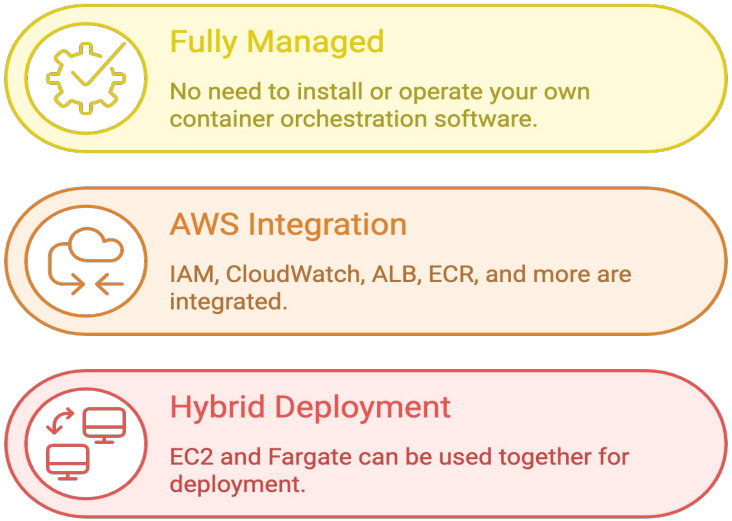
ECS

(Elastic Container Service)

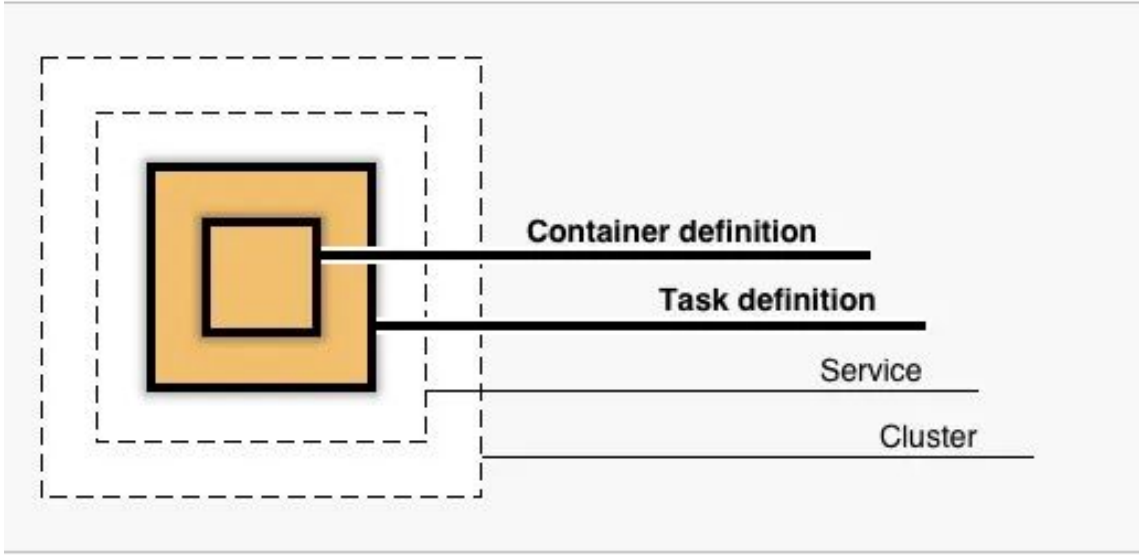
Amazon ECS

ECS – Elastic Container Service

- A fully managed container orchestration service by AWS.
- Supports Docker containers.
- Runs containers on **EC2 instances** or **AWS Fargate (serverless)**.



Components of ECS:



ECS vs K8s vs Docker Swarm

Amazon ECS

Platform	Amazon ECS	Kubernetes	Docker Swarm
Provider	AWS	Open-source (CNCF), cloud-agnostic	Docker Inc.
Management	Fully managed (with Fargate or EC2)	Self-managed or managed (e.g., EKS, GKE, AKS)	Self-managed
Scalability	High (integrated with AWS auto-scaling)	Very high (designed for large-scale deployments)	Moderate
Community & Ecosystem	AWS ecosystem	Large community, vast 3rd-party tooling	Smaller community
Vendor Lock-in	Yes (AWS-specific)	No (multi-cloud or hybrid support)	No
Use Case Fit	AWS-native workloads, quick scaling	Enterprise-scale, multi-cloud, complex systems	Simple applications, small teams

ECS Key Components

Amazon ECS

ECS Components and Their Roles

Launch Types

Methods to deploy containers

Cluster

Manages and organizes container instances

Container

Isolated environment for running applications

Task Definition

Blueprint for running containers

Service

Ensures tasks run and scale as needed

Task

Running instance of a container

1. Cluster

- A logical grouping of compute resources (EC2 or Fargate).
- ECS places and runs tasks inside the cluster.

2. Task Definition

- A JSON blueprint describing:
    - Container images
    - CPU/memory
    - Ports
    - Environment variables
    - Volumes
- Reusable for creating multiple tasks/services.

3. Task

- An instance of a Task Definition.
- One or more containers running together as defined.

4. Service

- Maintains the desired number of tasks running.
- Monitors and replaces failed tasks automatically.

5. Container

- The actual Docker container that runs inside a task.

6. Launch Types

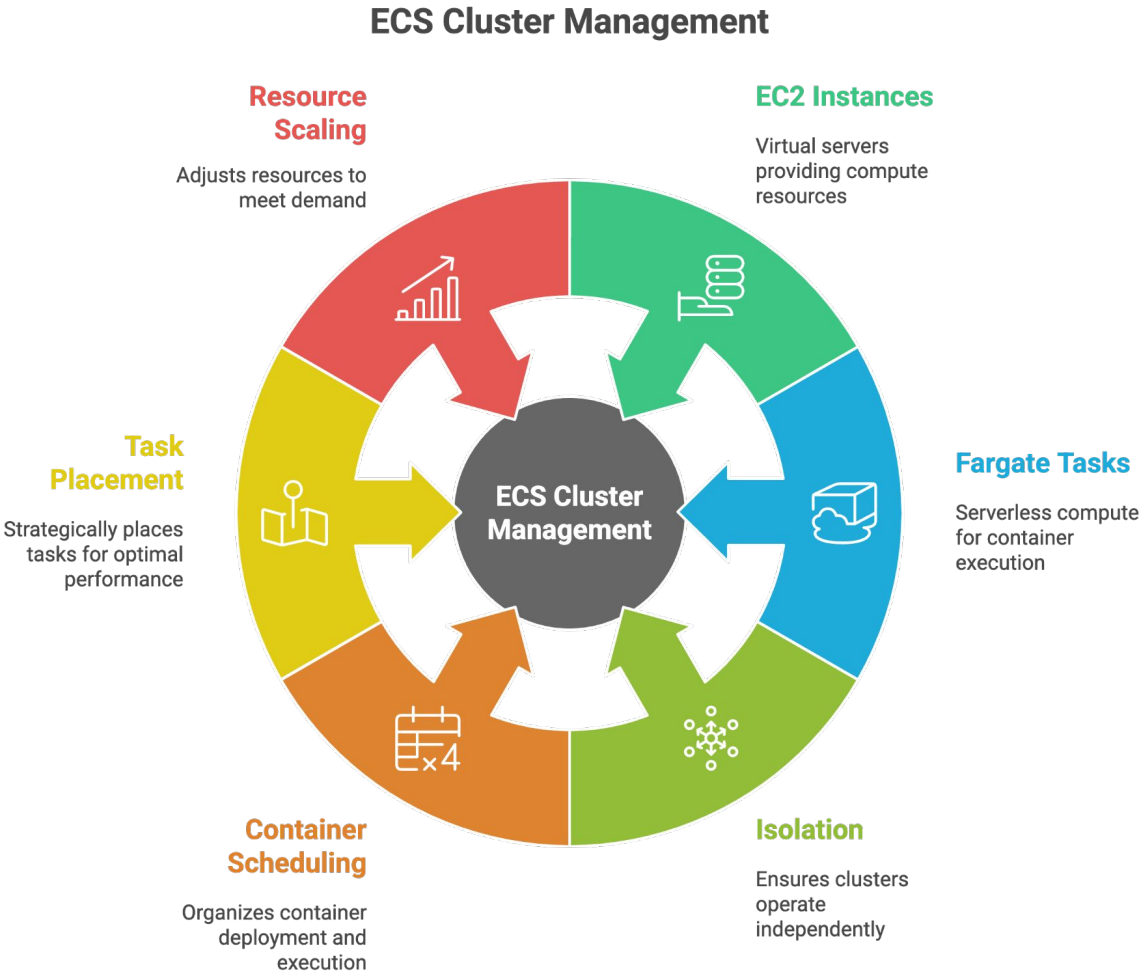
- **EC2**: You provision EC2 instances to run containers.
- **Fargate**: Serverless, AWS manages infrastructure; you only define CPU and memory.

ECS Cluster & Launch Types

Amazon ECS

ECS Cluster

- A **logical grouping of compute resources** where your containers run.
- Manages scheduling, networking, scaling, and health of tasks and services.



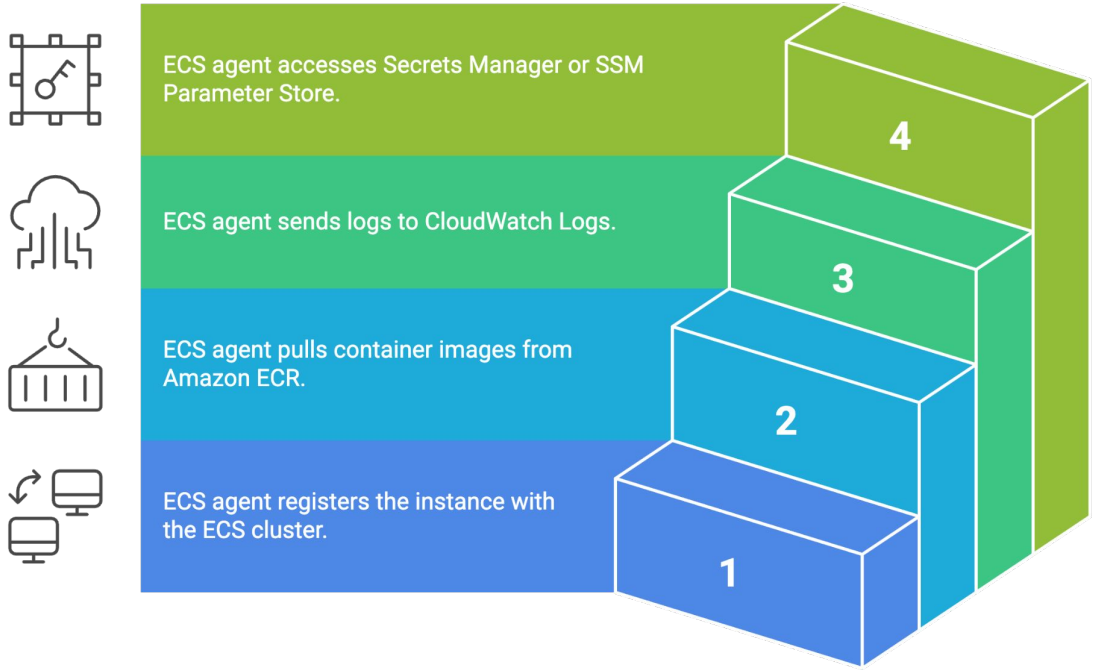
Aspect	Fargate	EC2
Infrastructure	Fully managed by AWS (serverless)	You provision and manage EC2 instances
Task Isolation	Task gets its own ENI (IP address)	Tasks share EC2 resources unless using awsvpc
Scaling	Automatic	Manual or via Auto Scaling Groups
OS/AMI Control	Not available	Full control over OS and EC2 AMI
Use Cases	Microservices, short-lived jobs	Long-running apps, custom OS or config needs
Startup Time	Slightly longer (cold start)	Faster if EC2 already running

ecsInstanceRole

Amazon ECS

ecsInstanceRole

- An **IAM role** assigned to an **EC2 instance** that is part of an ECS cluster.
- Allows the instance to perform **ECS-related actions** securely on your behalf.

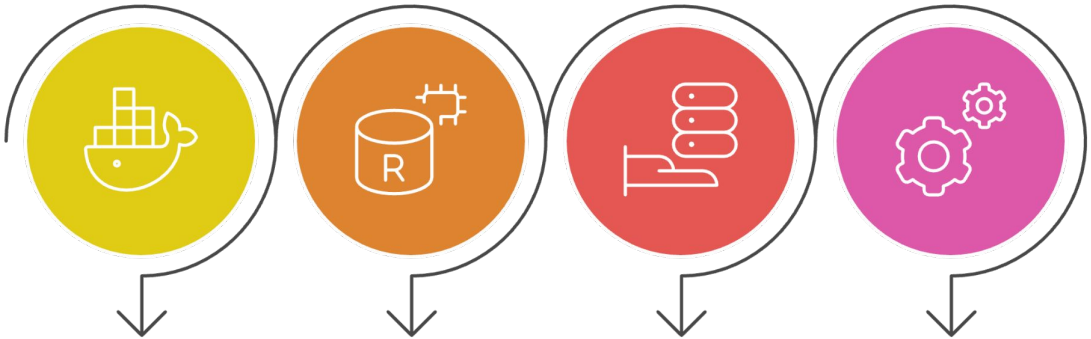


Task Definition

Amazon ECS

ECS – Task Definition

- A **Task Definition** is a blueprint that describes how Docker containers should run in Amazon EC.
- Task Definitions are **versioned**
- Required to launch **Tasks** or **Services**



Component	Description
Container Definitions	Docker image, commands, ports, volumes
Task Role	IAM role for the task to access AWS services
Execution Role	IAM role for ECS to pull images and write logs
Network Mode	awsvpc, bridge, host, or none
Resources	CPU and Memory per task or container
Volumes	Data volumes shared across containers
Environment Variables	Key-value pairs passed to the containers
Logging	Configuration for Amazon CloudWatch Logs

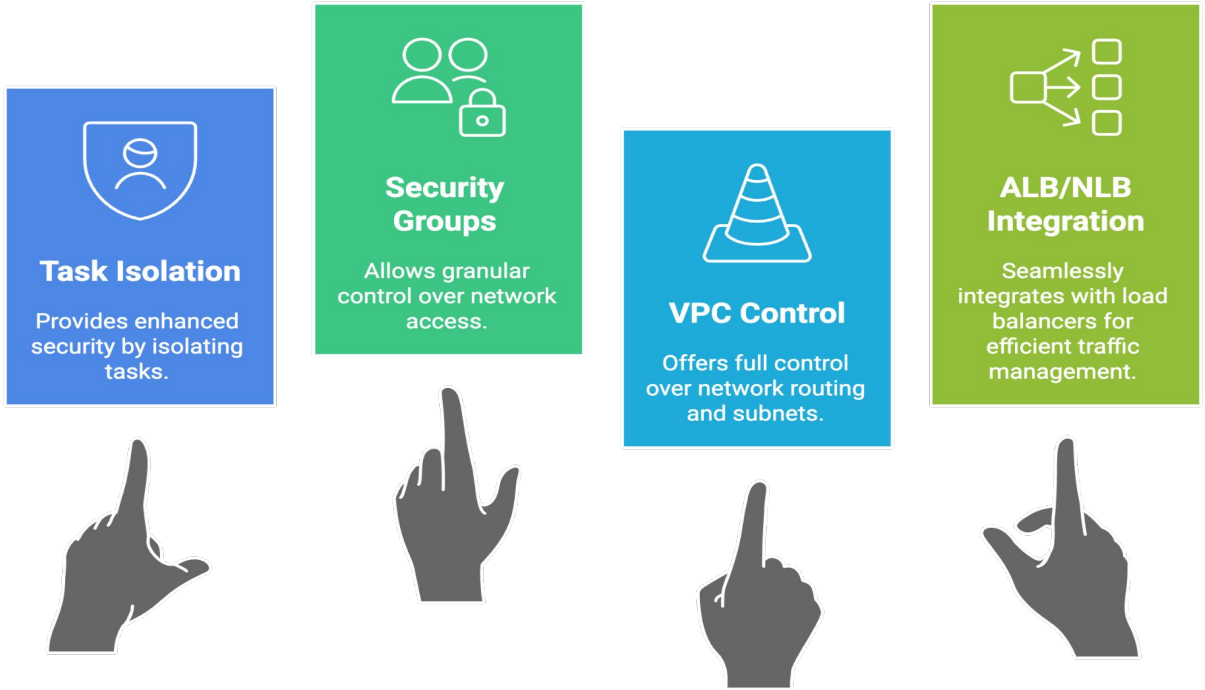
Network Modes

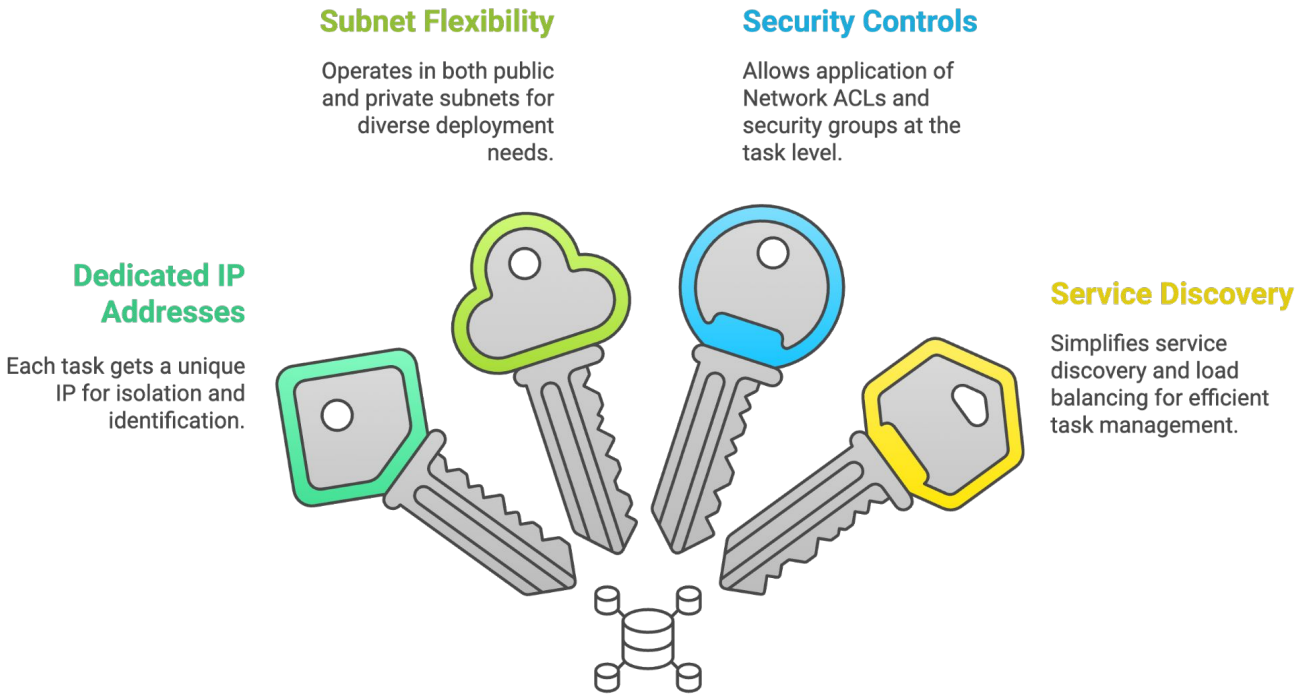
Fargate

Amazon ECS

Network Modes – Fargate

- **Fargate supports only one network mode: awsvpc**
- Each task gets its **own Elastic Network Interface (ENI)** and **private IP** in your VPC.





Use Cases:

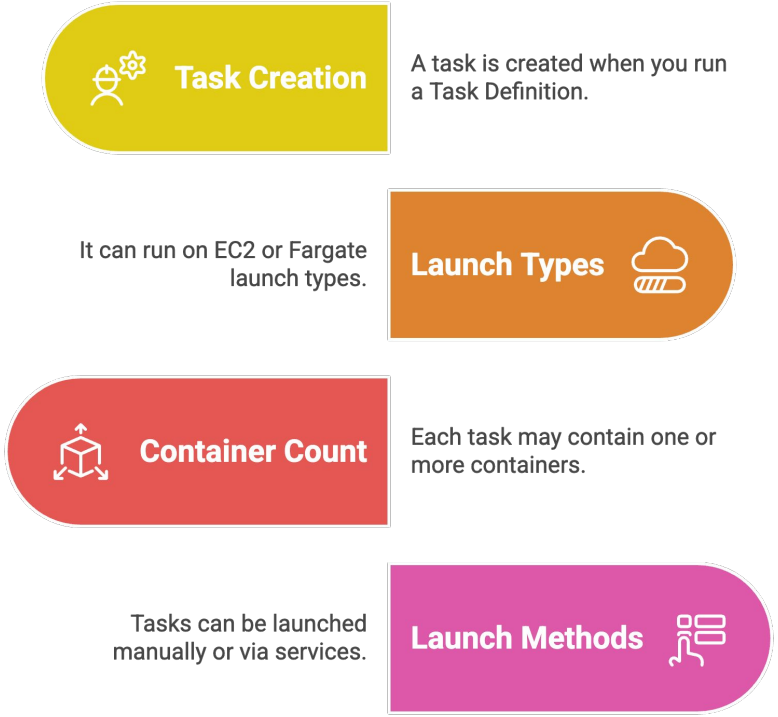
- Microservices with strict security
- Apps requiring direct communication with other VPC resources
- Tasks behind load balancers

ECS Tasks

Amazon ECS

ECS – Tasks

- A **Task** is the **running instance** of a **Task Definition** in Amazon ECS.



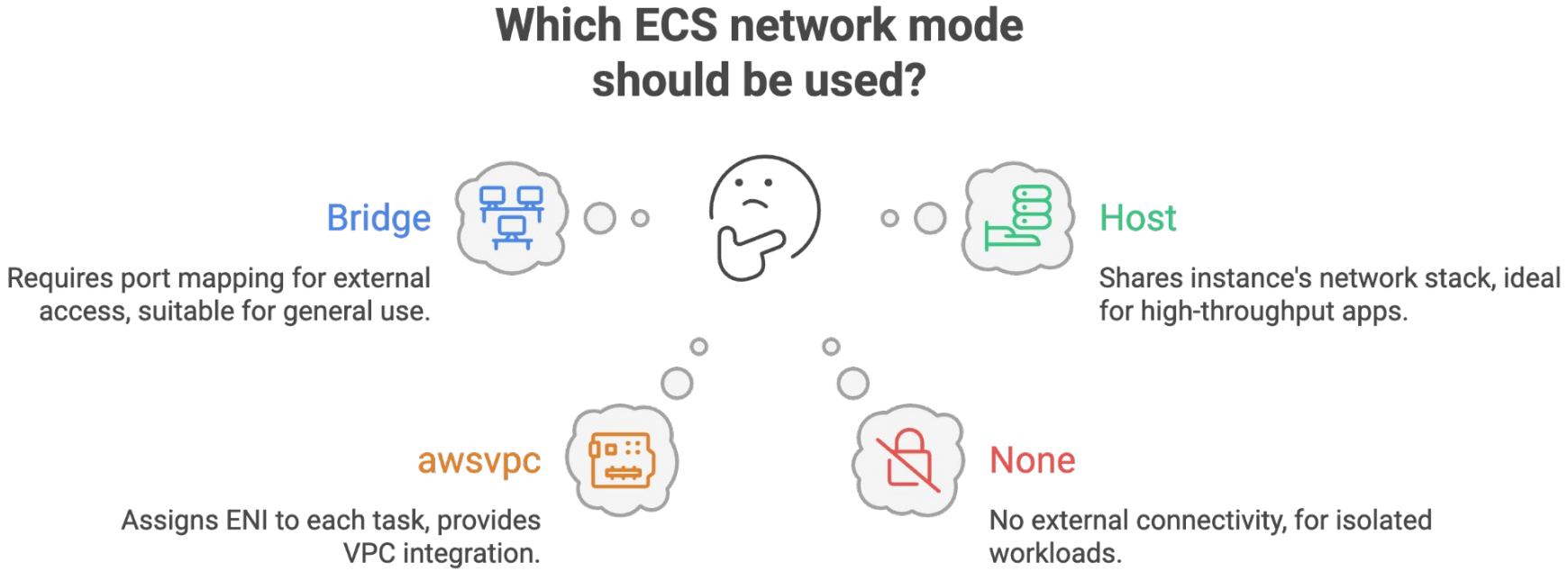
Component	Purpose
Containers	Docker images performing the workload
Networking	Configured using network mode (awsvpc, bridge, etc.)
IAM Roles	Task Role (AWS API access), Execution Role (pull images/logs)
Environment Vars	Pass runtime configuration to containers
Health Checks	Ensure the container is healthy and available
Placement	Defines where tasks run (spread, binpack, random) in EC2 mode

Network Modes

EC2

Amazon ECS

Network Modes – EC2



Mode	IP Assignment	Isolation	Performance	Use Cases
bridge	Private IP via Docker	Medium – per container	Medium	General apps needing NAT & port mapping
host	EC2 instance IP	Low – shares instance stack	High	Low-latency, high-throughput network apps
awsvpc	ENI in VPC	High – per task	High	Microservices, secure comms, direct VPC access
none	None (loopback only)	Very High – isolated	N/A	Test, dev, or isolated compute workloads

Task Placement Strategies

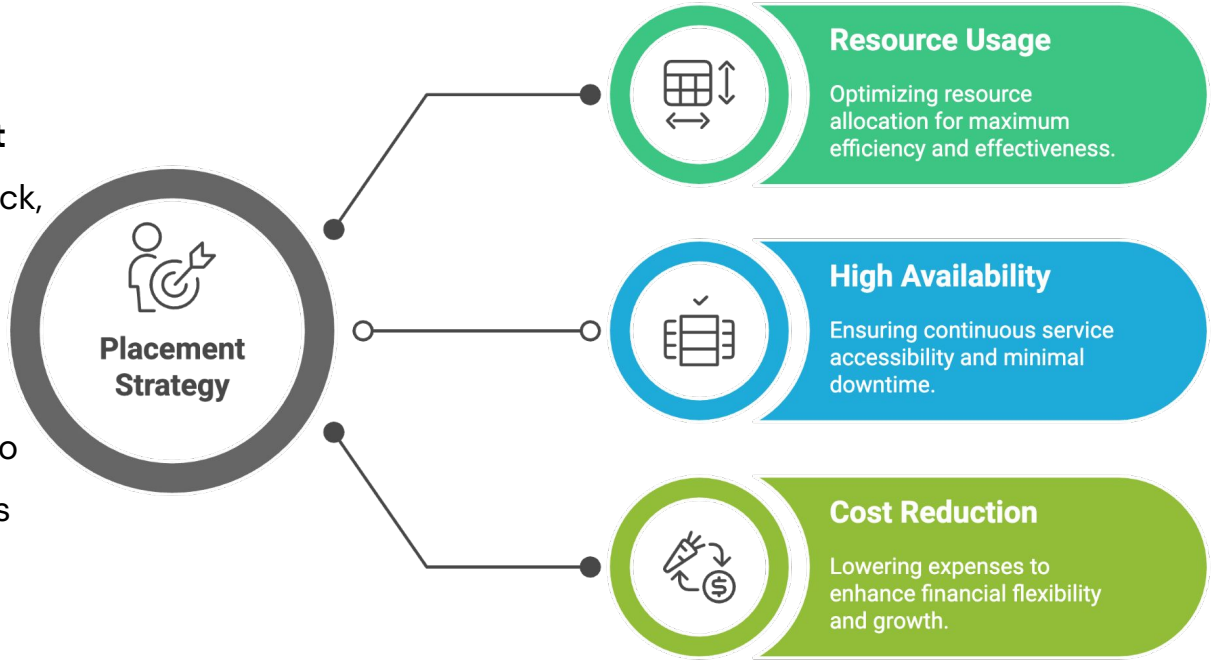
Amazon ECS

Task Placement Strategies

- Task Placement Strategies determine **how ECS schedules tasks** across your cluster's instances (for **EC2 launch type**).

How It Works:

- ECS evaluates **placement strategies** (spread, binpack, random)
- Combines them with **placement constraints** to choose where to run tasks



Strategy	Description	Best For
spread	Evenly distributes tasks across specified attributes (AZ, instance)	High availability
binpack	Places tasks on instances using **least available CPU/memory** first	Cost efficiency, resource consolidation
random	Randomly selects instances for placement	Load balancing without strict rules

ENI Limitations

Amazon ECS

ENI Limitations

1. Instance Type ENI Limits

- Each EC2 instance type has a **maximum number of ENIs** and **private IPv4 addresses** it can support.
- Limit = **Primary ENI (for the instance) + Secondary ENIs (for tasks)**.
- Example:
  - t3.small → 2 ENIs max
  - m5.2xlarge → 4 ENIs max

2. ECS Impact

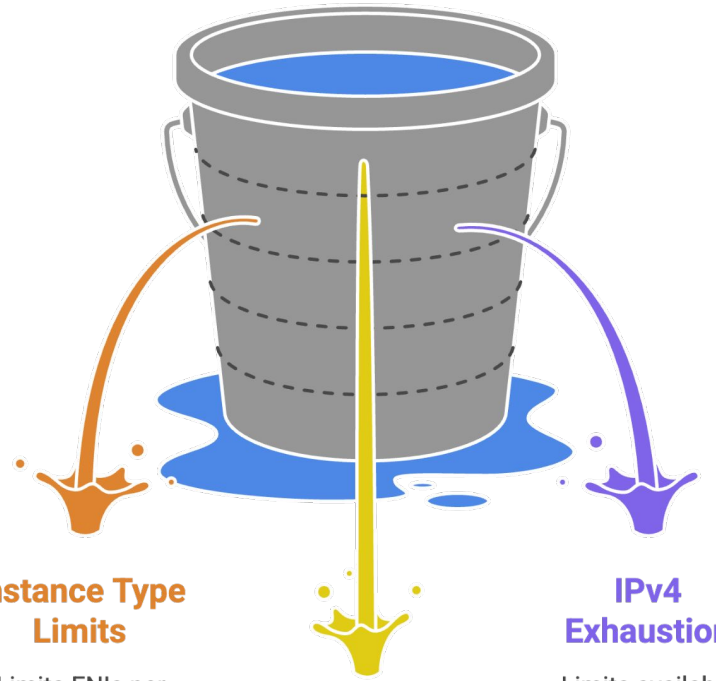
- In **awsvpc** mode, each task gets **its own ENI**.
- This **directly limits** how many tasks can run on a single EC2 instance.

3. IPv4 Address Exhaustion

- Even if ENI limit allows more, each ENI has a **max private IPs per ENI**.
- Once IPs are exhausted, no more tasks can be placed.

4. Workarounds

- Use **larger instance types** (higher ENI/IP limits)
- Scale out with **more EC2 instances**
- Consider **Fargate** to avoid ENI task-per-limit restrictions



Instance Type Limits

Limits ENIs per instance

ECS Impact

Limits task scale per instance

IPv4 Exhaustion

Limits available addresses for ENIs

Service Discovery

Amazon ECS

Service Discovery

- A mechanism that allows ECS services and tasks to **find and connect** with each other without manually managing IP addresses.
- Uses **AWS Cloud Map** for DNS-based service registration and discovery.

Why It's Needed:

- ECS tasks often have **dynamic IPs** (especially in awsvpc mode).
- Services need a **consistent way** to locate each other.

Service Discovery Benefits

Automatic DNS

Automatic DNS name assignment for services.

Multiple Services

Works seamlessly with multiple services and tasks.

No IP Management

Eliminates the need for manual IP address management.

Multi-AZ Connectivity

Provides service connectivity across multiple availability zones.

Workflow:

1. Enable **Service Discovery** when creating ECS Service.
2. ECS registers running tasks with **AWS Cloud Map**.
3. Other services use **DNS name** (e.g., myservice.example.local) to connect.
4. If tasks stop or move, DNS records automatically update.

Use Cases:

- Microservices communication
- Multi-service applications with dynamic scaling
- Internal service-to-service traffic in private VPC

Service Connect

Amazon ECS

Service Connect

- A feature of ECS that simplifies **service-to-service communication** within a cluster and across multiple clusters.
- Provides **built-in service discovery, connectivity, and secure communication** without extra networking configuration.

Why It's Needed:

- No need to manually set up **VPC peering, PrivateLink, or complex routing**.
- Allows ECS services in different VPCs or accounts to communicate securely.

Service Discovery

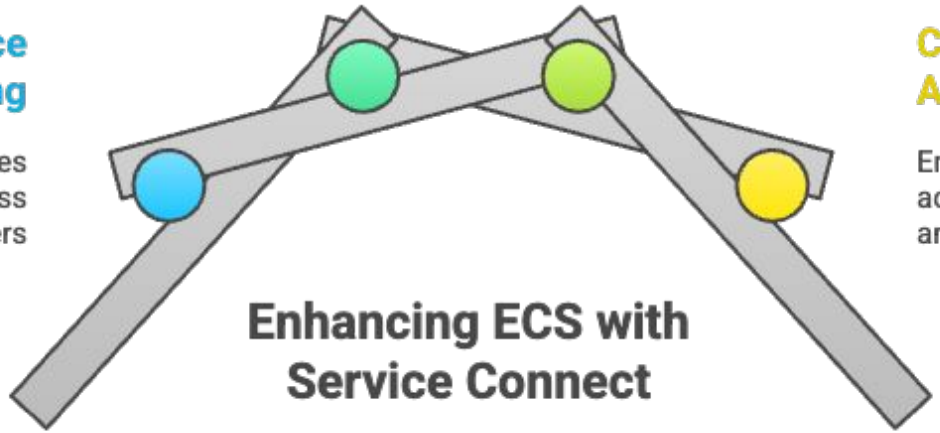
Uses names instead of IPs for service identification

Encrypted Traffic

Ensures secure data transmission by default

Simplified Service Networking

Streamlines communication across ECS clusters



Cross-VPC and Account Support

Enables connectivity across different VPCs and accounts

Enhancing ECS with Service Connect

Workflow:

1. **Enable Service Connect** when creating an ECS service.
2. ECS automatically registers the service with **service connect**.
3. Other ECS services can connect using **service names**.
4. Traffic is routed securely without manual networking setup.

Use Cases:

- **Microservices** spread across multiple VPCs
- **Multi-account ECS architectures**
- **Hybrid** service communication without complex networking

Service Discovery

vs

Service Connect

Amazon ECS

Feature	Service Discovery	Service Connect
Scope	Works **within one VPC/Cluster**	Works **across VPCs, subnets, accounts**
How It Works	Uses **Cloud Map DNS** names for service-to-service communication	Uses **App Mesh** to provide connectivity, discovery, and secure routing
Use Case	Simple **microservices** (e.g., frontend → backend in same VPC)	**Cross-VPC, multi-cluster, multi-account** architectures
Discovery Method	DNS-based lookup (e.g., backend.myapp.local)	Logical **service name** registered with Service Connect
Setup	Requires enabling **Cloud Map namespace**	Enabled when creating an **ECS Service with Service Connect**

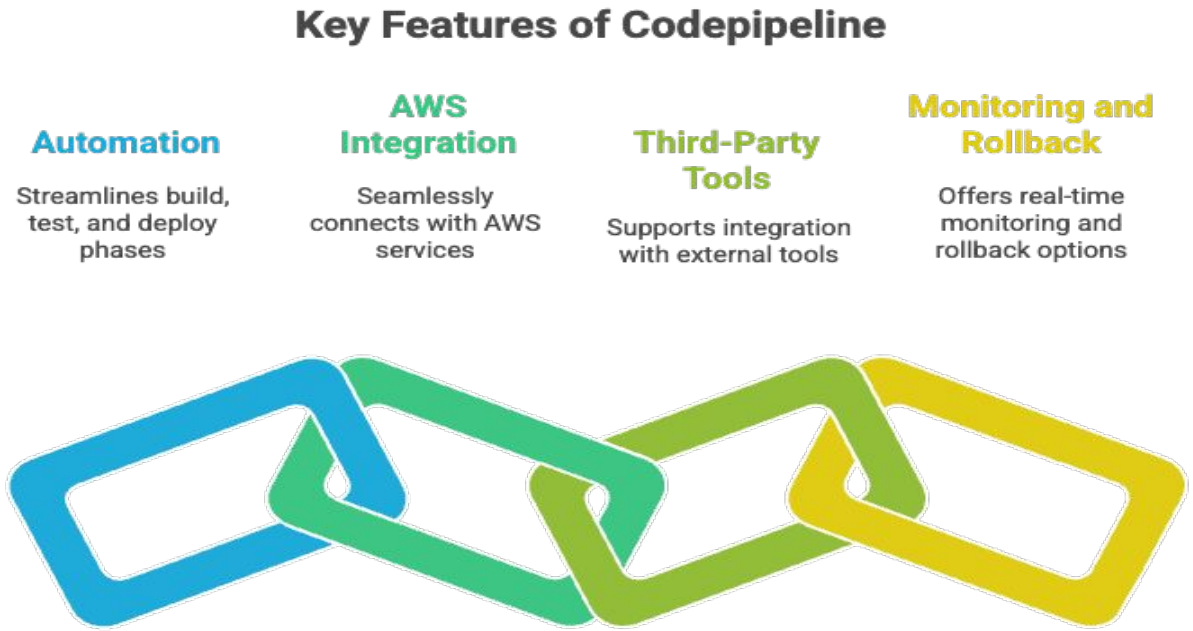
AWS DevOps

Codepipeline

Amazon ECS

AWS DevOps – Codepipeline

- A fully managed **continuous integration and continuous delivery (CI/CD)** service that automates release pipelines for fast and reliable updates.



Components:

1. Source

- Stores code (e.g., CodeCommit, GitHub, S3) and triggers pipeline when new changes are pushed.

2. Build

- Compiles code, runs tests which is usually powered by **AWS CodeBuild** or external tools (Jenkins).

3. Test (Optional)

- Automated testing to ensure quality before deployment.

4. Deploy

- Deploys to environments like **EC2, ECS, Lambda, Elastic Beanstalk, S3, CloudFormation**.

5. Stages & Actions

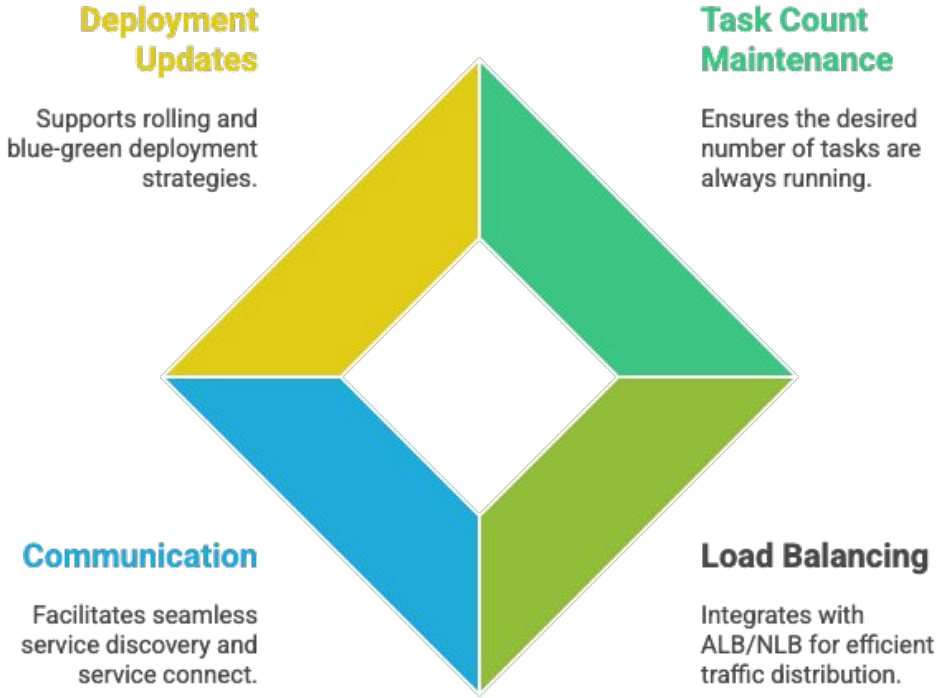
- A pipeline has **stages** (Source, Build, Deploy) and each stage contains **actions** (tasks to perform).

Services

Amazon ECS

Services

- A **long-running ECS task manager** that maintains the desired number of task instances.
- Ensures tasks are restarted if they fail or stop.



Benefits:

- High availability for containers
- Automated scaling (with CloudWatch alarms/Application Auto Scaling)
- Integration with AWS networking and monitoring

Use Cases:

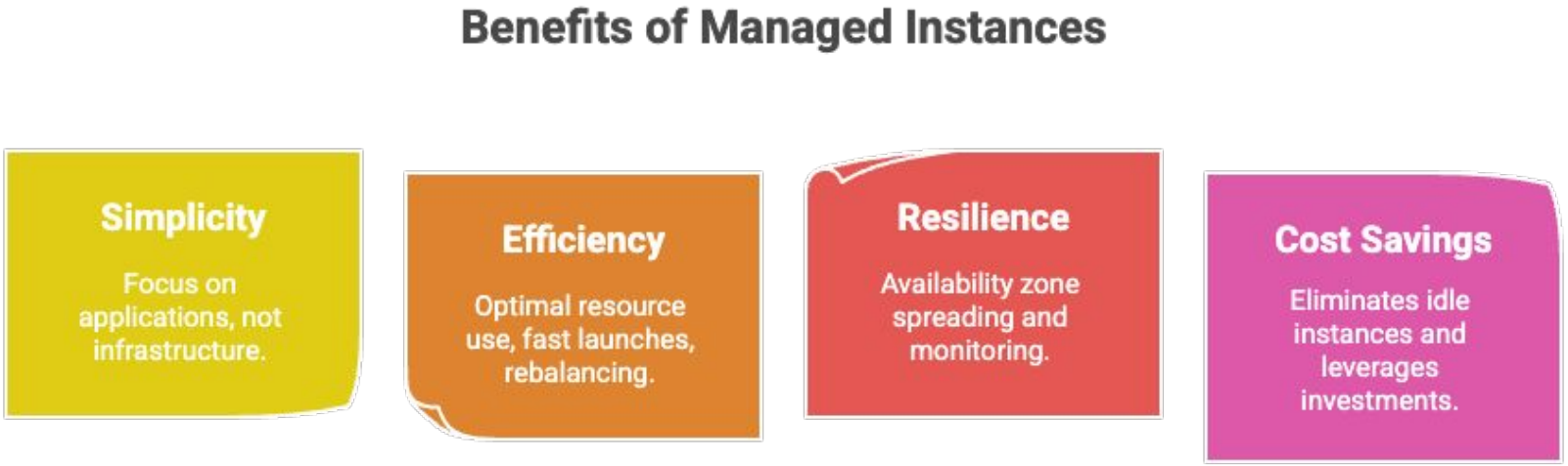
- Web applications behind a load balancer
- APIs and microservices that must run 24/7
- Multi-AZ distributed services for resilience

ECS Managed Instances

Amazon ECS

ECS – Managed Instances

- A fully managed compute option for Amazon ECS, blending Fargate's simplicity with EC2's flexibility—AWS handles provisioning, scaling, patching, and idle termination.



Infrastructure - *advanced* Info

Configure the manner of obtaining compute resources that will be used to host your application.

Select a method of obtaining compute capacity

Your cluster is automatically configured for AWS Fargate (serverless), but you may choose to add Amazon EC2 instances (servers).

- (•) **Fargate only**
  Serverless – you don't think about creating or managing servers. Great for most common workloads.
- ( ) **Fargate and managed instances**
  New
  Managed instances - Amazon ECS will manage patching and scaling on your behalf while giving you configurability about the types of instances. Great for more advanced workloads.
- ( ) **Fargate and Self-managed instances**
  Self-managed instances - you must ensure the instances are patched and scaled properly, and you have full control over the instances.

Core Features:

- Supports diverse EC2 types (GPU, ARM, high-network).
- Security: Bottlerocket OS with bi-weekly patches; integrates with Spot/Reserved Instances.