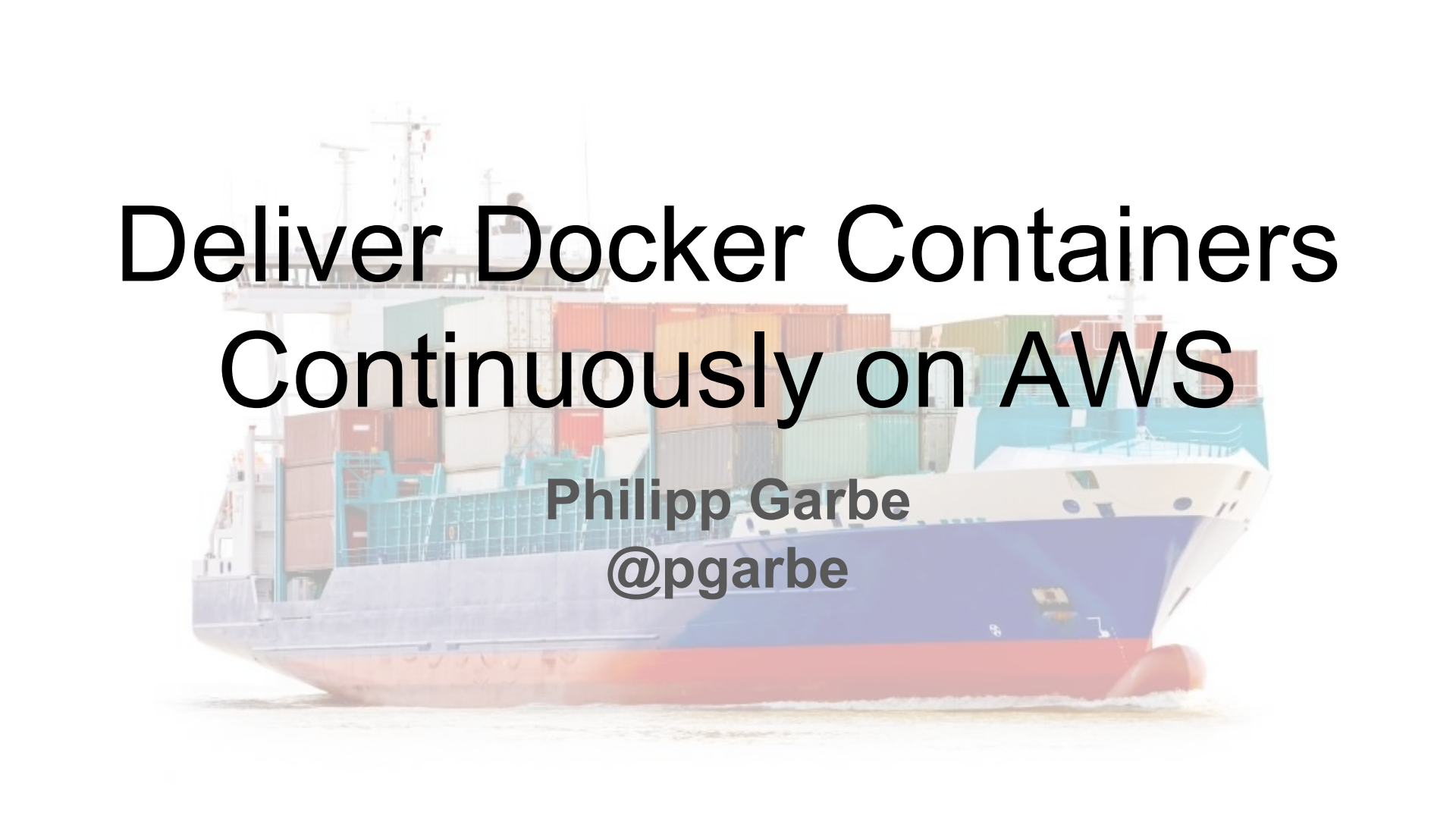


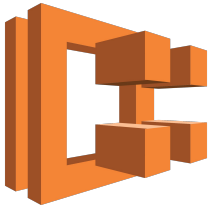
Deliver Docker Containers Continuously on AWS

Philipp Garbe
@pgarbe

So many choices...

Azure Container Services

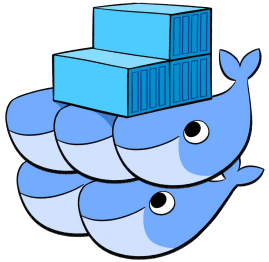
Google Container Engine



Amazon ECS

Cloud Foundry's Diego

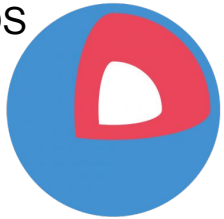
Kubernetes



Docker Swarm

Mesosphere Marathon

CoreOS Fleet



https://www.linux.com/news/8-open-source-CONTAINER-ORCHESTRATION-TOOLS-KNOW

About Me

- Philipp Garbe
- Lead Developer @Scout24
- Docker Captain
- Living in Bavaria
- Working in the Cloud

SCOUT 24


“Hello ECS”

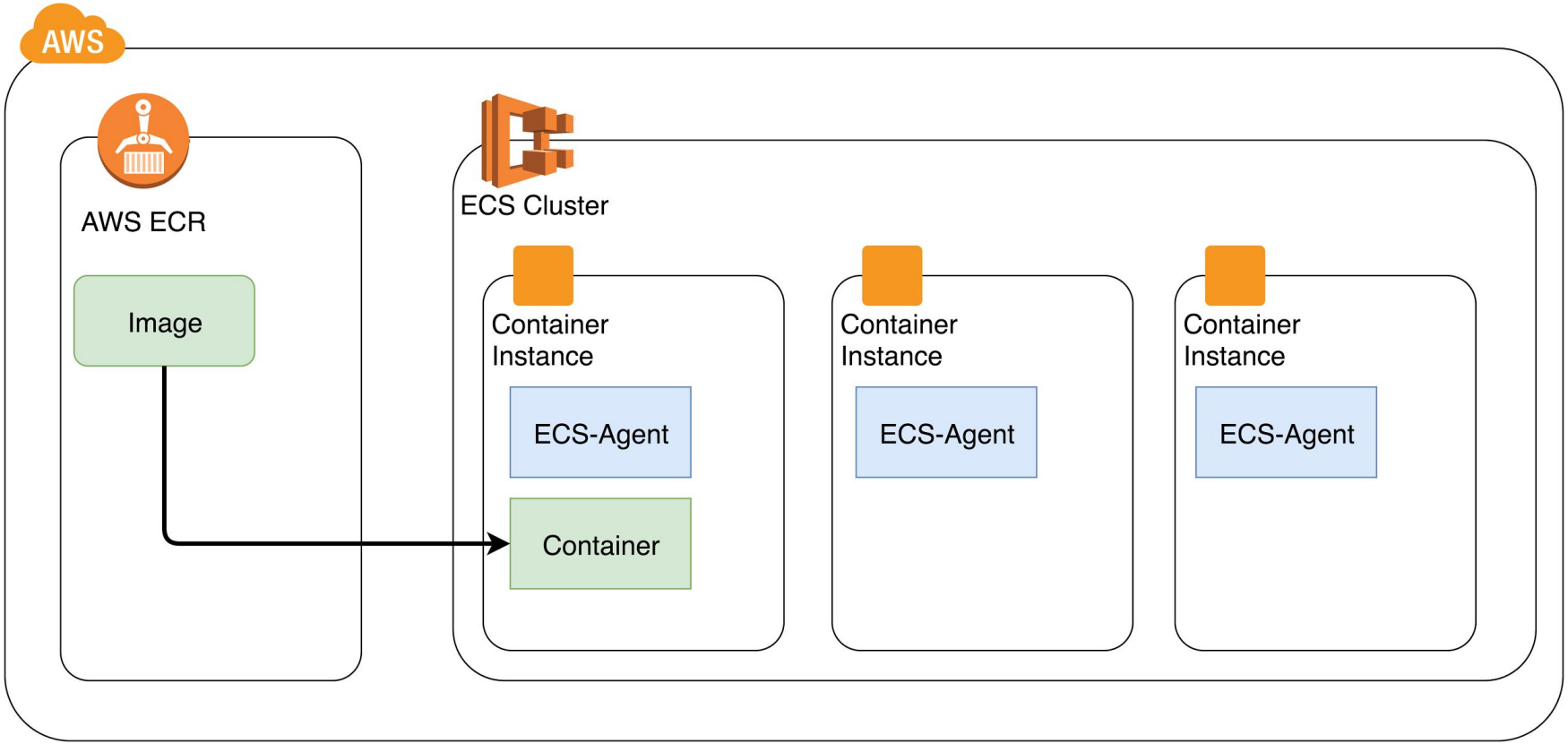
Our first ECS cluster

ECS Cluster: Deployment Options

	AWS Console	AWS CLI	ECS CLI	CloudFormation
Easy to start	Yes	No	Yes	No
Automation	No	Yes	Yes	Yes
Infrastructure as Code	No	No	No	Yes
Auto Scaling	Yes	Yes	No	Yes

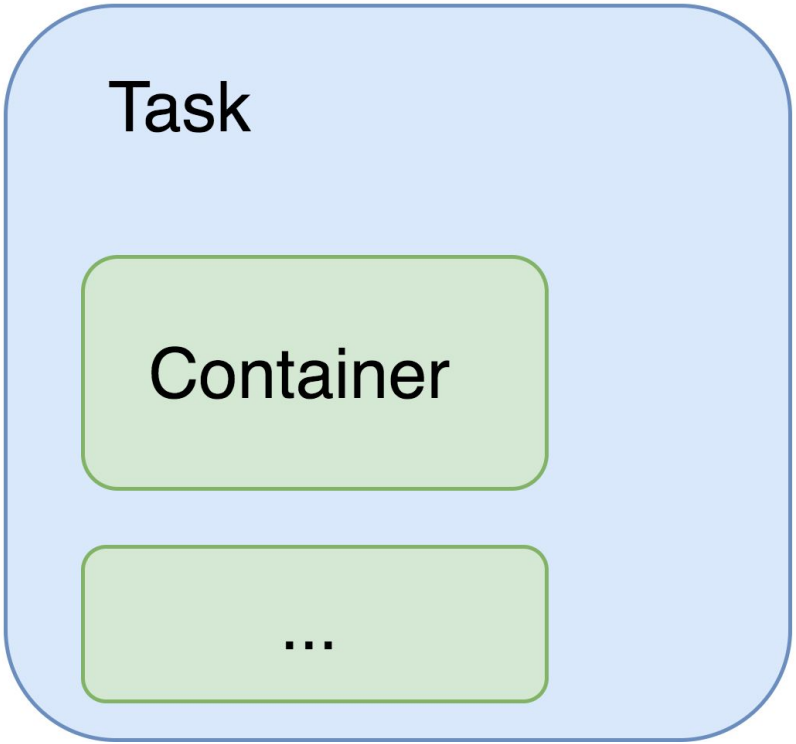
```yaml
AWSTemplateFormatVersion: '2010-09-09'

Parameters:
 KeyName:
   Type: AWS::EC2::KeyPair::KeyName
   Description: EC2 KeyPair to enable SSH access.
   ...

Resources:
 ECSCluster:
   Type: AWS::ECS::Cluster

 ECSAutoScalingGroup:
   Type: AWS::AutoScaling::AutoScalingGroup
   Properties:
     VPCZoneIdentifier: !Ref ServiceSubnets
     LaunchConfigurationName: !Ref LaunchConfig
     MinSize: !Ref ClusterMinSize
     MaxSize: !Ref ClusterMaxSize

 LaunchConfig:
   Type: AWS::AutoScaling::LaunchConfiguration
   Metadata:
     AWS::CloudFormation::Init:
       config:
         commands:
           01_add_instance_to_cluster:
             command: !Sub |
               #!/bin/bash
               echo ECS_CLUSTER=${ECSCluster} >> /etc/ecs/ecs.config
   Properties:
     ImageId: !FindInMap: [AWSRegionToAMI, Ref: AWS::Region, AMIID]
     InstanceType: !Ref InstanceType
     IamInstanceProfile: !Ref EC2InstanceProfile
     KeyName: !Ref KeyName
     ...

Outputs:
 ClusterName:
   Value: !Ref ECSCluster
   Export:
     Name: !Sub "${AWS::StackName}-ClusterName"
```

The first deployment



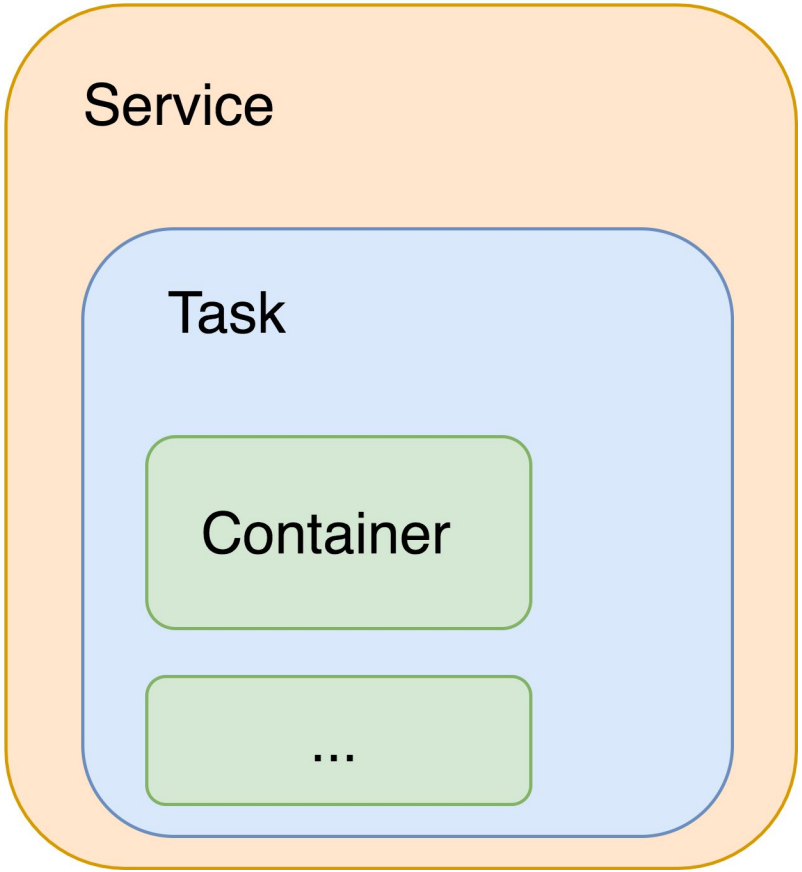
Container Definition

- Image
- Port mapping
- Mount points
- Network options
- Docker options



Task Definition

- IAM Task Role
- Volumes
- Network Mode
- Task Placement Constraints



Service Description

- Loadbalancer
- AutoScaling
- Deployment Configuration
- Task Placement Strategy

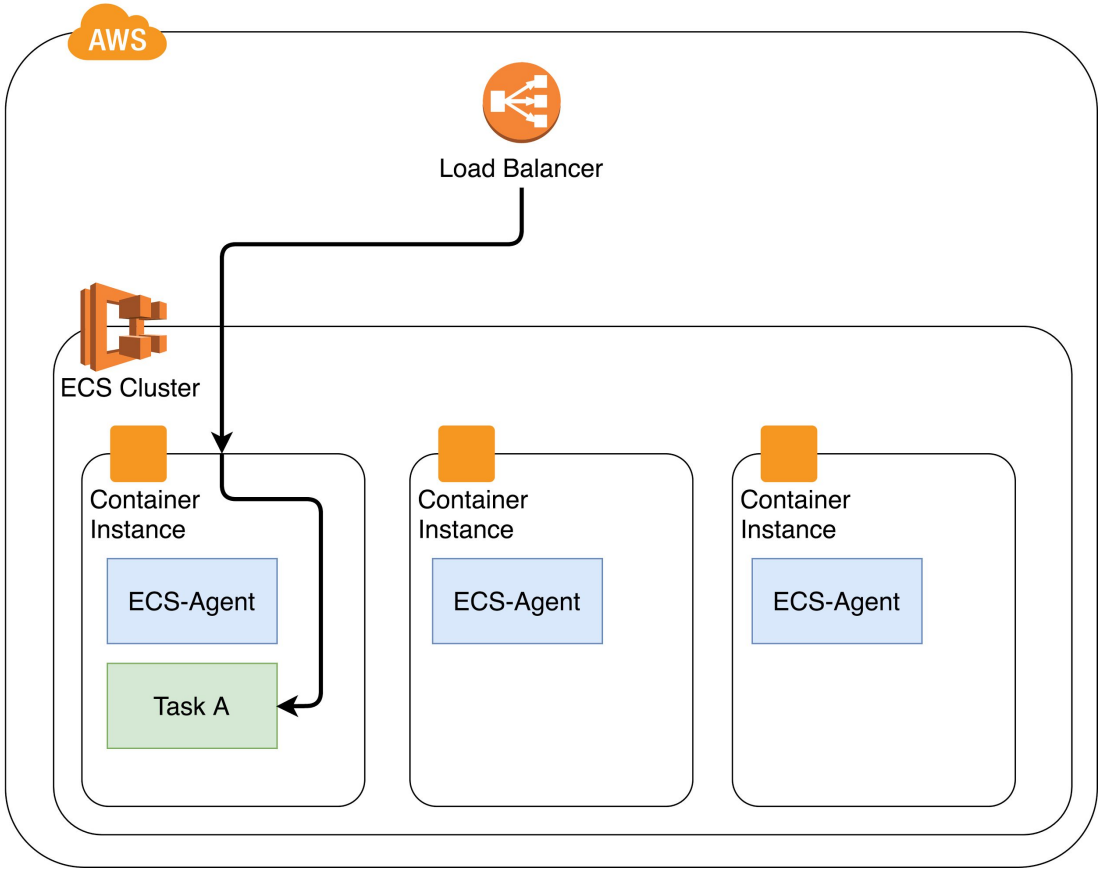
ECS Service: Deployment Options

	AWS Console	AWS CLI	ECS CLI	CloudFormation
Easy to start	Yes	No	Yes	No
Automation	No	Yes	Yes	Yes
Configuration as Code	No	No	Partially	Yes
Auto Scaling	Yes	Yes	No	Yes
Load Balancer	Yes	Yes	No	Yes
Task Placement	Yes	Yes	No	No *

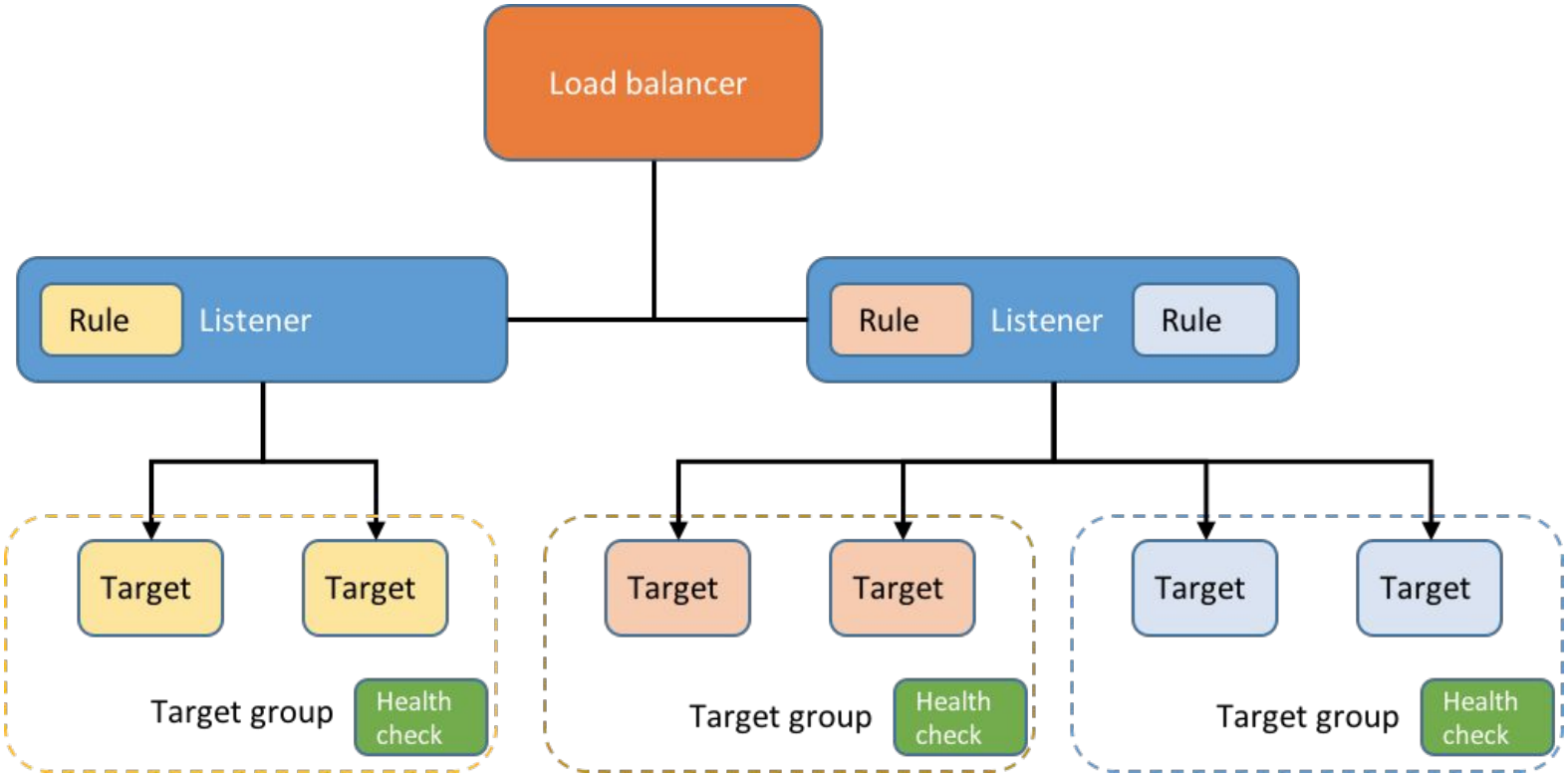
```yaml
AWSTemplateFormatVersion: '2010-09-09'

Parameters:
 DesiredCount:
   Type: Number
 ClusterStack:
   Type: String
   Description: Name of the cluster stack
 ...

Resources:
 TaskDefinition:
   Type: AWS::ECS::TaskDefinition
   Properties:
     TaskRoleArn: !Ref TaskAuthRole
     ContainerDefinitions:
     - Name: nginx
       Image: !Sub nginx:${Version}
       Cpu: '2048'
       PortMappings:
       - ContainerPort: 80
       Memory: '1024'
       Essential: 'true'
```

```yaml
WebApp:
  Type: AWS::ECS::Service
  Properties:
    Cluster:
      "Fn::ImportValue": !Sub "${ClusterStack}-ClusterName"
    TaskDefinition: !Ref TaskDefinition
    DesiredCount: !Ref DesiredCount
    DeploymentConfiguration:
      MaximumPercent: 200
      MinimumHealthyPercent: 100
    Role: !Ref ServiceAuthRole
    LoadBalancers:
    - ContainerName: nginx
      ContainerPort: 80
      TargetGroupArn: !Ref TargetGroup
```

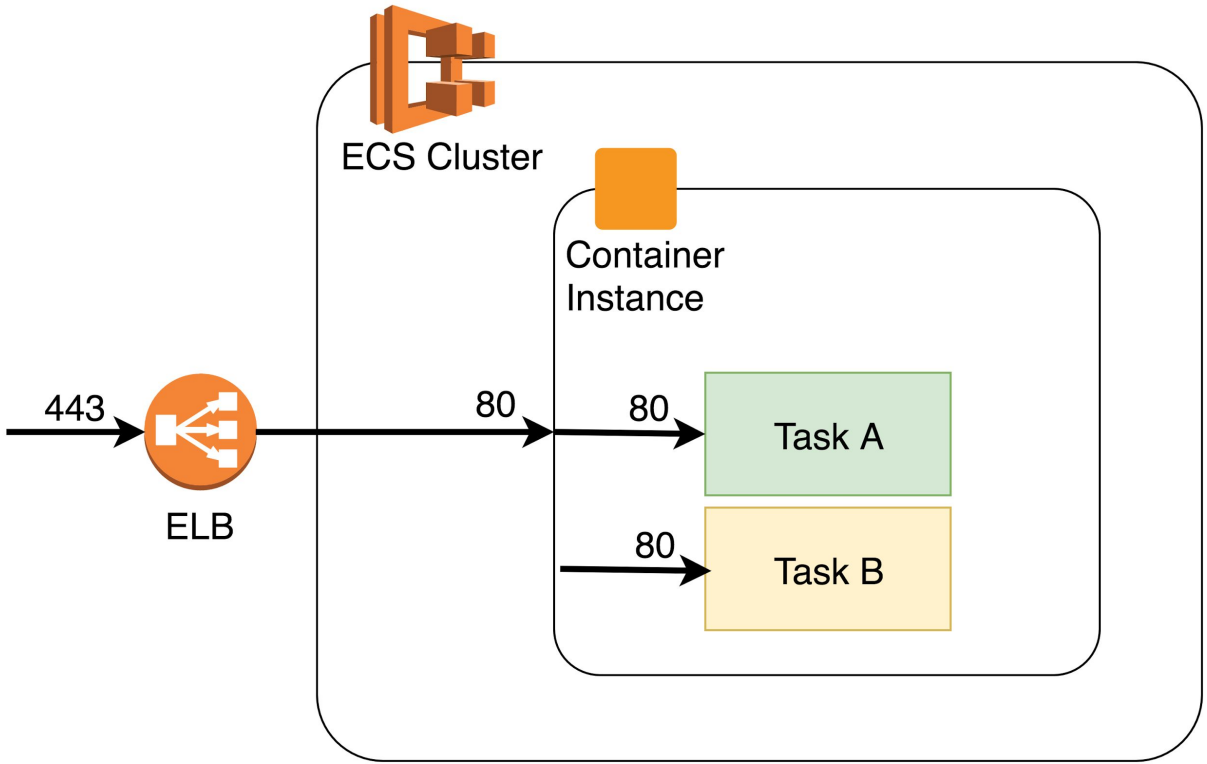
Load Balancing



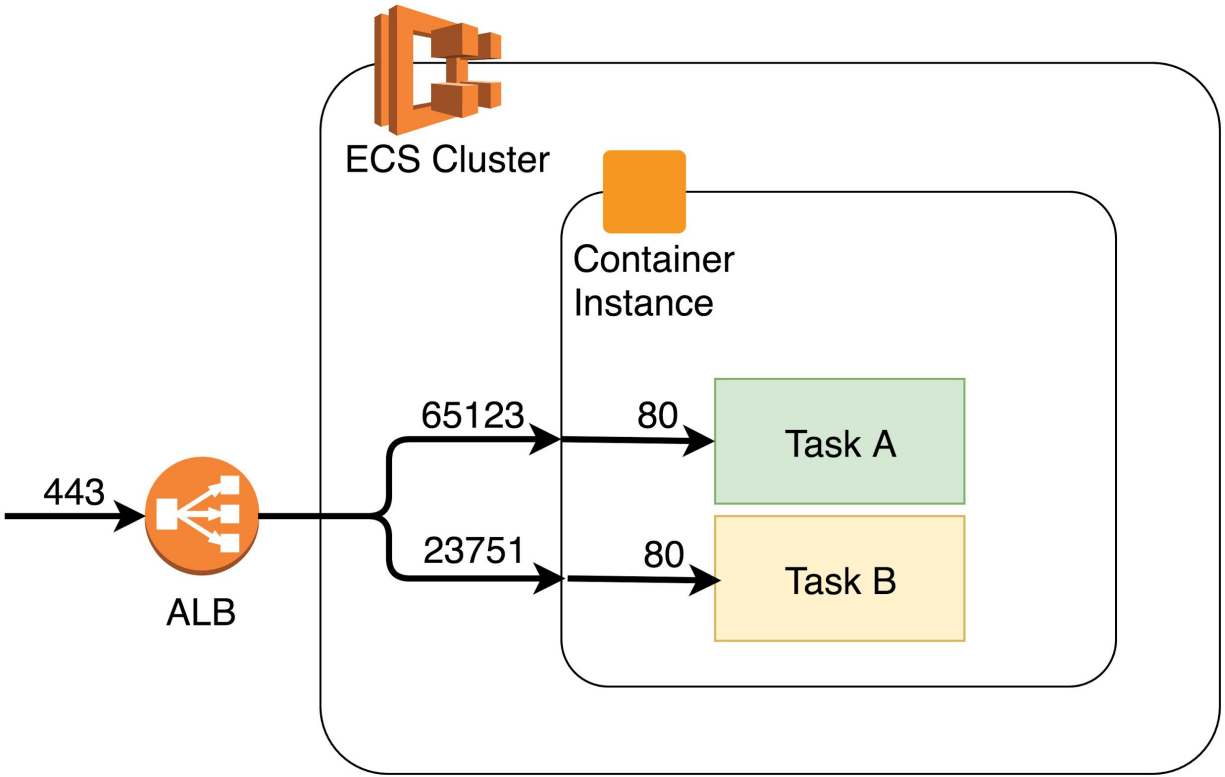
Application Load Balancer (ALB)



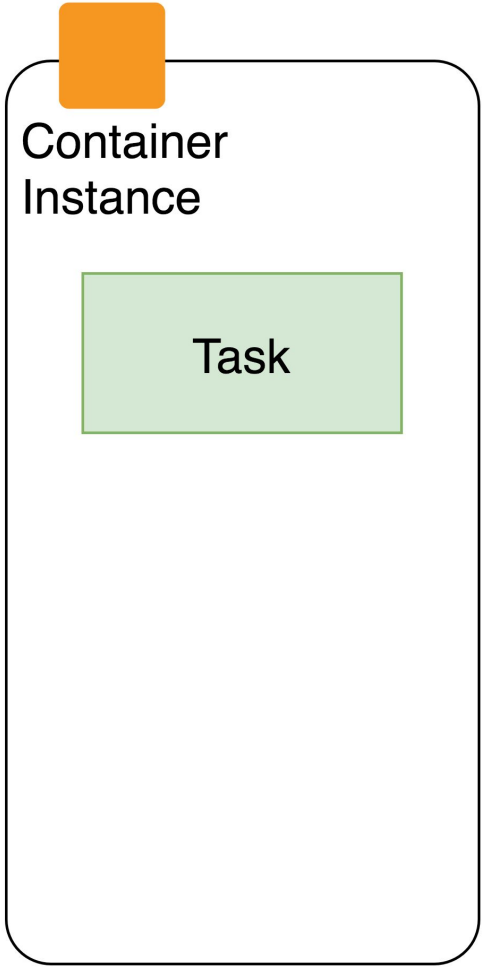
Static Port Mapping (ELB)

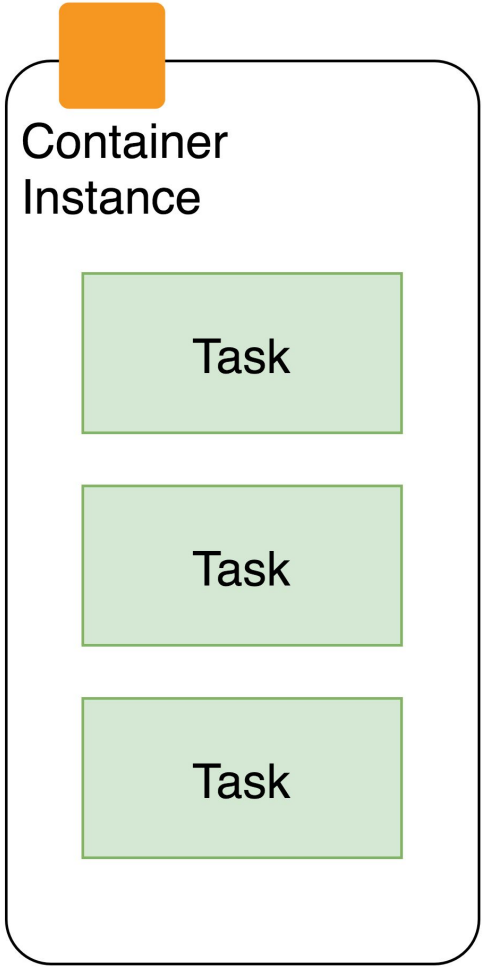


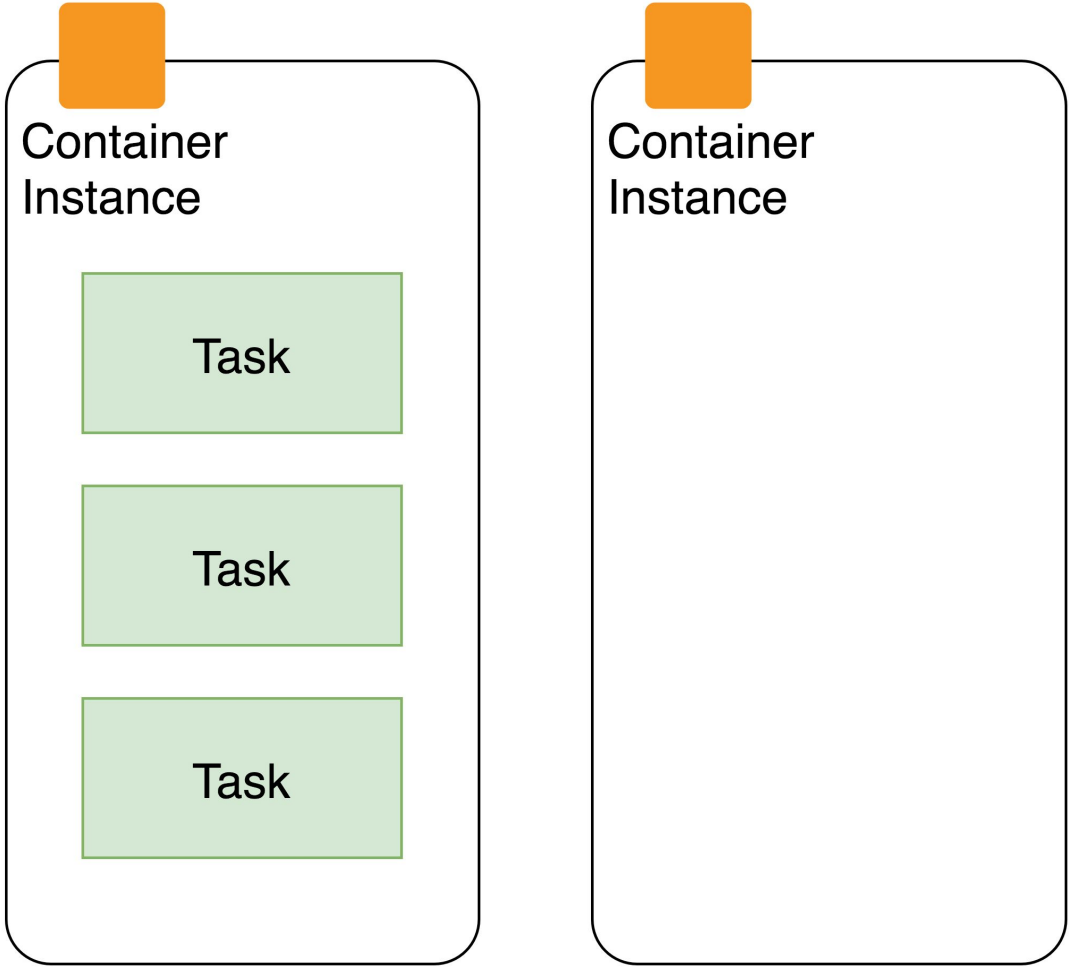
Dynamic Port Mapping (ALB)

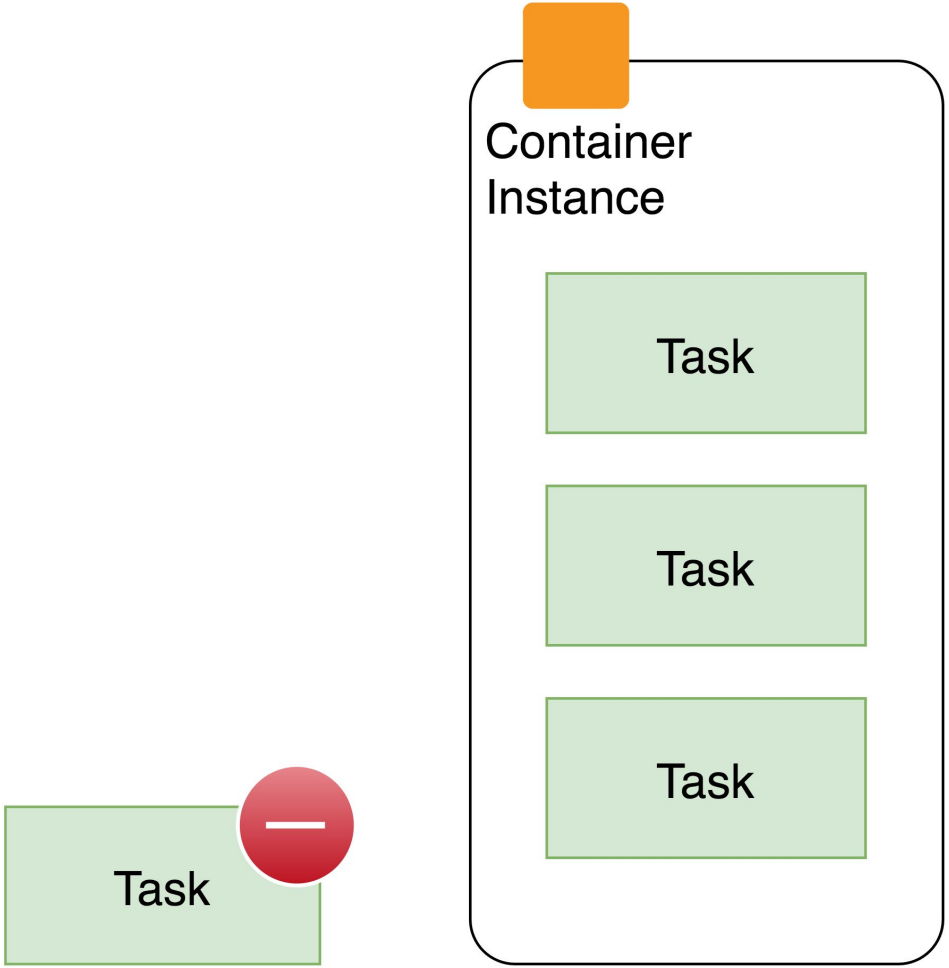


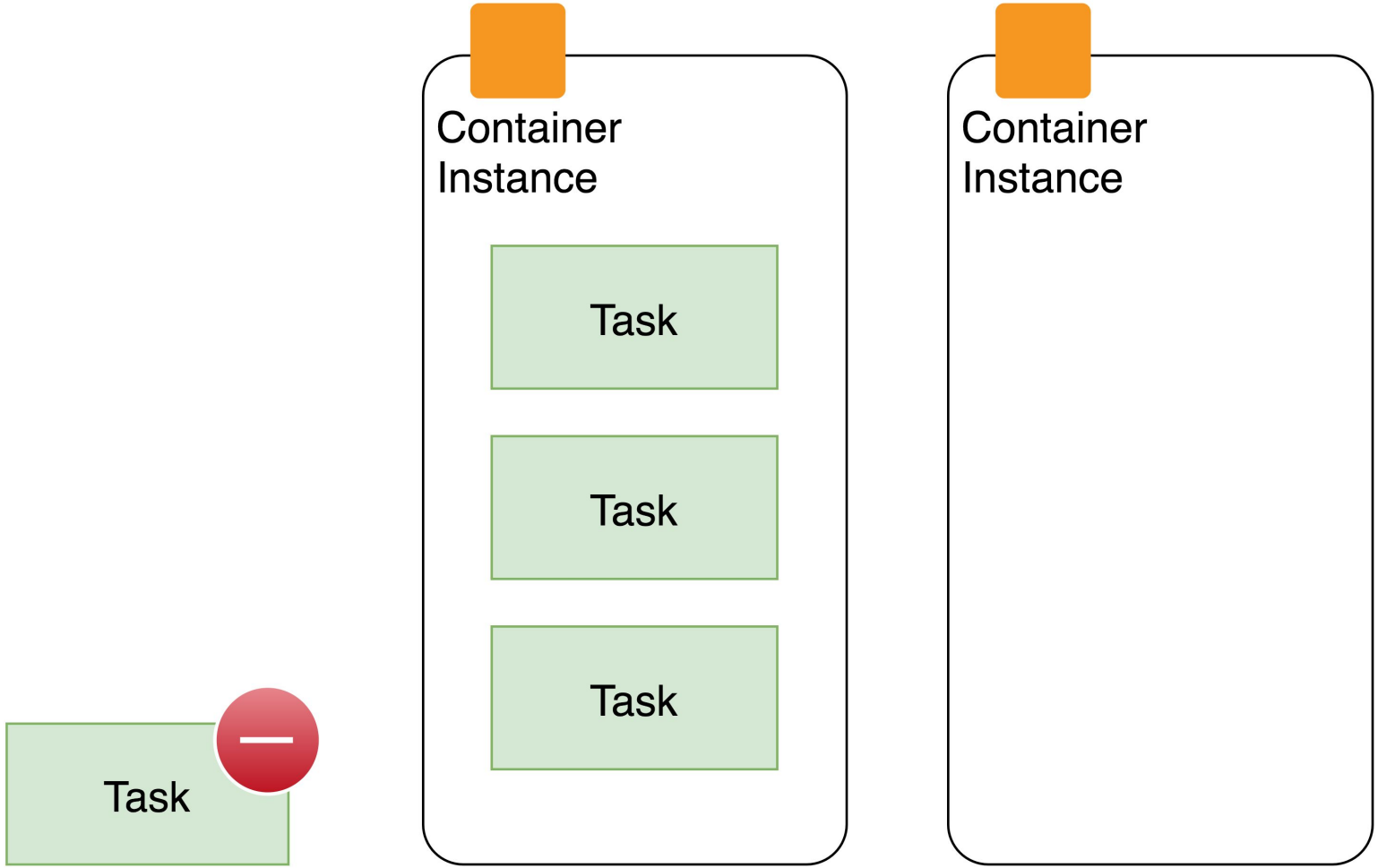

Up & Down
/29
:00
10/30
00:00
10/31
00:00
11/1
00:00
11/2
00:00
11/3
00:00

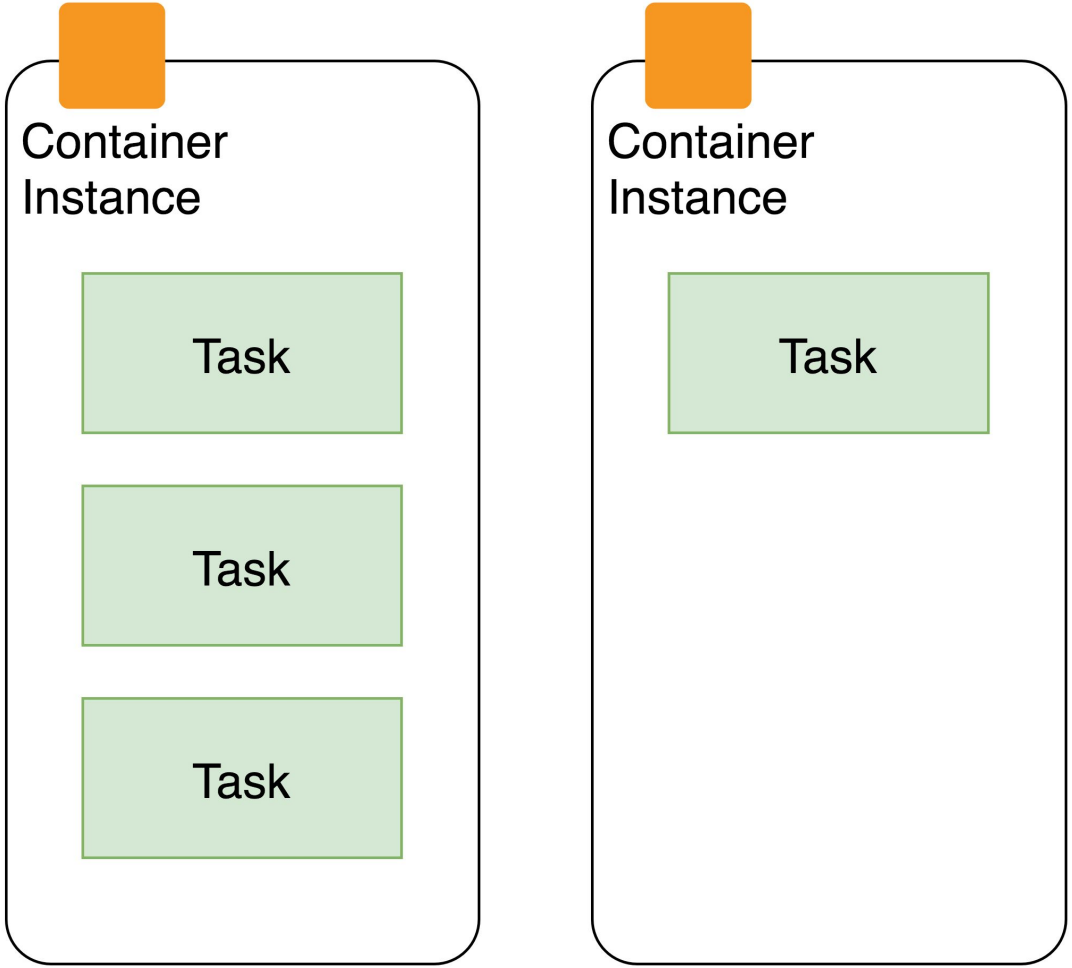

Container
Instance
Task

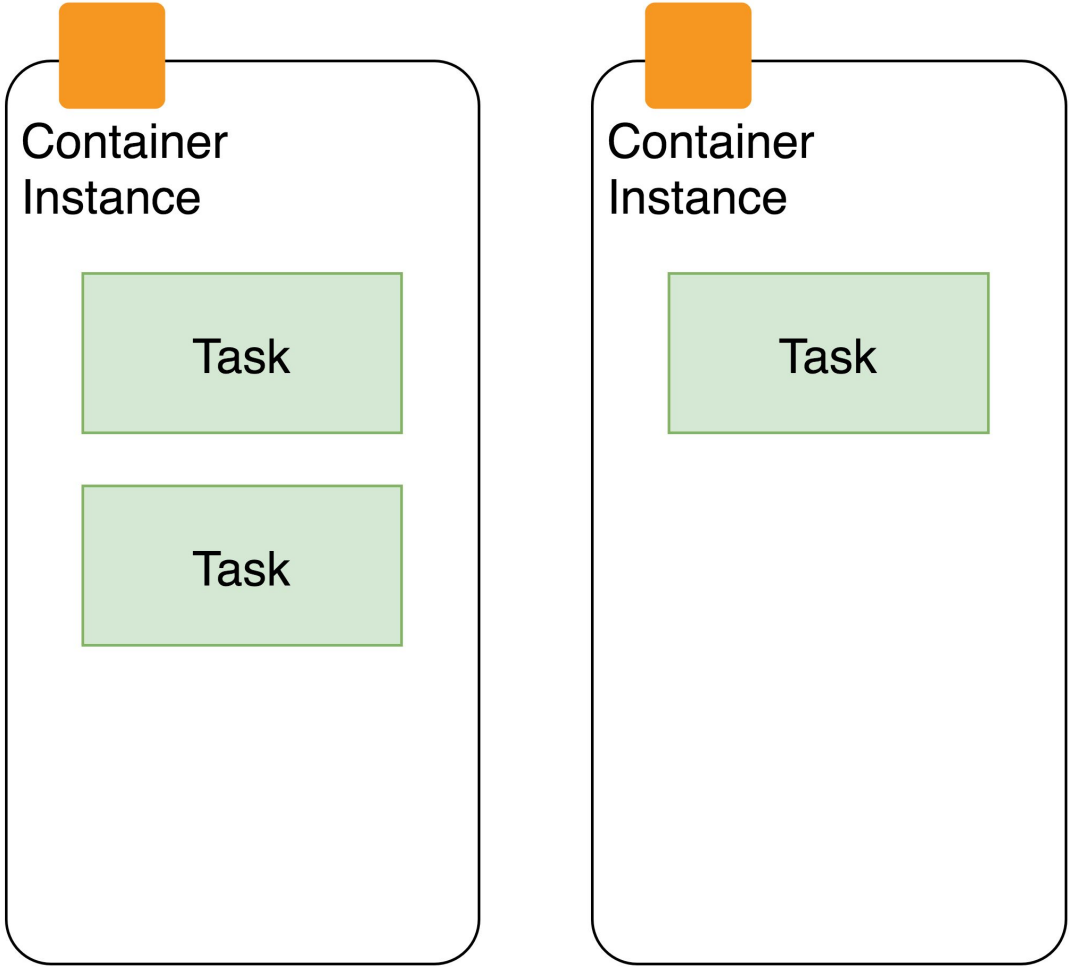

Container
Instance
Task
Task
Task

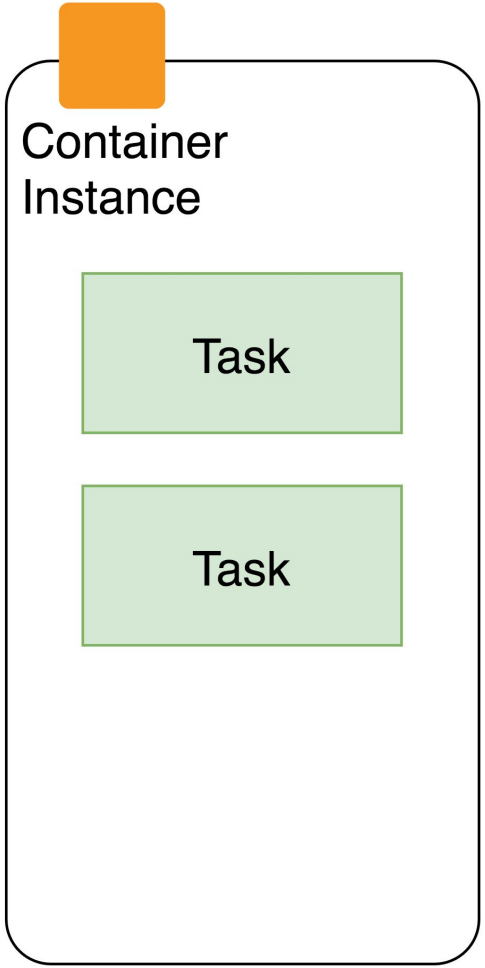

Container
Instance
Task
Task
Task
Container
Instance

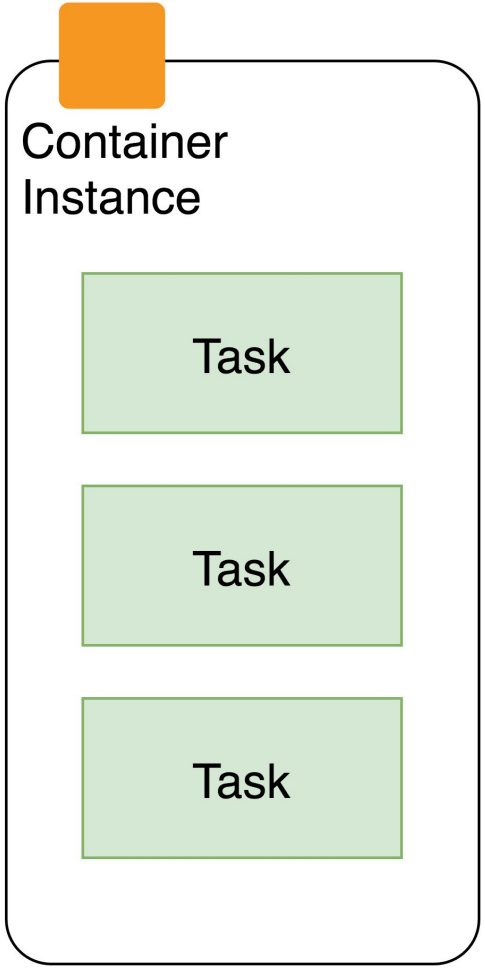

Container
Instance
Task
Task
Task
Task


Container
Instance
Task
Task
Task
Container
Instance
Task


Container
Instance
Task
Task
Task
Container
Instance
Task


Container
Instance
Task
Task
Container
Instance
Task


Container
Instance
Task
Task


Container
Instance
Task
Task
Task

AutoScaling: Conclusion

- Two different kinds of scaling (cluster and service)
  - Cluster: Use cpu / memory reservation metrics
  - Service: Use cpu / memory utilization metrics
- Scale down to save money, but avoid endless-loop
- Scaling takes awhile to take effect
- ASG is not aware of ECS

AutoScaling: Rule of Thumb

Threshold = (1 - max(Container Reservation) / Total Capacity of a single Container Instance) * 100

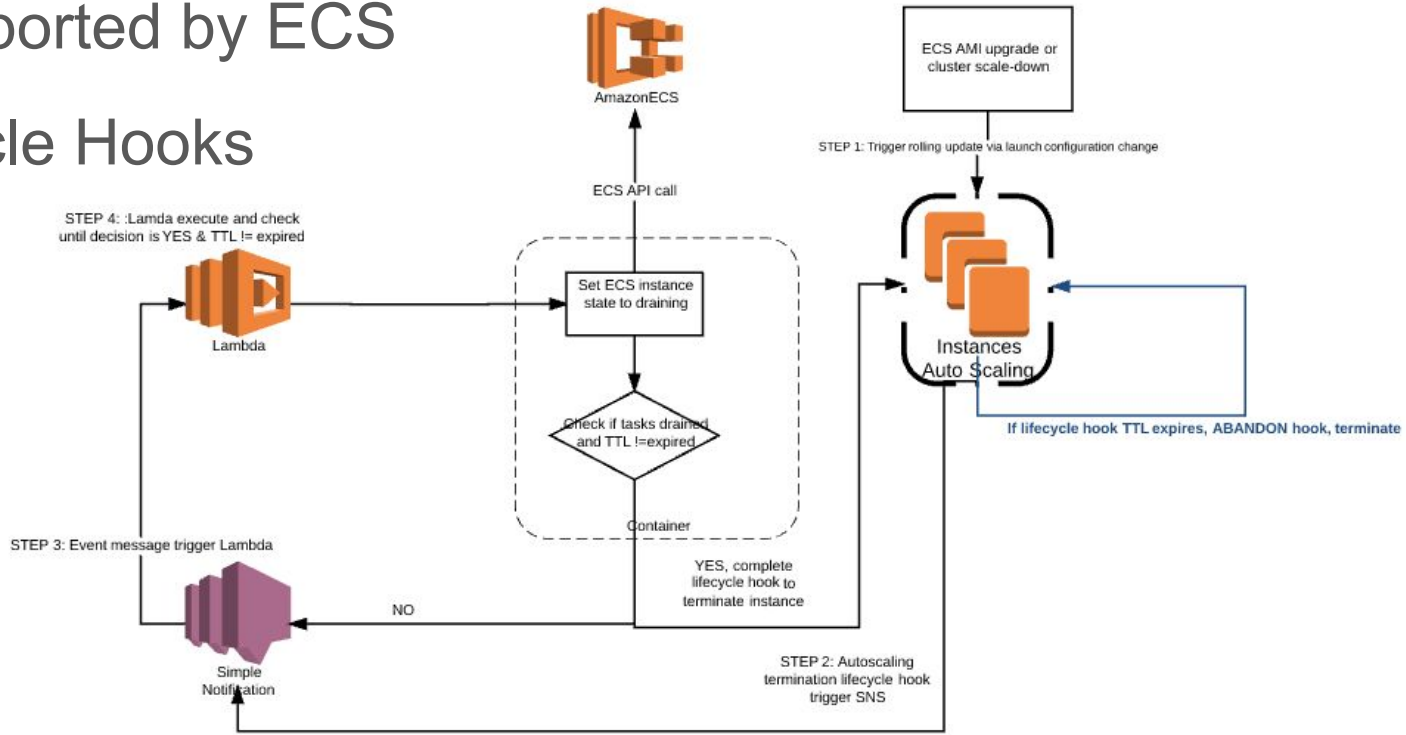
Example:

Container instance capacity: 2048 MB
Container reservation: 512 MB

Threshold = (1 - 512 / 2048) * 100
Threshold = 75%

Node draining

- Finally supported by ECS
- Use Lifecycle Hooks



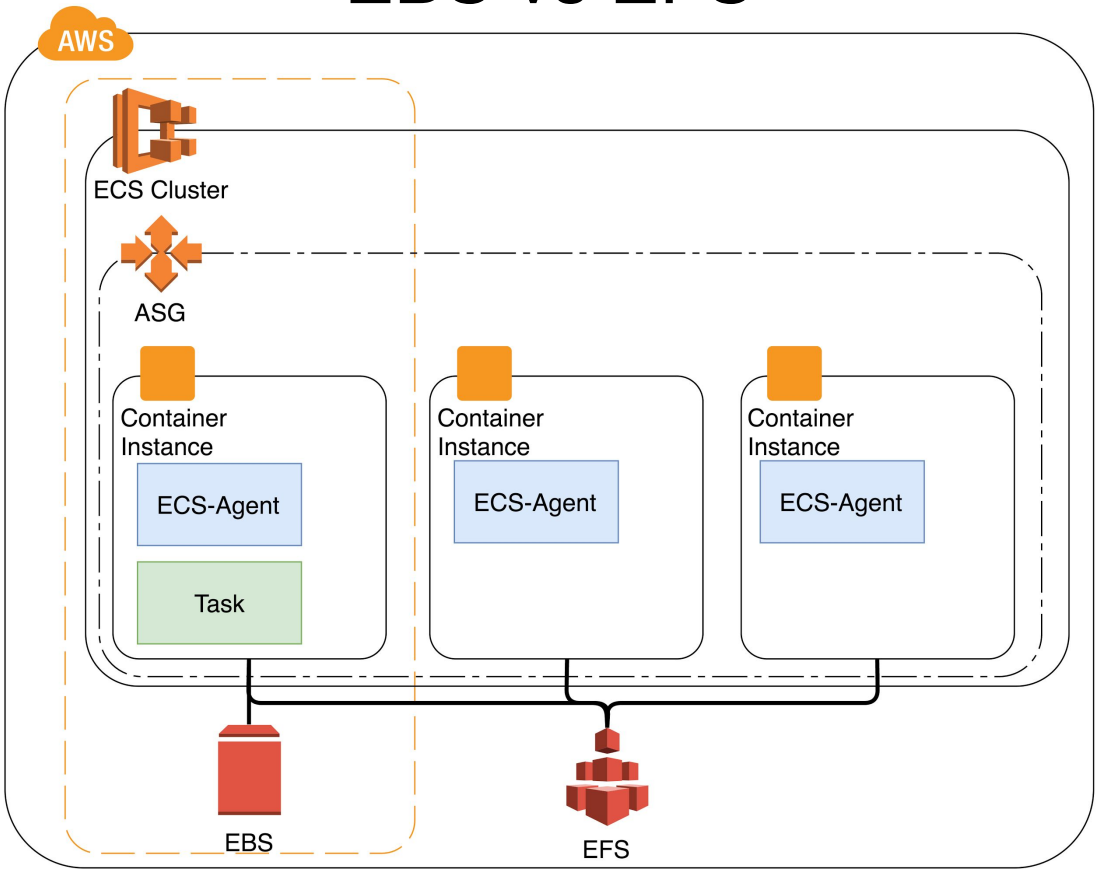
https://aws.amazon.com/blogs/compute/how-to-automate-container-instance-draining-in-amazon-ecs/

Best practices for ECS Cluster

- ASG UpdatePolicy defines deployment strategy
- cfn-init: Ensure Docker and ECS-Agent is running
- Put build no in UserData to enforce new EC2 instances

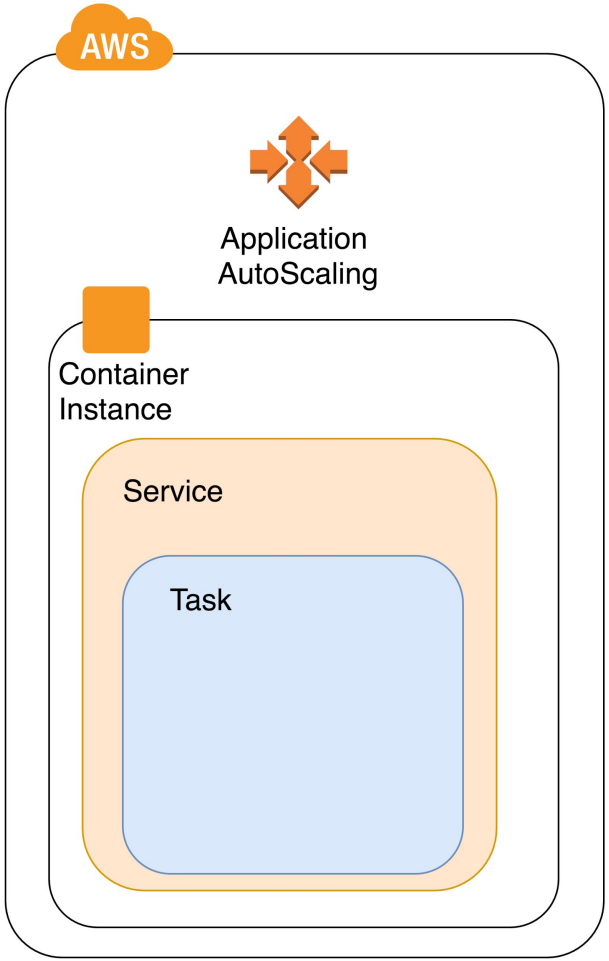
Volumes

EBS vs EFS


AWS
ECS Cluster
ASG
Container Instance
ECS-Agent
Task
Container Instance
ECS-Agent
Container Instance
ECS-Agent
EBS
EFS


Security

https://iam.cloudonaut.io



IAM Security Roles

ecsAutoScalingRole

- Read CloudWatch Metrics
- Modify App AutoScaling

ecsContainerInstanceRole

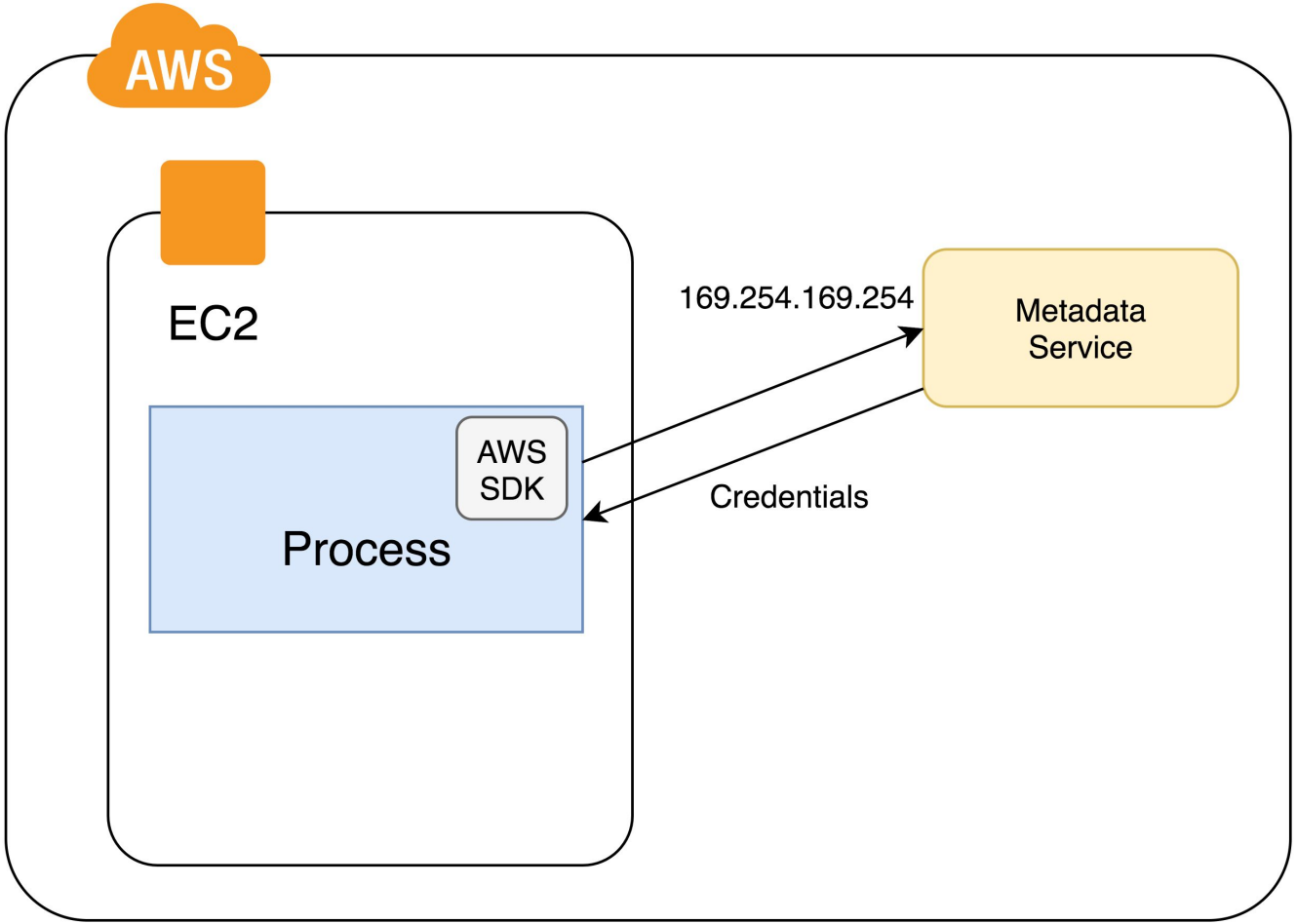
- ECR: Get Images
- ECS: De/Register Container Instances

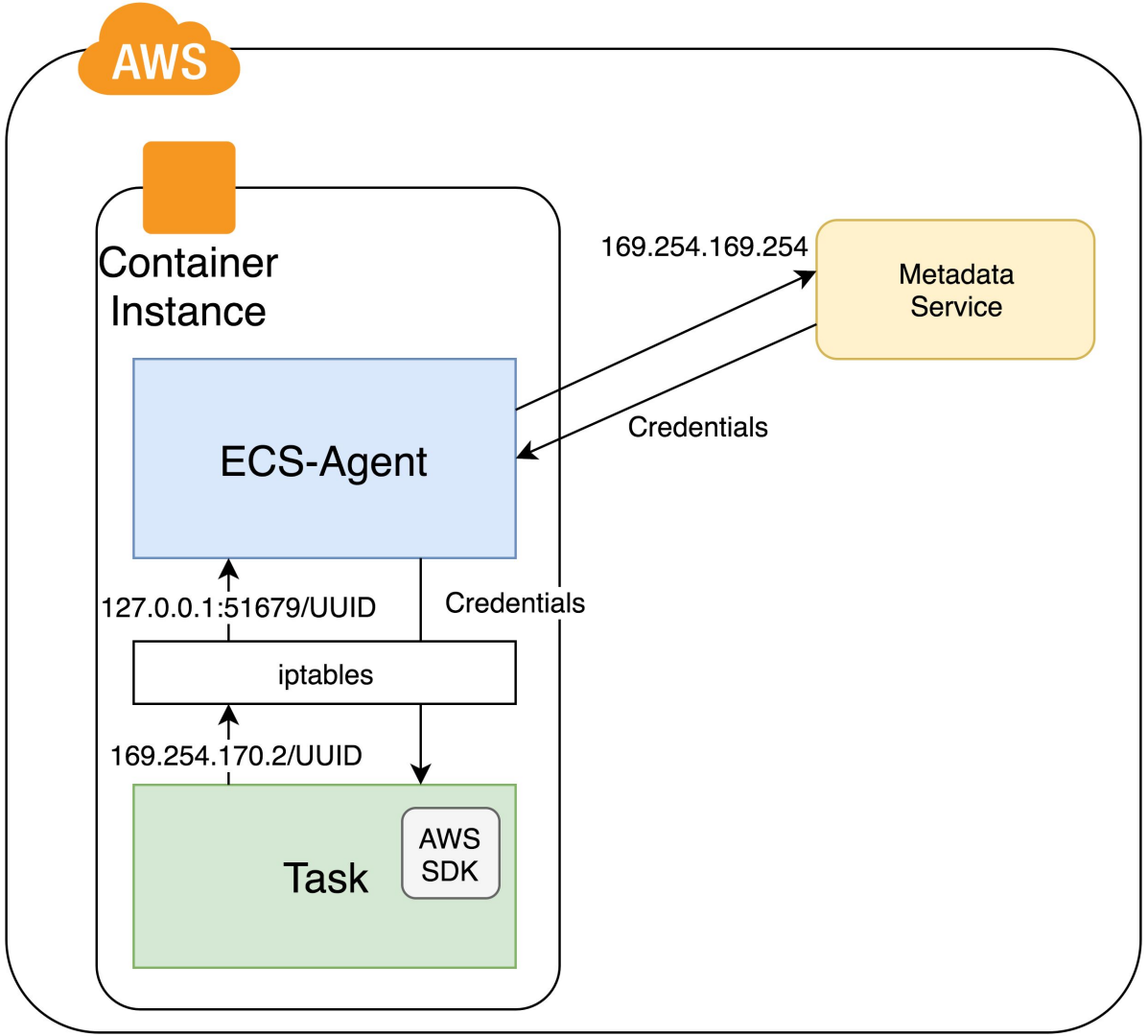
ecsServiceRole

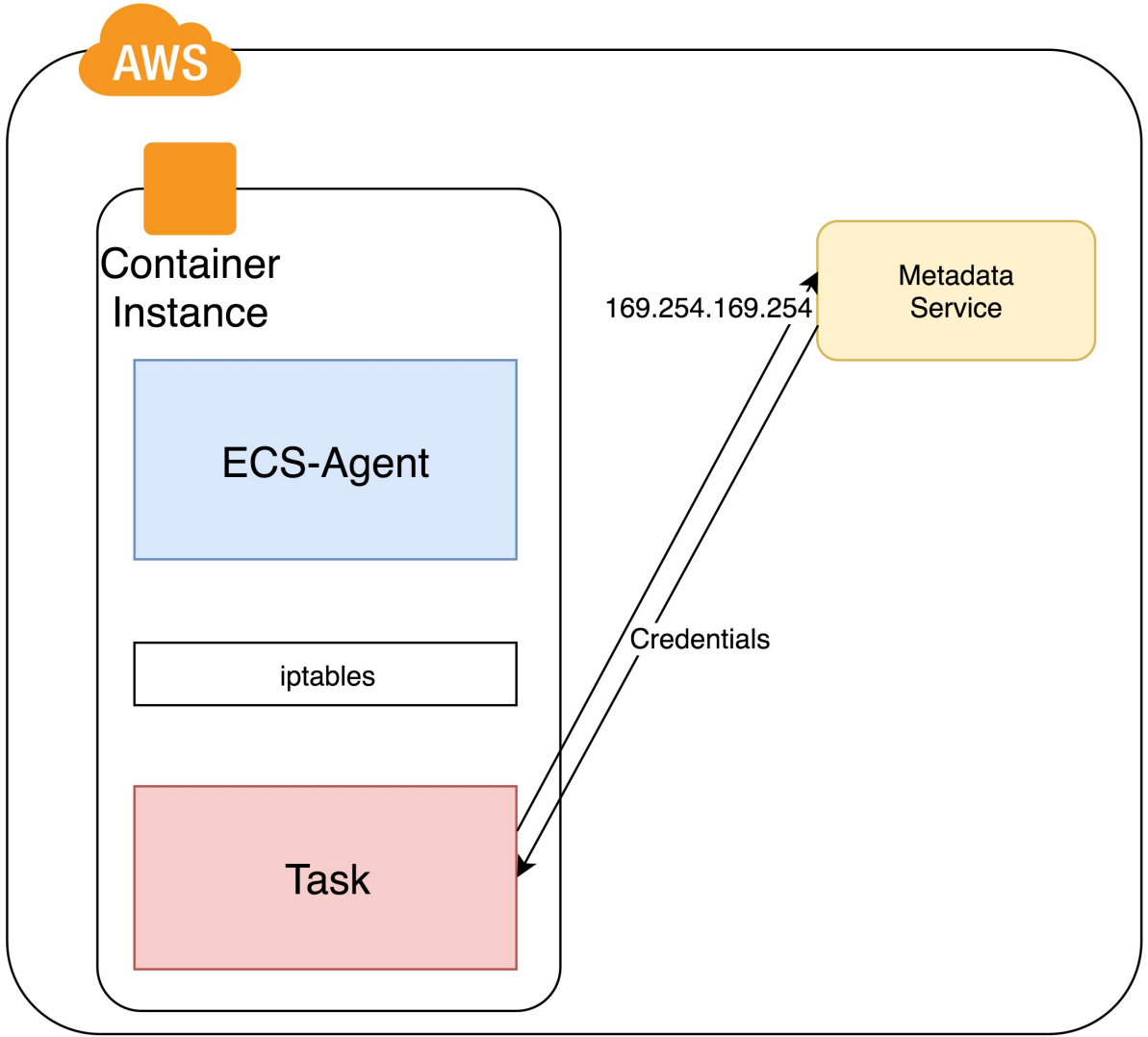
- De/Register Instances with Load Balancer

ecsTaskRole

- Everything your task needs to do


AWS
EC2
AWS
SDK
Process
169.254.169.254
Metadata
Service
Credentials


AWS
Container
Instance
169.254.169.254
Metadata
Service
Credentials
ECS-Agent
127.0.0.1:51679/UUID
Credentials
iptables
169.254.170.2/UUID
Task
AWS
SDK


AWS
Container
Instance
ECS-Agent
iptables
Task
169.254.169.254
Metadata
Service
Credentials

How to protect yourself

EC2

- Disallow access to metadata service from tasks (containers)

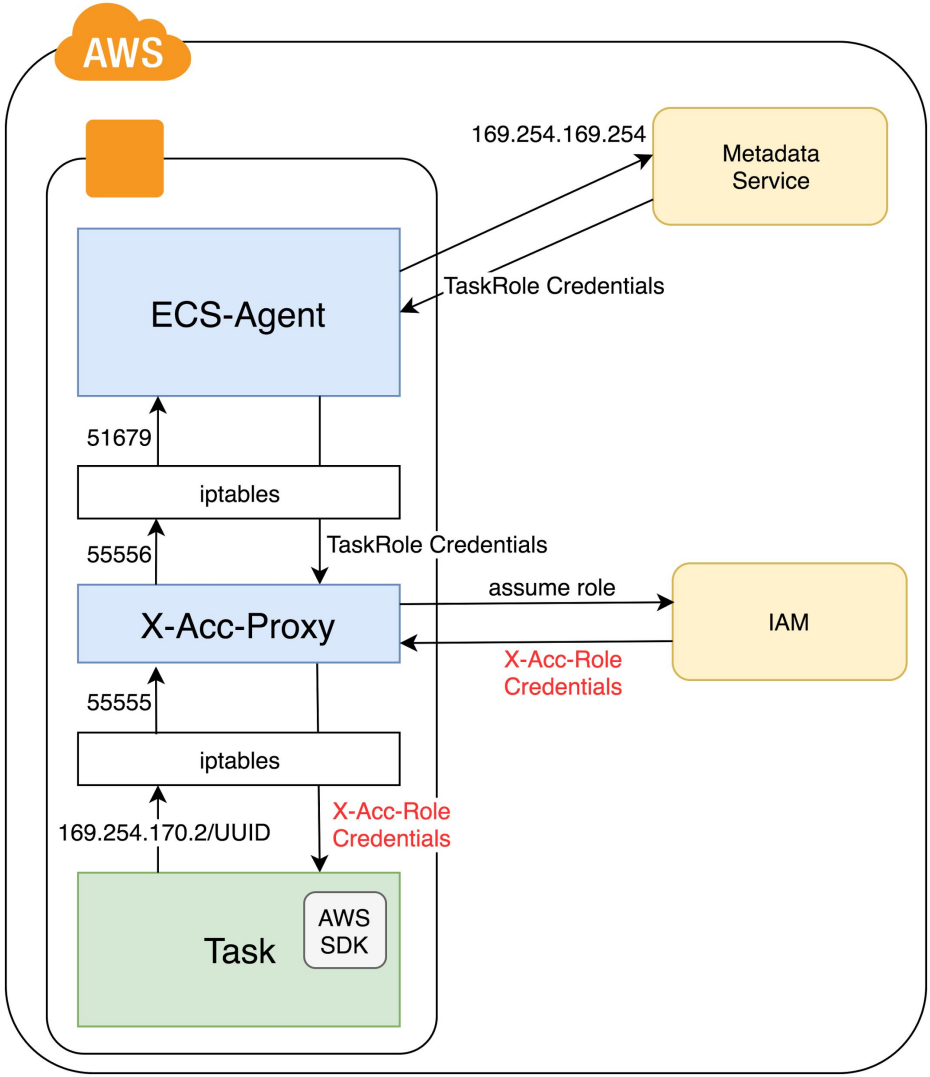
```
iptables --insert FORWARD 1 --in-interface docker+ --destination 169.254.169.254/32
--jump DROP
```

IAM

- Give the instance role only the credentials it needs (according to aws docs)

Cross Account Proxy

- Re-route call to ECS-Agent
- ECS-Agent gets credentials based on configured TaskRole
- TaskRole needs only one permission: AssumeRole
- X-Acc-Proxy assumes role (Role ARN comes from Docker Label)
- X-Acc-Proxy returns credentials from assumed role



Summary

What did we miss?

- Networking
- Logging
- Monitoring
- CloudWatch Events
- EC2 System Manager parameter store

Where ECS shines…

- Stable Environment
- Catched up with task placement engine
- Native support of IAM
- AutoScaling for hosts and services
- CloudFormation all the way

And where not...

- Does not support all the Docker features (e.g. HEALTHCHECK)
- Disconnect between Docker Compose and Task Definition
- Network philosophy is different (Still no SecurityGroups for Containers)
- Volumes still not natively supported (3rd party tools needed)
- It’s not a managed container service

Philipp Garbe


http://garbe.io


@pgarbe


https://github.com/pgarbe

SCOUT 24

IMMOBILIEN SCOUT 24

AUTO SCOUT 24

https://boards.greenhouse.io/scout24