Amazon ECS and Container Orchestration

Instructor: Govind Kumar
Lecture 13 | AWS Cloud Hosting Workshop

ECS Architecture

Understand the core components and architecture of Amazon ECS.

Task Orchestration

Learn how ECS manages container deployment and scaling.

Networking & Security

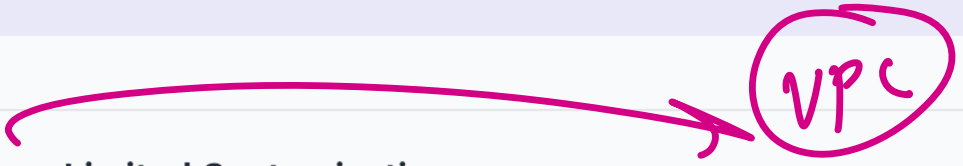
Explore networking options and security best practices for ECS.

Challenges with App Runner

When to Move Beyond App Runner

While AWS App Runner is great for simple containerized applications, it has limitations that may require moving to ECS:

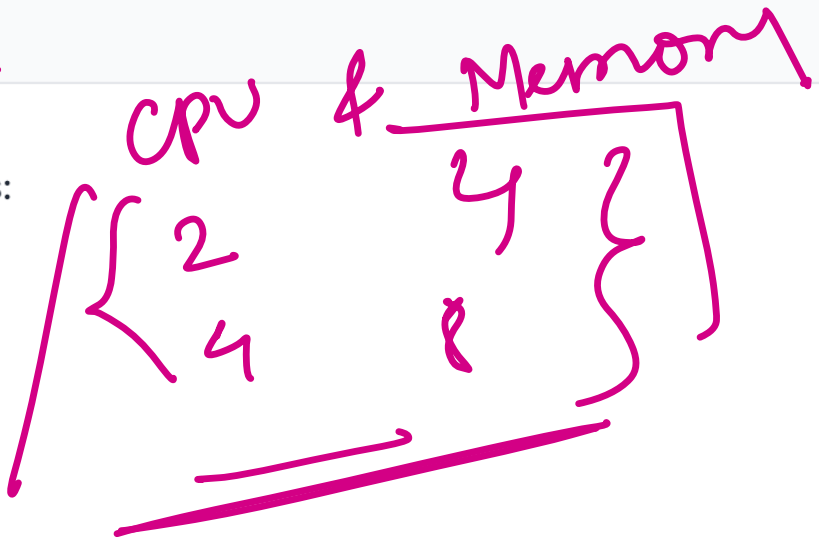
> Understanding these limitations helps you make informed decisions about when to use ECS instead of App Runner.



Scaling Limitations | **Networking Constraints** | **Limited Customization**

App Runner has constraints on scaling that may not meet complex application needs:

- Limited to 25 concurrent instances
- Less granular scaling controls
- No support for custom scaling metrics
- Fixed scaling thresholds



ECS Architecture | Launch Types | Deployment Options

Amazon ECS consists of several key components that work together to run your containerized applications.

Clusters

Logical grouping of infrastructure resources where your containers run

↓

Task Definitions

Blueprint for your application that defines containers, resources, and networking

↓

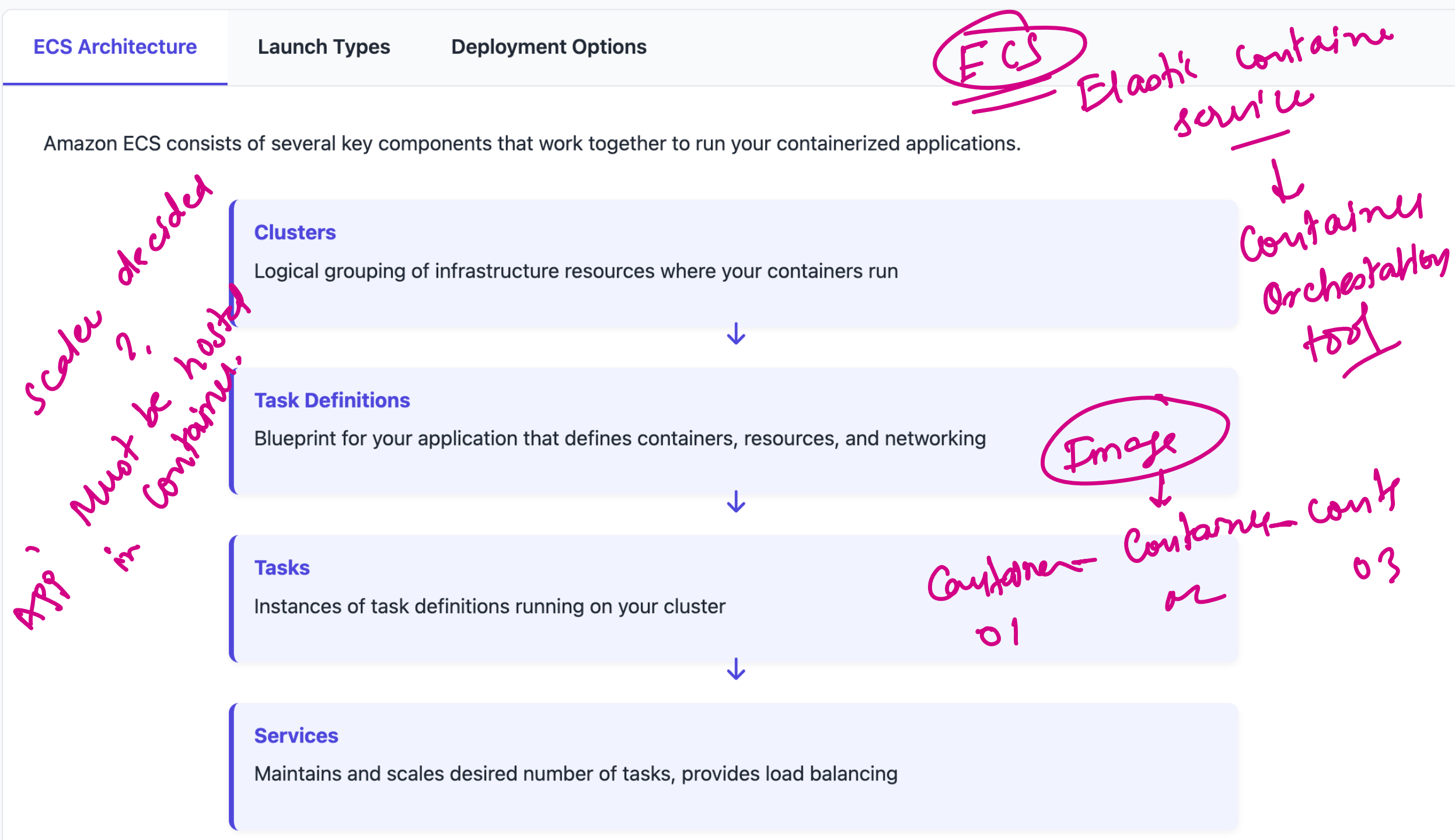
Tasks

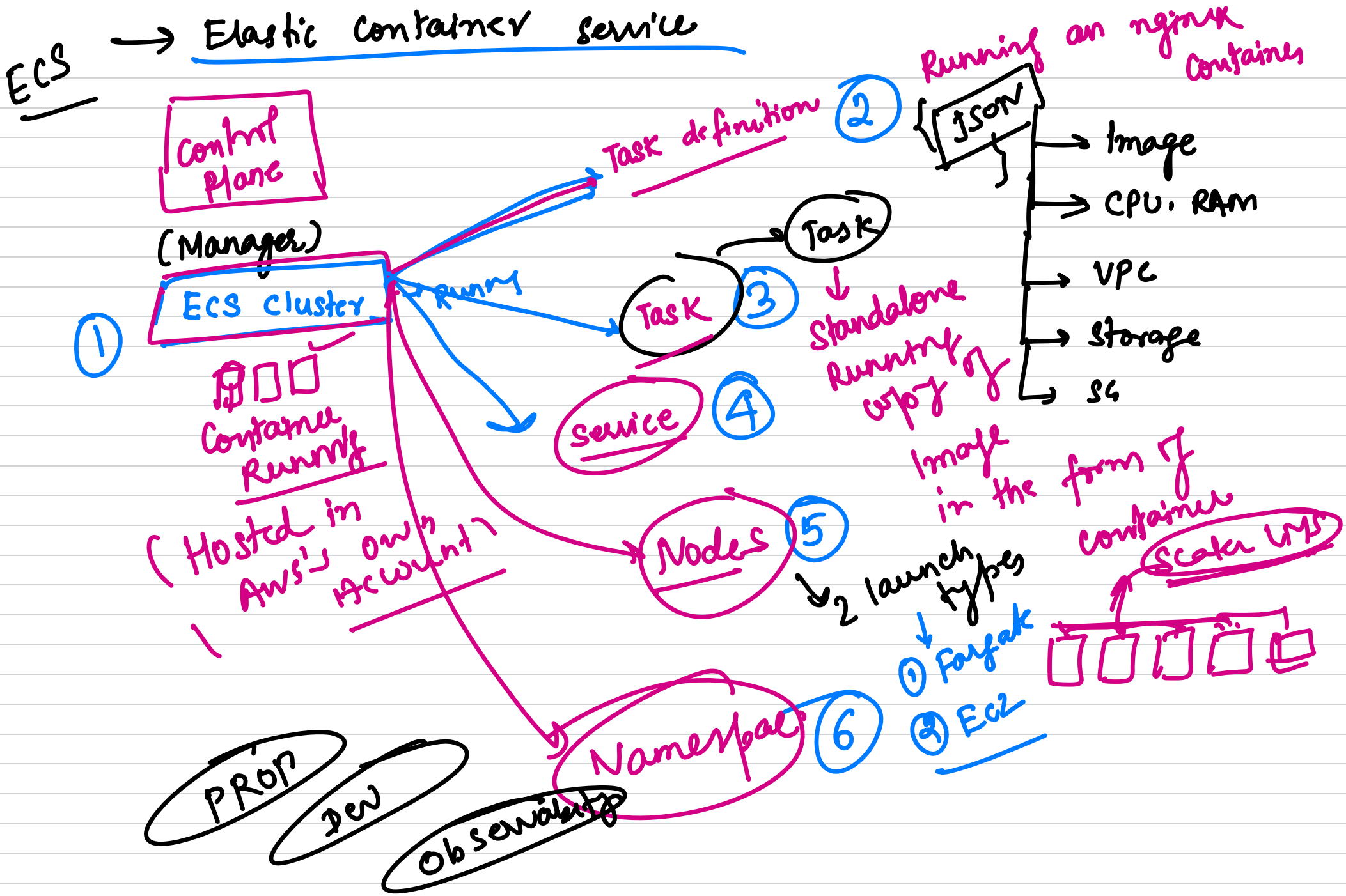
Instances of task definitions running on your cluster

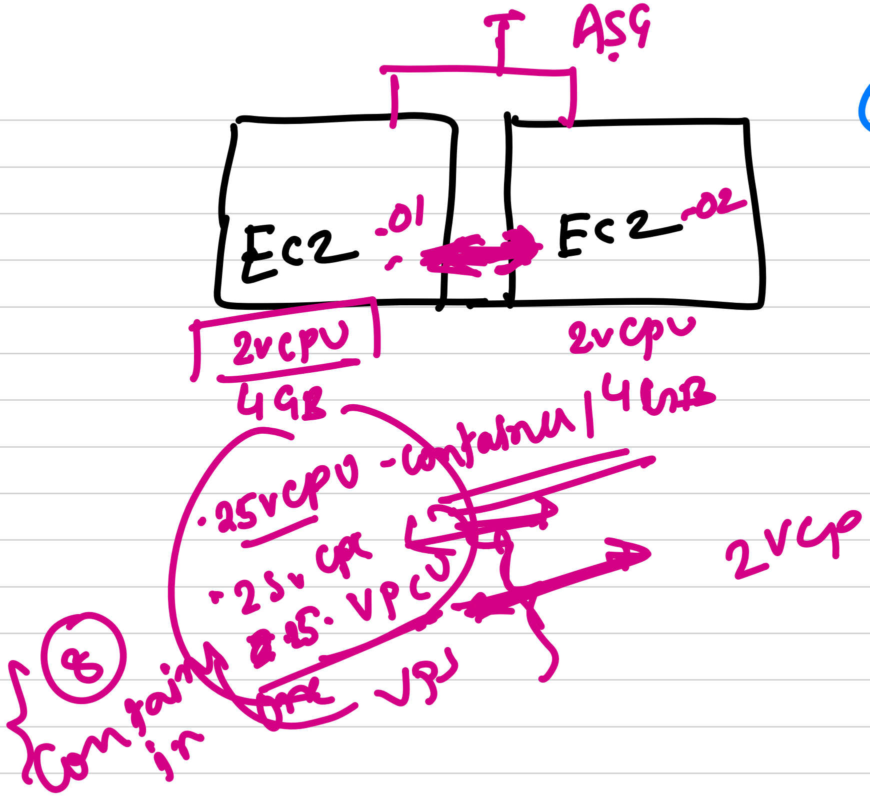
↓

Services

Maintains and scales desired number of tasks, provides load balancing




ECS → Elastic container service
Control Plane
(Manager)
ECS Cluster
1
Container Running
(Hosted in Aws's own account)
Task definition
2
Running an nginx Container
JSON
Image
CPU, RAM
VPC
Storage
SG
Task
Task
3
Standalone Running copy of
Running
Service
4
Nodes
5
Image in the form of container
Scale UP
2 launch types
1 Fargate
2 EC2
Namespace
6
PROD
DEV
Observability



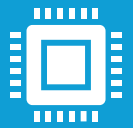
Rep

Service Autoscaling

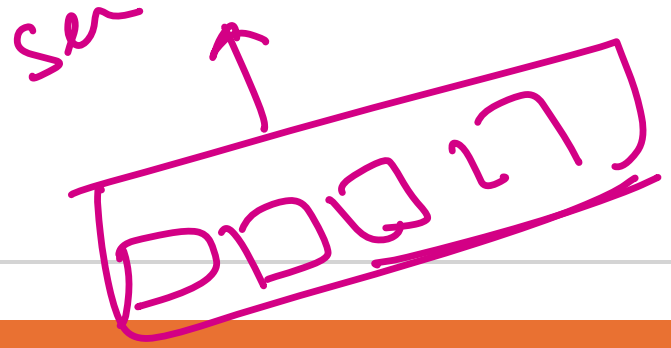
Cluster Autoscaler

Cluster
A logical grouping of tasks or services that run on infrastructure resources (EC2 instances or Fargate).
It provides isolation between different workloads and can span multiple Availability Zones within a region.

Task Definition
A blueprint that describes how a container should be launched, including image, CPU, memory, ports, and volumes.
It's an immutable specification that can be versioned and used to run individual tasks or as part of a service.



Fargate
A serverless compute engine for containers that eliminates the need to provision and manage servers.
It allows you to focus on designing and building applications instead of managing the infrastructure that runs them.



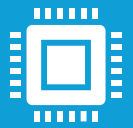
Service
A configuration that maintains a specified number of task instances running simultaneously and can automatically replace failed tasks.
It provides features like load balancing, auto-scaling, and rolling updates for long-running applications.

Task
A running instance of a task definition, representing the instantiation of containers defined in the task definition.
Tasks are ephemeral and can be created, stopped, and terminated as needed based on your application requirements.

Fargate
A serverless compute engine for containers that eliminates the need to provision and manage servers.
It allows you to focus on designing and building applications instead of managing the infrastructure that runs them.

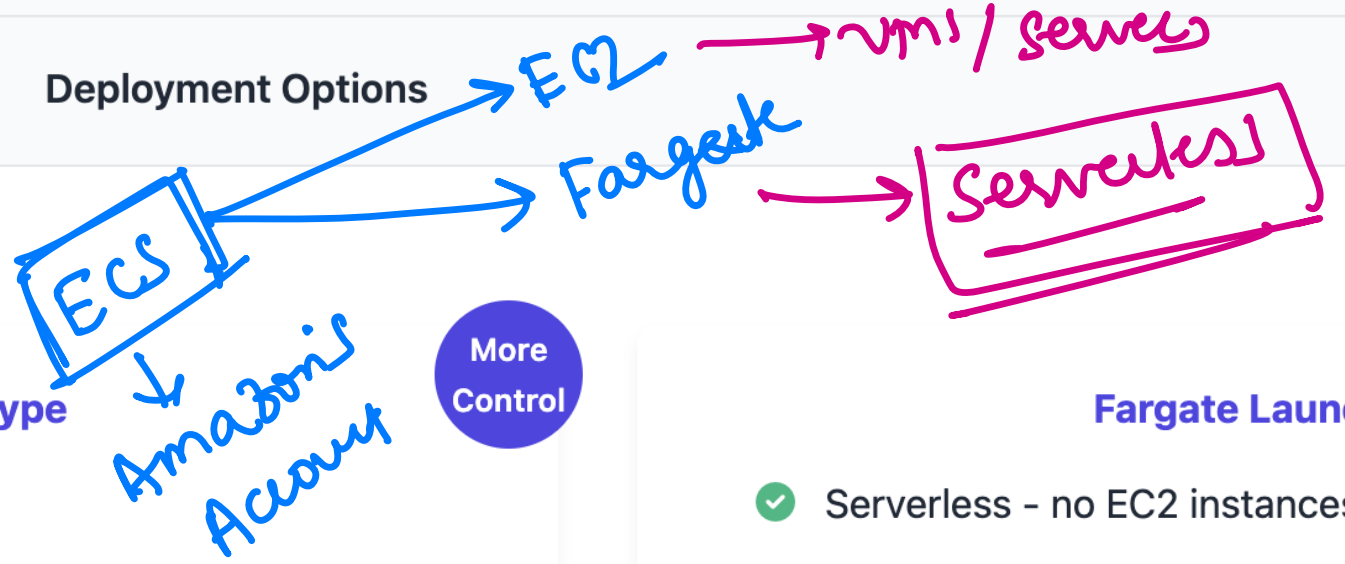
Namespace
A logical boundary that isolates resources and enables service discovery within AWS Cloud Map.
It allows containers to find and connect to each other using DNS or API calls without hardcoded IPs.

Nodes
The EC2 instances that host your containers and are part of your ECS cluster.
They run the ECS container agent and are responsible for starting and stopping tasks as instructed by the ECS service.



Fargate
A serverless compute engine for containers that eliminates the need to provision and manage servers.
It allows you to focus on designing and building applications instead of managing the infrastructure that runs them.

ECS Architecture | Launch Types | Deployment Options



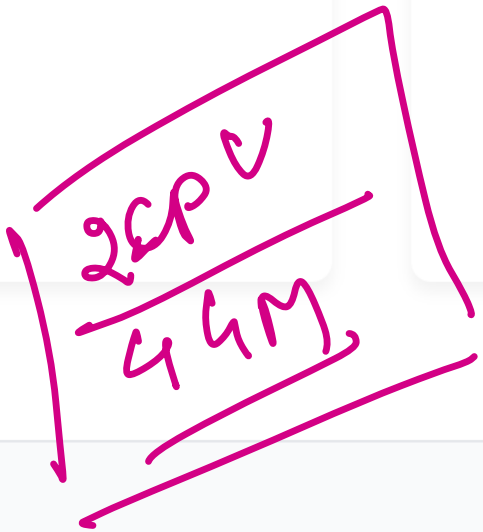
More Control

EC2 Launch Type

- You manage EC2 instances
- More granular control over infrastructure
- Can use Spot Instances for cost savings
- Support for custom AMIs
- Access to instance-level features

Fargate Launch Type

- Serverless - no EC2 instances to manage
- Pay only for resources used by containers
- Simplified operations and maintenance
- Automatic scaling with no infrastructure
- Enhanced security isolation



ECS Architecture | Launch Types | Deployment Options

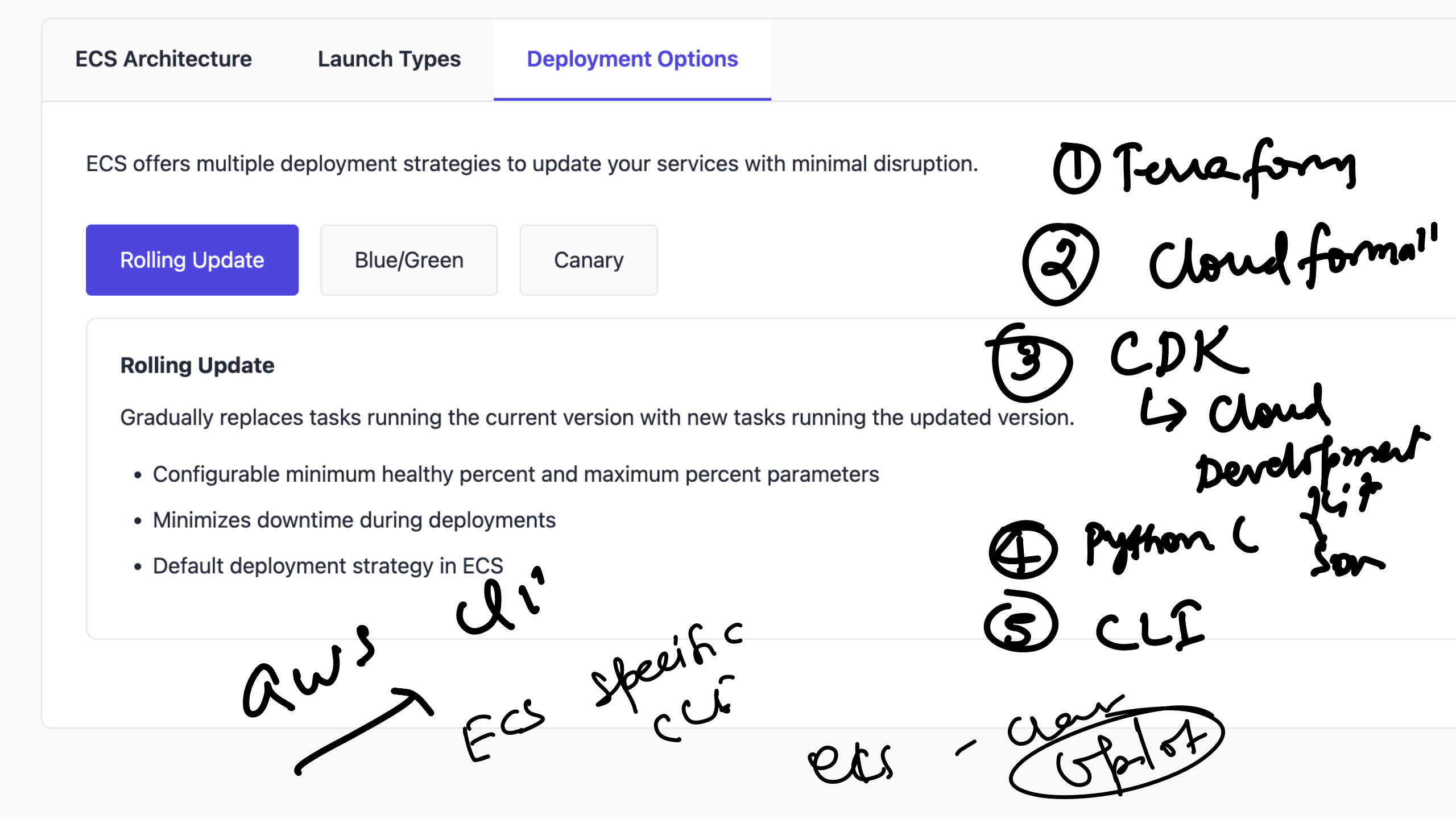
ECS offers multiple deployment strategies to update your services with minimal disruption.

Rolling Update | Blue/Green | Canary

Rolling Update

Gradually replaces tasks running the current version with new tasks running the updated version.

- Configurable minimum healthy percent and maximum percent parameters
- Minimizes downtime during deployments
- Default deployment strategy in ECS



ECS Architecture | Launch Types | Deployment Options

ECS offers multiple deployment strategies to update your services with minimal disruption.

Rolling Update | Blue/Green | Canary

Blue/Green Deployment

Runs two identical environments (blue and green) and switches traffic all at once.

- Implemented using AWS CodeDeploy
- Allows for easy rollback if issues are detected
- Eliminates downtime during deployments

ECS Architecture | Launch Types | Deployment Options

ECS offers multiple deployment strategies to update your services with minimal disruption.

Rolling Update | Blue/Green | Canary

Canary Deployment

Shifts small percentage of traffic to new version before full deployment.

- Implemented using AWS CodeDeploy
- Allows testing with a subset of users
- Configurable traffic shifting percentages and intervals