Intro to Amazon ECS

Amazon EC2 Container Service

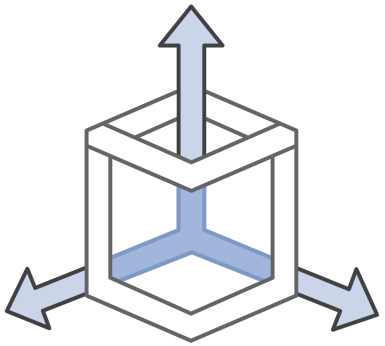
Amazon EC2 Container Service (ECS)

Highly scalable, high performance container management system.

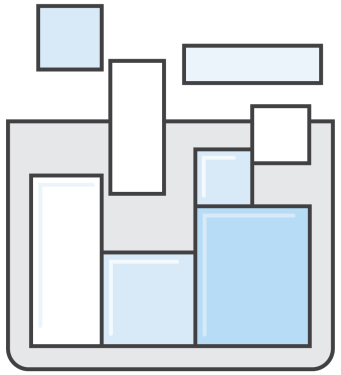
Eliminates the need to install, operate, and scale your own container management infrastructure.

Amazon EC2 Container Service (ECS)

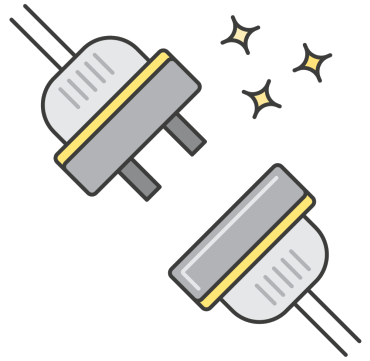
ECS provides a managed platform for:



Cluster management

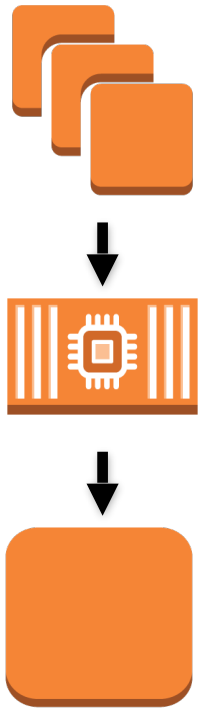


Container orchestration



Deep AWS integration

How does ECS map to traditional workloads?

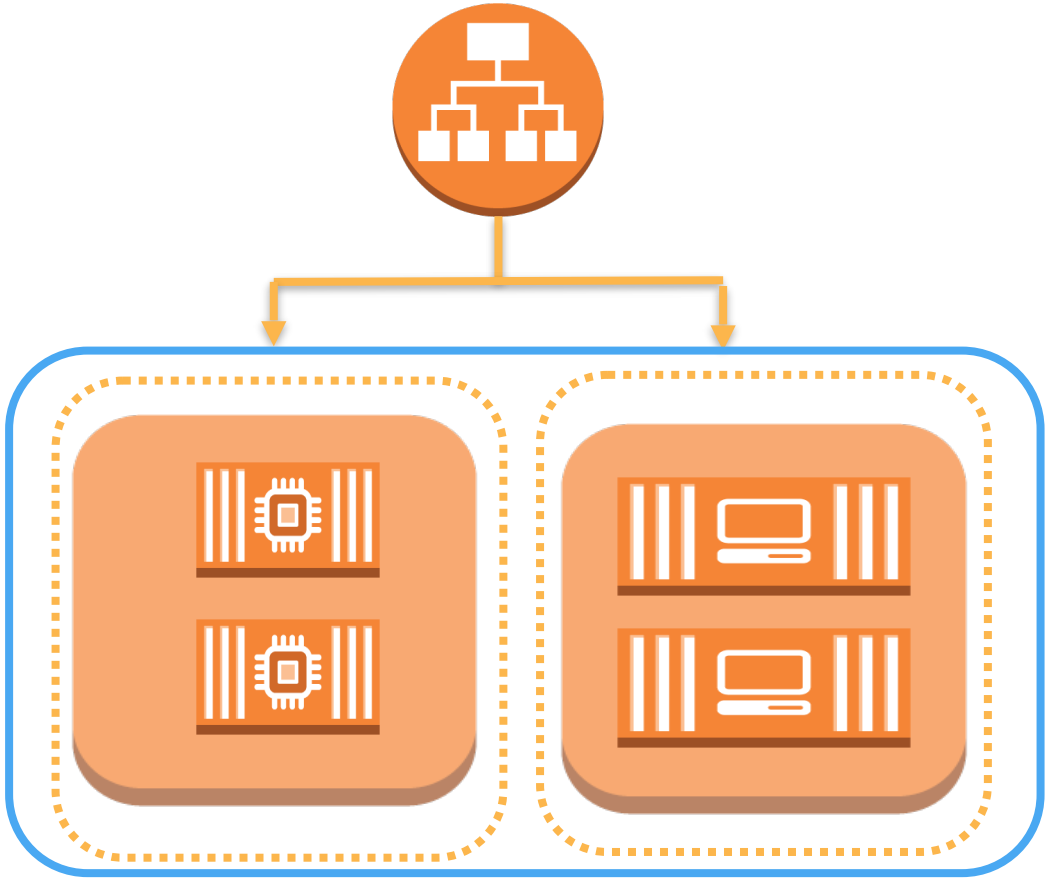


Instances: standard EC2 boxes. Once registered to a Cluster, your Tasks run here

Services: layer that manages and places Tasks

Tasks: container wrapper and configuration around processes running on the instance

How does ECS work?



Load balancer: (ALB or EC2 classic) routes traffic to the cluster instances.

Cluster is made up of one or more EC2 instances

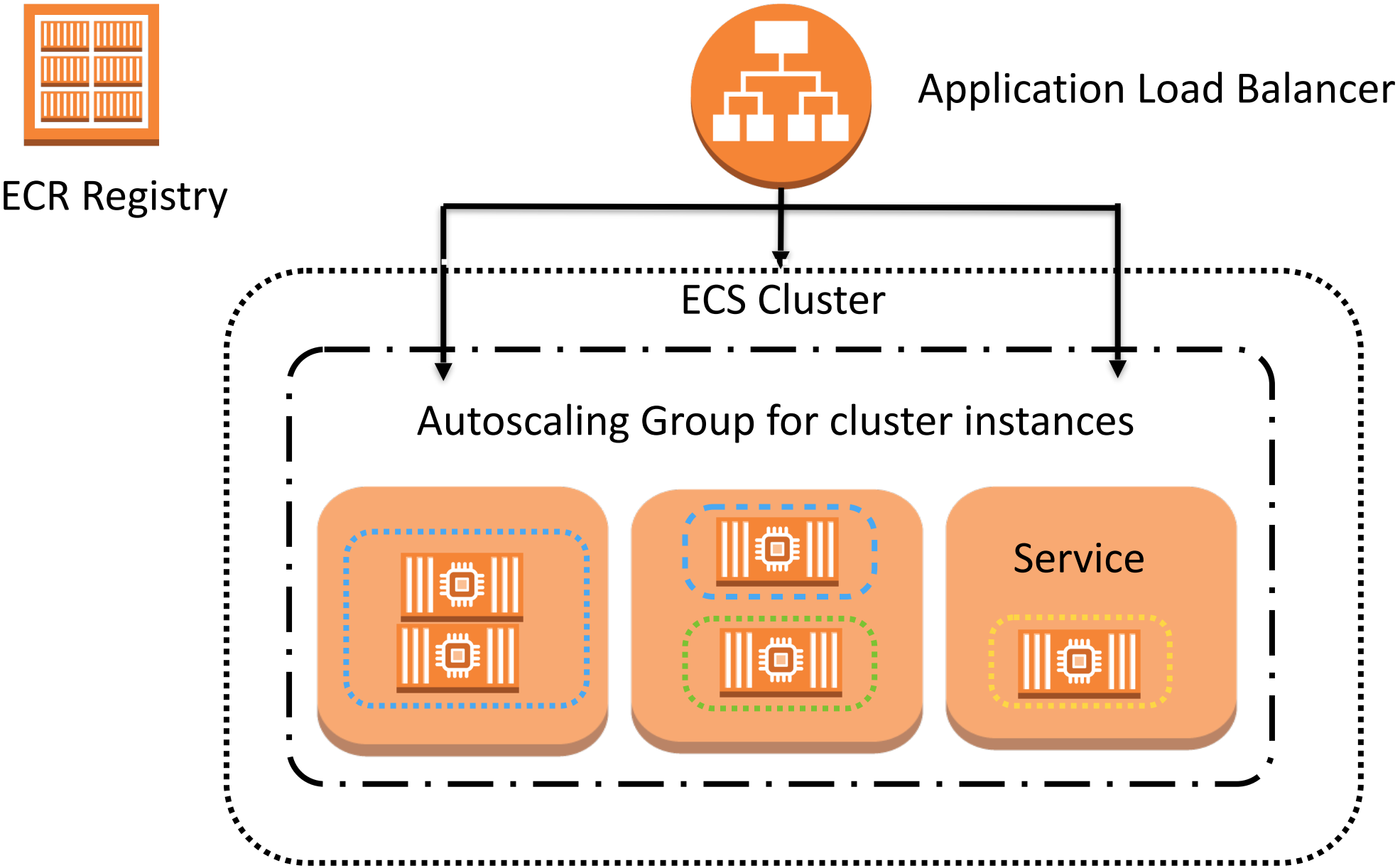
Each **cluster instance** runs one or more **Services**

How does ECS work?

Each **cluster instance** runs one or more **Services**

A **Service** controls things like the number of copies of a Task you want running (Desired Count), and registers your Service with a load balancer

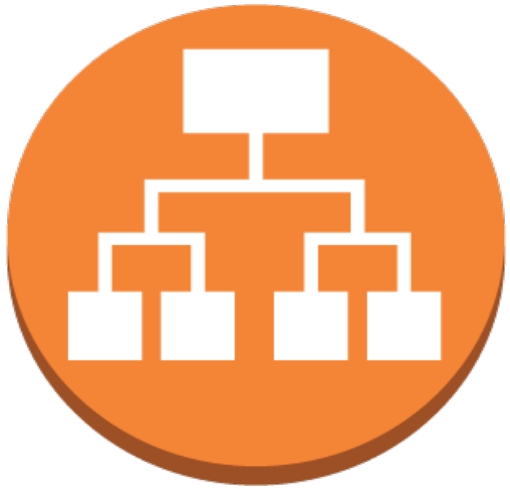
A **Task Definition** controls things like container image, environment variables, resource allocation, logger, and other parameters


ECR Registry
Application Load Balancer
ECS Cluster
Autoscaling Group for cluster instances
Service

Let’s talk about ALB

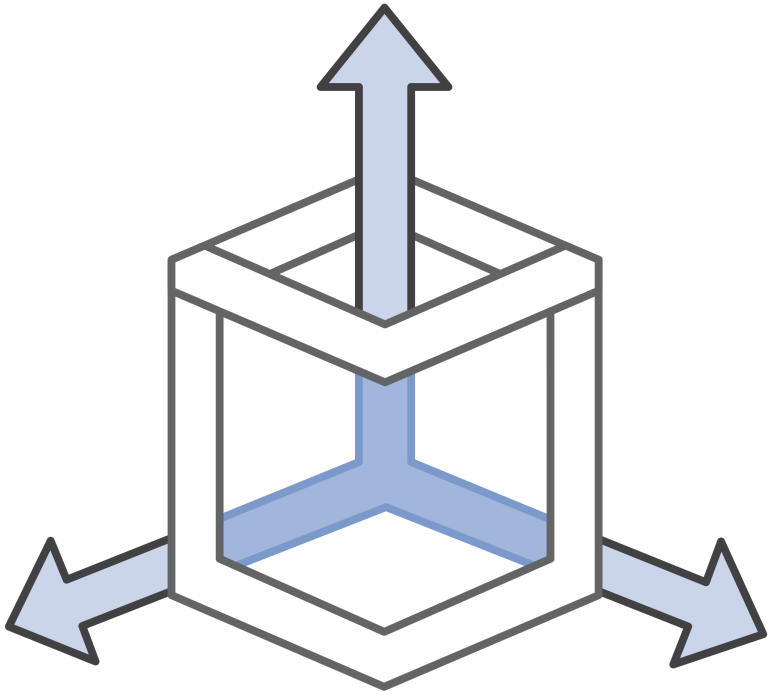
- **Define routing rules based on content**. Fancy way of saying “send traffic to different services based on endpoint”. This is magical.

- As a bonus, this allows ECS to allocate ports dynamically rather than statically, and one ALB can handle multiple services.



Why ECS?

Bottom line: containers and microservices can require a lot of orchestration and moving pieces. ECS removes a lot of this heavy lifting.

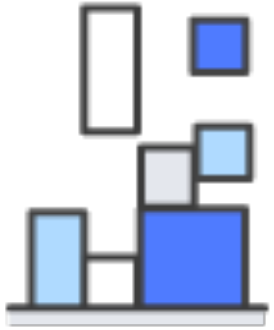


Who is using ECS?

…and many more!

Let’s get (feature) specific

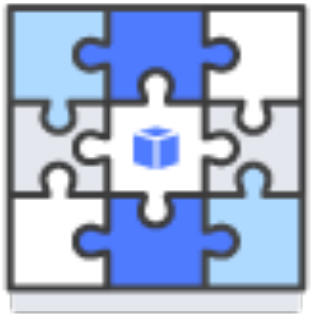
A few features, but many more.



Amazon ECS Task Placement

IAM Roles for Tasks

Flexible scaling for performance

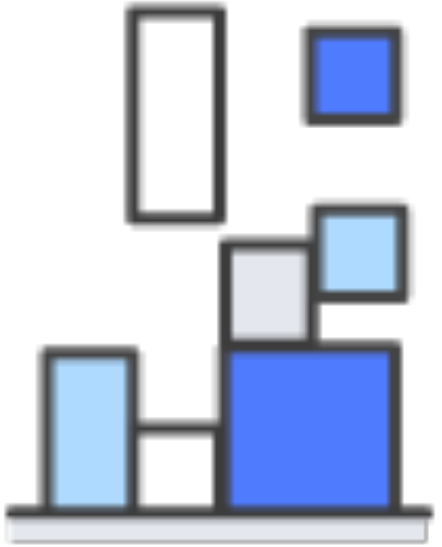


Amazon ECS Event Stream for Cloudwatch Logs

Fast, hassle-free deployments

Amazon ECS Task Placement

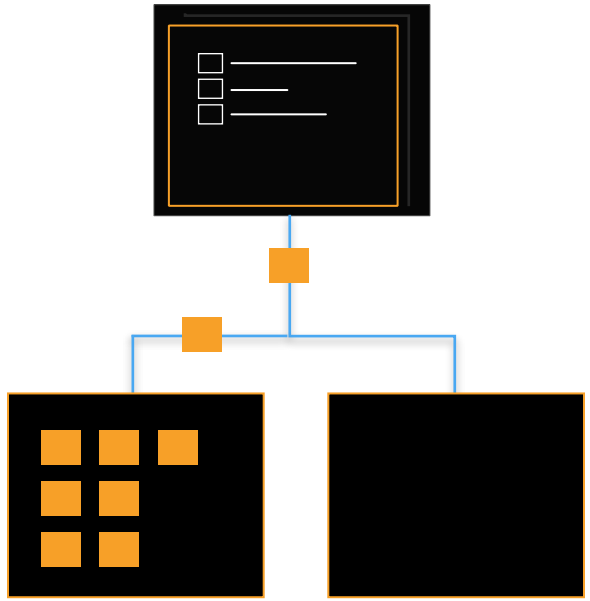
- A *task placement strategy* is an algorithm for selecting instances for task placement, or tasks for termination
- A *task placement constraint* is a rule taken into consideration during task placement
- Strategies and constraints can be used together



How can strategies and policies be used?

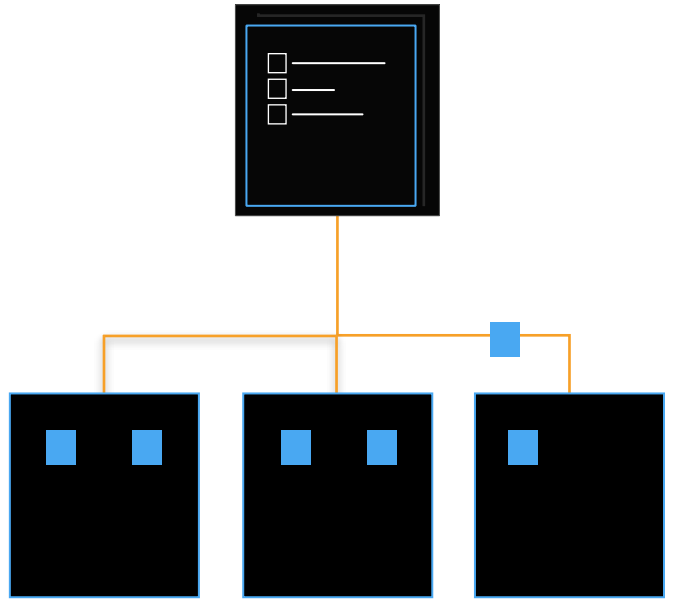
Name	Example
AMI ID	attribute:ecs.ami-id == ami-eca289fb
Availability Zone	attribute:ecs.availability-zone == us-east-1a
Instance Type	attribute:ecs.instance-type == t2.small
Distinct Instances	type="distinctInstances"
Custom	attribute:stack == prod

Multiple strategies are supported



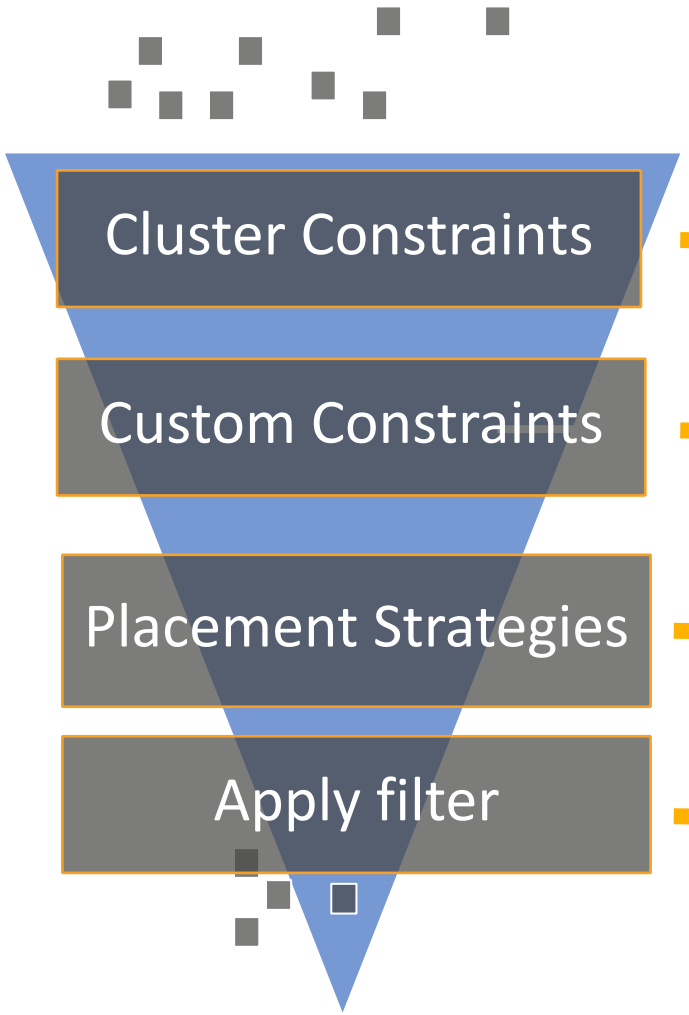
Binpacking

Random



Spread

How it works



Satisfy CPU, memory, and port requirements

Filter for location, instance-type, AMI, or custom attribute constraints

Identify instances that meet spread or binpack placement strategy

Select final container instances for placement

Amazon ECS Event Stream for Cloudwatch Logs

- Receive near real-time updates about both the current state of both the container instances within the ECS Cluster, and the current state of all tasks running on those container instances.
- Can be used to build custom schedulers, or to monitor cluster state and handle those state changes by consuming events with other AWS services, such as Lambda.

IAM Roles for ECS Tasks

- **Specify an IAM role used by the containers in a task.**
- Credential Isolation: containers can only access the role for the specific task that they are assigned to.
- **Authorization:** Unauthorized containers cannot access IAM role credentials defined for other tasks.
- **Auditability:** Audit through CloudTrail. Can track the Task credentials taskARN to show which task is using which role.

Fast, hassle-free deployments

- **Services deploy and scale quickly**. Very easily extensible through API calls; for example, trigger a deployment based on a commit to a branch on Github through your CI tool.
- **Plus, extra protection baked in**. ECS will only drain connections from the previous Task Definition if the new Task Definition passes health checks.

Flexible scaling for performance

- **Scale a service up or down based on CloudWatch alarms.** Autoscaling is built into the Service during the registration process.
- Since Clusters are part of EC2 Autoscaling Groups, you can also **scale the Cluster itself based on resources**, like you would any other group.

A great disturbance in the force

- With the shift to microservices, comes a shift in thinking: more and more options are moving from just the server level to the containers themselves.
- Don’t just move a service over to containers and call it a day: decompose and rebuild.
- Security (IAM), scaling (Task-level autoscaling), traffic distribution (ALB and NLB), configuration, settings → all happening at the container/ service level now.

With more services comes more responsibility

- More moving pieces
- Safety and security first
- Choose the right option (tool, language, setting) that works for you.
- Use your resources! Document, alert, automate.

IAM Roles for Tasks

Task Definition: message:12

View detailed information for your task definition. To modify the task definition, you need to create a new revision and then make the required changes to the task definition

Create new revision | Actions

Builder | JSON

Task Definition Name message

Task Role message-service-role

Inline Policies

This view shows all inline policies that are embedded in this role.

Create Role Policy

Policy Name	Actions
message-dynamodb-table	Show Policy | Edit Policy | Remove Policy | Simulate Policy
message-queue	Show Policy | Edit Policy | Remove Policy | Simulate Policy

Amazon ECS Task Placement

Service : message

Update Delete

Details

Cluster	demo
Status	ACTIVE
Task Definition	message:12
Desired count	2
Pending count	0
Running count	2
Service Role	ecsServiceRole

Load Balancing

Target Group Name	Container Name	Container Port
message	message	3000

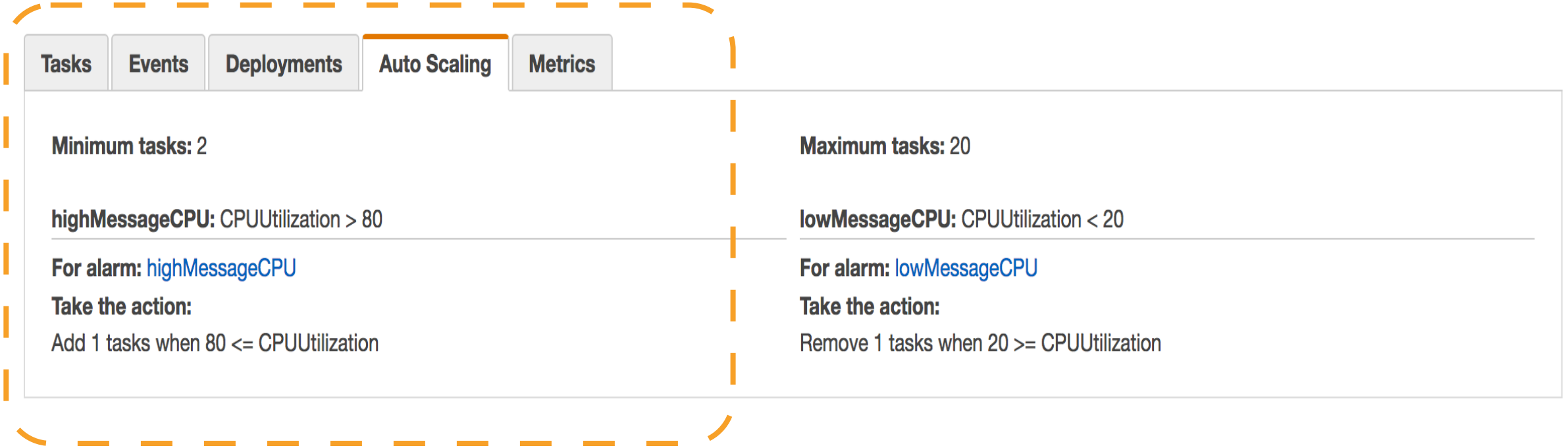
Deployment Options

Minimum healthy percent	50
Maximum percent	200

Task Placement

Strategy	spread(attribute:ecs.availability-zone), spread(instanceId)
Constraint	No constraints

Autoscaling



Deployments

Filter in this page

1-100

Event Id	Event Time	Message
33d4ee40-7aea-443c-a340-d34b218ce936	2017-02-21 13:44:04 -0500	service web has reached a steady state.
b836d703-bc01-4235-ba57-13c48e4a8b6c	2017-02-21 13:43:52 -0500	service web has stopped 2 running tasks: task fedd838d-36b4-4510-b3a1-a97a9fe1f427 task 3bb17653-d5a8-4ada-b575-21bc17006e8d.
c0882f96-24b3-4e65-9c8d-8fadfdf5400e	2017-02-21 13:38:47 -0500	service web has begun draining connections on 2 tasks.
19a38f62-ad84-4125-ad5a-ddd801a4b277	2017-02-21 13:38:47 -0500	service web deregistered 2 targets in target-group web
895b98ab-b21c-4334-a35b-de4bf0b3e74c	2017-02-21 13:38:35 -0500	service web registered 2 targets in target-group web
d385c9d6-a2ac-4810-ad9e-3d37bdcb4677	2017-02-21 13:38:21 -0500	service web has started 2 tasks: task 46b292ba-3c02-411c-a8ca-e4039d7885dc task e00a426d-4f4f-476d-934d-46ed525da640.

Amazon ECS Event Stream for CloudWatch

Event Pattern Schedule

Build event pattern to match events by service

Service Name: EC2 Container Service (ECS)

Event Type: State Change

Any detail type Specific detail type(s)

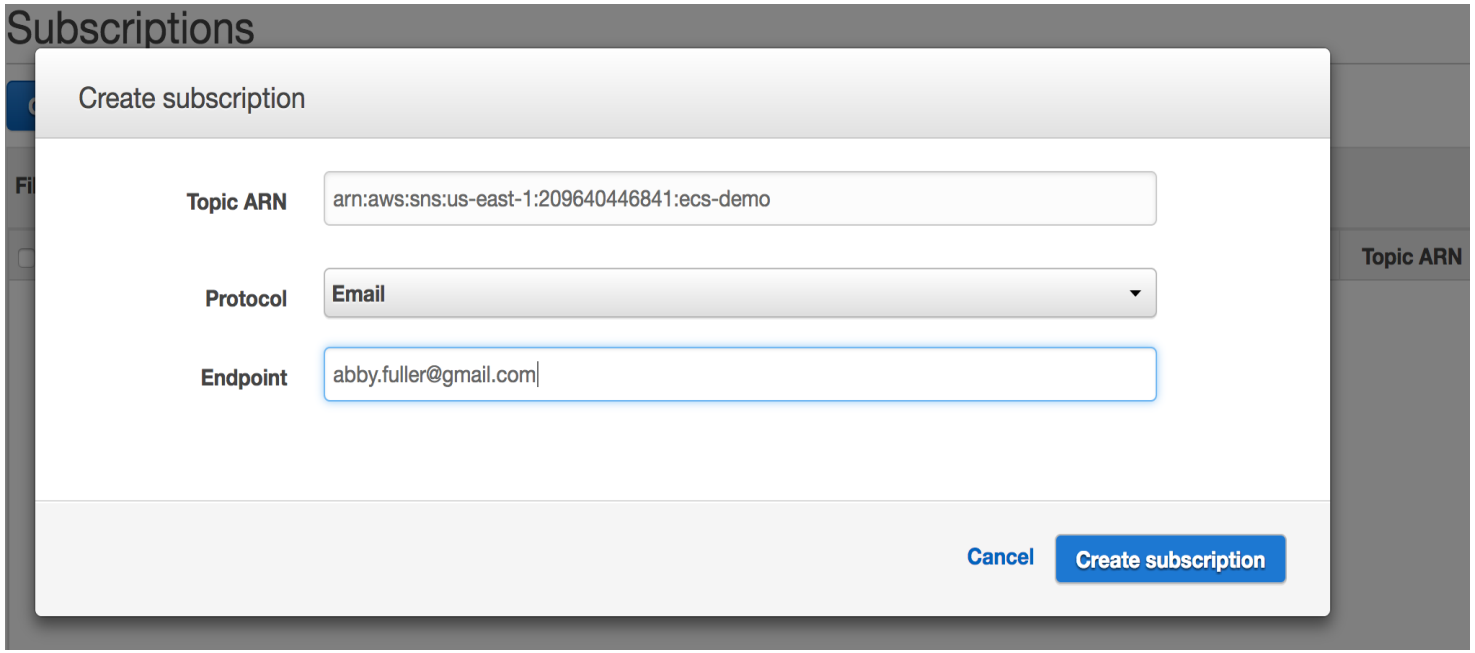
× ECS Task State Change

Any cluster Specific cluster(s)

× arn:aws:ecs:us-east-1:209640446841:cluster/demo

Event Pattern Preview Copy to clipboard Edit

```
{
  "source": [
    "aws.ecs"
  ],
  "detail-type": [
    "ECS Task State Change"
  ],
  "detail": {
    "clusterArn": [
      "arn:aws:ecs:us-east-1:209640446841:cluster/demo"
    ]
  }
}
```



Some ECS resources

- AWS docs: https://aws.amazon.com/ecs/
- ECS first run wizard: https://console.aws.amazon.com/ecs/home?region=us-east-1
- Nathan Peck's ECS repo: https://github.com/nathanpeck/awesome-ecs
- More talks of mine: https://aws.amazon.com/evangelists/abby-fuller/
- ECS "Getting Started" workshop: https://www.github.com/abby-fuller/ecs-demo

Questions?