward,TargetGroupArn=arn:tg/ecs-target-group
```

You can later add rules like:

- /api → send to target group A
 - /web → send to target group B
-

✅ Step 4: Configure ECS Service to Use ALB

When creating or updating your **ECS Service**, choose:

- Launch type: Fargate or EC2
- Network mode: awsvpc (required for Fargate)
- Load Balancer: Select **Application Load Balancer**
- Target Group: Select the one you created above

📌 **Sample in ECS task/service definition (JSON-like):**

json

```
"loadBalancers": [  
  {  
    "targetGroupArn": "arn:aws:elasticloadbalancing:xx:targetgroup/ecs-target-group",  
    "containerName": "my-app-container",  
    "containerPort": 80  
  }  
],  
"networkConfiguration": {  
  "awsvpcConfiguration": {
```

```
"subnets": ["subnet-aaa", "subnet-bbb"],
"securityGroups": ["sg-xyz"],
"assignPublicIp": "ENABLED"
}
}
```

✅ What Happens Automatically?

- ECS will **register** your running tasks to the ALB's target group.
- If any task fails the **health check**, it's removed automatically.
- You get **load balancing, scaling, and blue/green deployment support**.

🧠 Analogy:

Imagine your app is a restaurant with multiple chefs (containers).

The **ALB** is the receptionist who takes incoming customers and **routes** them to any available chef (ECS task), depending on their order (URL path).

✅ Summary:

Step What You Do

- | | | |
|---|--|-----------------------------------|
| 1 | Create ALB with 2 subnets | To distribute traffic |
| 2 | Create target group (type = ip) | To receive traffic from ALB |
| 3 | Create listener and routing rules | To tell ALB where to send traffic |
| 4 | Attach ALB and target group to ECS service | To link containers with ALB |

Why?

3. What is target group health checking in ECS?

A **Target Group** in ECS (used with a Load Balancer) keeps track of whether your containers (tasks) are **healthy and running properly**.

🔄 Simple Flow:

1. **ECS task** (your container) runs your app (like a website or API).
2. **Target Group** sends a health check request (like a ping or HTTP call) to each task regularly (e.g., every 30 seconds).
3. If a task **responds OK**, it's marked **healthy**. If it fails several times, it's marked **unhealthy** and **removed from load balancing**.







🧠 Why It Matters:

This ensures users only reach **working containers**.

If one crashes or hangs, it won't get traffic — so your app stays reliable.

4. Suppose we have two types of ECS tasks—one set for video conversion and another for image conversion. We use an ALB with listener rules to route requests based on the path, like /imageConv or /videoConv. If there are 100 tasks running for image conversion, how does the ALB target group forward requests to all these tasks? How are the incoming requests distributed or mapped to the 100 tasks?

How ALB Distributes Requests to ECS Tasks

1.  **Target Registration**
 - Each ECS task registers with a **target group** on startup.
 - ALB maintains a list of **healthy task IP:port pairs**.
2.  **Load Balancing Algorithm**
 - ALB uses **round-robin** by default:
 - Request 1 → Task 1, Request 2 → Task 2, ..., Request 101 → Task 1 again.
3.  **Request Flow with Path-Based Routing**
 - Client requests `/imageConv/process`
 - ALB listener rule matches `/imageConv/*`
 - ALB forwards to the **image conversion target group**
 - ALB selects one healthy task and sends the request
4.  **Health Checks & Dynamic Scaling**
 - ALB removes **unhealthy tasks** from rotation
 - **New tasks** are auto-added as they register
 - No manual intervention needed
5.  **Connection Handling**
 - **Stateless handling** by default (no sticky sessions)
 - ALB adapts to **task scaling up/down** automatically
6.  **Key Benefits**
 - **Even load distribution** across all tasks
 - **High availability** via health checks
 - **Auto-scaling friendly**
 - **Workload isolation** with path-based routing (e.g., image vs. video)

5. How do you implement blue-green deployments with ECS and ALB?

Blue-green deployment means running **two versions** of your app — the old one (**blue**) and the new one (**green**) — and smoothly switching traffic from blue to green.

Flow:

1. ECS runs two **task sets** (blue = old version, green = new version) behind the **same ALB**.
2. **Target groups** are created for each version, both connected to the ALB.
3. You use **listener rules** or **CodeDeploy** to shift traffic from blue to green after testing — with **zero downtime**.

Why It Matters:

It lets you **test safely**, **rollback easily**, and **deploy without downtime** — keeping users happy even during updates.

6. What are the security group considerations for ECS tasks?

1. **Fargate (awsvpc mode):** You assign security groups **directly to each task**.
 - Allow only necessary ports (e.g., port 80 for web).
 - Place tasks in **private subnets** with NAT if they need secure internet access.
2. **EC2 (bridge/host mode):**
 - Security groups are attached to the **EC2 instance**, not tasks.
 - You must **map ports manually** and allow those in the instance's security group.



Best Practices:

- **Least privilege:** Only open required IPs and ports.
- Avoid 0.0.0.0/0 (open to the world) unless truly needed.
- Combine **SG + NACL + private subnet** for layered security.

1. Which network mode is required for Fargate tasks in ECS?

- A. host
- B. bridge
- C. none
- D. awsvpc

D. awsvpc

2. In ECS, what does the awsvpc network mode do?

- A. Shares the host network with NAT
- B. Gives all containers a shared public IP
- C. Assigns each task its own ENI and private IP
- D. Disables networking

C. Assigns each task its own ENI and private IP

3. What is the default load balancing algorithm used by an ALB?


- A. Least latency
- B. Random
- C. Round-robin
- D. IP hash

Round-robin

4. In ECS with ALB, what is the role of the Target Group?

- A. Routes requests to the database
- B. Maintains a list of ECS task IPs and ports for routing

- C. Stores listener rules
- D. Manages scaling of ECS tasks

B. Maintains a list of ECS task IPs and ports for routing 

5. What is the primary benefit of using awsvpc mode on both EC2 and Fargate?

- A. Lower network latency
- B. Easier port mapping
- C. Better isolation and security (own IP + security group)
- D. Easier task restart

C. Better isolation and security (own IP + security group) 

6. In blue-green deployment with ECS and ALB, what is the main advantage?

- A. Uses fewer EC2 instances
- B. Allows zero-downtime switching between app versions
- C. Faster task launch time
- D. Reduces VPC costs

Allows zero-downtime switching between app versions 
