



Amazon ECS in Depth

Amazon ECS Developer Guide (as of 2026) | clusters, task definitions, services, launch types and capacity providers, network modes, Service Connect, load balancing, deployments, IAM roles, secrets, logging, storage, auto scaling, troubleshooting

ECS Fargate DVA-C02 SAA-C03 DOP-C02

The ECS object model

Object	What it is	Key settings
Cluster	regional pool of capacity	capacity providers, default strategy, Container Insights, Service Connect namespace
Task definition	versioned blueprint (family:revision)	containers, image, cpu/memory (task and container level), networkMode, roles, volumes, logConfiguration, runtimePlatform (X86_64 or ARM64, Linux or Windows)
Task	running instantiation of a task definition	standalone (RunTask), scheduled (EventBridge Scheduler) or owned by a service
Service	keeps N tasks running	desiredCount, deployment config, load balancers, placement, auto scaling, Service Connect
Container instance	EC2 (or external) host registered by the ECS agent	instance profile ecsInstanceRole, ECS-optimized AMI (AL2023, Bottlerocket)

Capacity and networking

Launch option	Capacity	Notes
Fargate	FARGATE / FARGATE_SPOT capacity providers	no instances; per-task micro-VM; Spot tasks get a 2-minute SIGTERM warning
EC2	ASG capacity provider with managed scaling (target capacity %) and managed termination protection	GPUs, daemon services, custom AMLs; you patch the instances
ECS Managed Instances (2025)	AWS provisions and patches EC2 capacity for you	EC2 instance choice with less operations; check Region availability
External (ECS Anywhere)	your servers with SSM + ECS agents	hybrid; no load balancer integration or awsvpc

- **Capacity provider strategy** base = tasks always on the first provider; weight = split of the rest, e.g. FARGATE base 2 weight 1 + FARGATE_SPOT weight 3 → after 2 on-demand tasks, 75 % on Spot
- **Placement (EC2)** strategies binpack (fewest instances), spread (by AZ or instance), random; constraints distinctInstance, memberOf (cluster query language)
- **Network modes** awsvpc (ENI per task, SG per task, target type ip), bridge (Docker bridge, dynamic host ports with ALB), host (container uses the host's ports), none
- **Service-to-service** Service Connect (Cloud Map namespace + managed Envoy, retries, metrics, optional TLS with Private CA), Cloud Map DNS service discovery, or internal ALB; App Mesh reaches end of support on 30 September 2026

taskdef.json (abridged)

```
1 {"family": "orders-api",
2   "requiresCompatibilities": ["FARGATE"],
3   "networkMode": "awsvpc",
4   "cpu": "1024", "memory": "2048",
5   "runtimePlatform": {"cpuArchitecture": "ARM64"},
6   "taskRoleArn": "arn:aws:iam::111:role/orders-task",
7   "executionRoleArn": "arn:aws:iam::111:role/orders-exec",
8   "containerDefinitions": [{
9     "name": "api",
10    "image": "111.dkr.ecr.eu-west-3.amazonaws.com/orders:1.4.2",
11    "portMappings": [{"containerPort": 8080}],
12    "secrets": [{"name": "DB_PASSWORD",
13                 "valueFrom": "arn:aws:secretsmanager:::db-XYZ"}],
14    "logConfiguration": {"logDriver": "awslogs",
15                         "options": {"awslogs-group": "/ecs/orders"},
16                         "healthCheck": {"command": ["CMD-SHELL",
17                                                "curl -f localhost:8080/health || exit 1"]}}}]}
```

- 1 Fargate requires awsvpc and task-level cpu/memory
- 2 1 vCPU, 2 GB: a valid Fargate combination
- 3 Graviton: cheaper per vCPU; image must be arm64 or multi-arch
- 4 permissions of the app code
- 5 pull image, read the secret, write logs
- 6 pin a tag or digest, never :latest
- 7 injected as env var at task start, not refreshed

Y

Without ENI trunking, how many awsvpc tasks of 0.25 vCPU fit on an m5.xlarge (4 vCPU, 4 ENIs, 15 IPv4 addresses per ENI)? What changes with trunking?

Given: ENIs E = 4 (one is the primary ENI of the instance); awsvpc task = 1 ENI each; trunking: each supported Nitro type has a branch-ENI limit listed in the ECS ENI trunking table (check yours)

Y

Step 1: ENIs free for tasks?

$E - 1 = 4 - 1 = 3$
the primary ENI belongs to the instance

T

Y

Step 2: Tasks without trunking?

3 tasks
each awsvpc task needs its own ENI; IPs per ENI do not help

T

Y

Step 3: Tasks with trunking?

up to the branch-ENI limit of the type (one trunk ENI carries many branch ENIs)
awsvpcTrunking account setting, Nitro instance

T

Y

Step 4: Would CPU allow it?

$4 \text{ vCPU} / 0.25 \text{ vCPU per task} = 16$
trunking removes the ENI limit, CPU/memory now decide

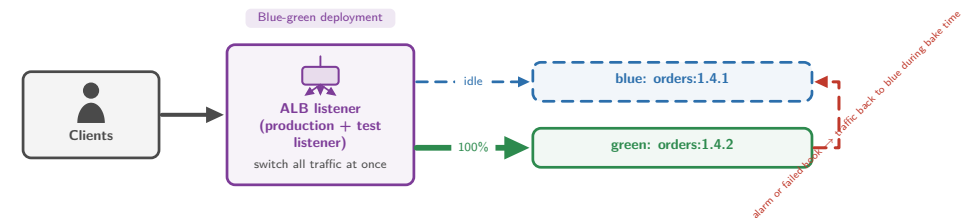
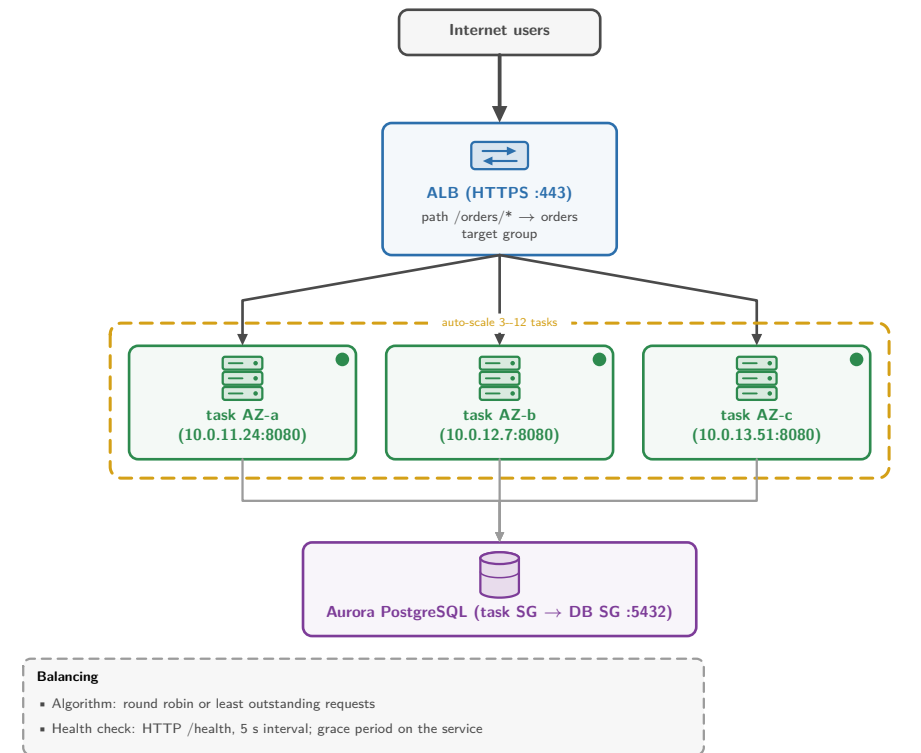
T

3 tasks without trunking; with trunking the branch-ENI limit or CPU (16 small tasks) decides

Check: Bridge mode avoids the ENI limit but loses per-task security groups

T

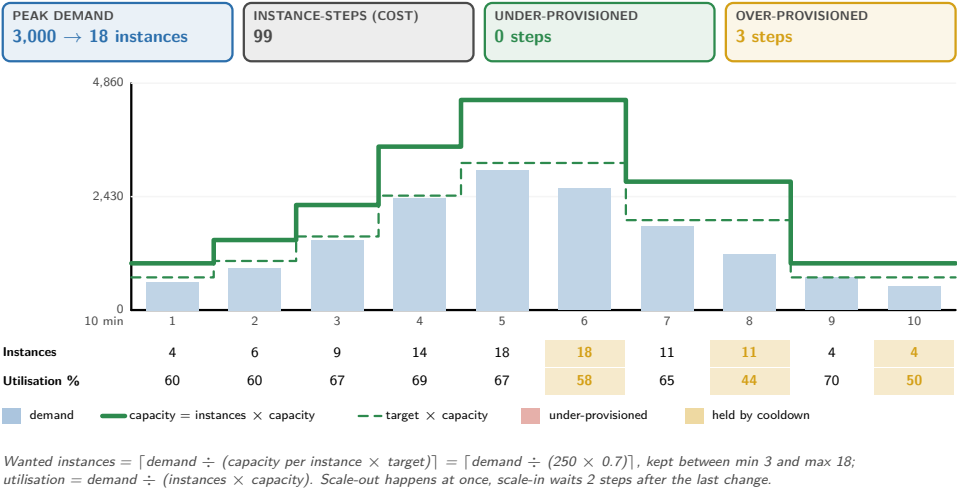
Load balancing and deployments



Deployment type	How it works	Rollback
Rolling (ECS)	replaces tasks within minimumHealthyPercent / maximumPercent (e.g. 100 / 200)	deployment circuit breaker with rollback; CloudWatch alarm-based rollback
Blue/green (ECS native, 2025)	new task set behind a test listener, lifecycle hooks (Lambda), traffic shift, bake time	instant switch back during bake time
Linear and canary (ECS native)	shift traffic in steps	automatic on alarms; check feature availability
Blue/green with CodeDeploy	deployment controller CODE_DEPLOY; canary, linear or all-at-once; appspec hooks	automatic on alarms or failed hooks
External controller	third-party tools manage task sets	tool-specific

⚙️ **Secrets, storage, logs, scaling**

- **Secrets** secrets → Secrets Manager or SSM Parameter Store, read by the execution role at start; rotate = redeploy (or read at runtime with the SDK and the task role)
- **Storage** Fargate ephemeral storage 20 GiB default, up to 200 GiB; EFS volumes (shared, multi-AZ, access points, IAM authorisation); EBS volumes attached per task (configured at deployment); bind mounts
- **Logs** awslogs driver to CloudWatch Logs; FireLens sidecar (Fluent Bit) to S3, Firehose, OpenSearch, Datadog...; Container Insights with enhanced observability
- **Service auto scaling** Application Auto Scaling: target tracking on ECSServiceAverage CPUUtilization, MemoryUtilization or ALBRequestCountPerTarget; step and scheduled scaling; predictive scaling
- **Cluster scaling (EC2)** capacity provider managed scaling keeps the ASG at the target capacity % of what tasks need



Target tracking on requests per task: scale-out at once, scale-in only after the cooldown (tasks = $\lceil \text{demand} / (250 \times 0.7) \rceil$)

Symptom	Likely cause	Fix
Task stopped: CannotPull ContainerError	no route to ECR (private subnet without NAT or ecr.api, ecr.dkr, S3 endpoints) or execution role lacks ECR permissions	add endpoints / NAT, fix role
ResourceInitializationError: unable to retrieve secret	execution role lacks secrets-manager:GetSecretValue / kms:Decrypt, or no endpoint	fix execution role and network
Tasks killed by the ALB	health check fails before the app starts	healthCheckGracePeriodSeconds, correct path and port
Service stuck at desired count	no capacity: ENIs, CPU, memory, subnet IPs, Fargate Spot capacity	trunking, bigger subnets, capacity provider strategy
Exit code 137	container OOM-killed (memory hard limit)	raise memory or fix the leak

🚩 Check, recap, self-test

Statement	True	False
1. Fargate tasks can use the bridge network mode → Fargate requires awsvpc.	☐	☒
2. The task execution role is used to pull images and inject secrets → The task role is for the application code.	☒	☐
3. Secrets injected through secrets update automatically after rotation → They are read at task start; redeploy or read at runtime.	☐	☒
4. A capacity provider strategy can mix FARGATE and FARGATE_SPOT → Use base and weight to keep some tasks on on-demand.	☒	☐
5. ECS Anywhere tasks can register with an ALB → External instances have no load balancer integration.	☐	☒

Tick a box only when you could explain the point to someone else

☐ Two roles per task execution (agent) and task (app)	☐ awsvpc gives each task an ENI and SG trunking for density on EC2
☐ Capacity providers base + weight, Spot for savings	☐ Safe deployments circuit breaker, alarms, blue/green with bake time
☐ Scale tasks and capacity Application Auto Scaling + managed scaling	

- Q1 Which IAM role does ECS use to fetch a Secrets Manager secret into an environment variable?**
A The task execution role.

- Q2 Network mode required for Fargate?**
A awsvpc.

- Q3 Run 25 % of tasks on Fargate Spot but always keep 2 on on-demand?**
A Capacity provider strategy: FARGATE base 2 weight 3, FARGATE_SPOT weight 1.

- Q4 Automatically roll back an ECS rolling deployment whose tasks keep failing?**
A Enable the deployment circuit breaker with rollback (and alarm-based rollback).

- Q5 Send container logs to S3 and OpenSearch at once?**
A FireLens with Fluent Bit outputs.

- Q6 Service-to-service names, retries and metrics without your own mesh?**
A ECS Service Connect.

- Q7 Scale an ECS service on requests per task?**
A Target tracking on ALBRequestCountPerTarget via Application Auto Scaling.

- Q8 Run a container every night at 02:00 without a service?**
A EventBridge Scheduler (or a scheduled rule) running an ECS task.

- Q9 What does exit code 137 usually mean?**
A The container was OOM-killed (SIGKILL) after hitting its memory limit.