Auto Scaling ECS Applications

Justin Menga
FULL STACK TECHNOLOGIST
@jmenga pseudo.co.de

Introduction

Auto Scaling ECS Applications

- ECS Auto Scaling
  - Scales your ECS services
- EC2 Auto Scaling
  - Scales your ECS clusters
- Auto Scaling Solution
  - CloudWatch Events
  - ECS Capacity Manager Lambda Function
  - CloudWatch Metrics and Alarms
  - Auto Scaling Policies
  - CloudFormation support

ECS Auto Scaling Challenges

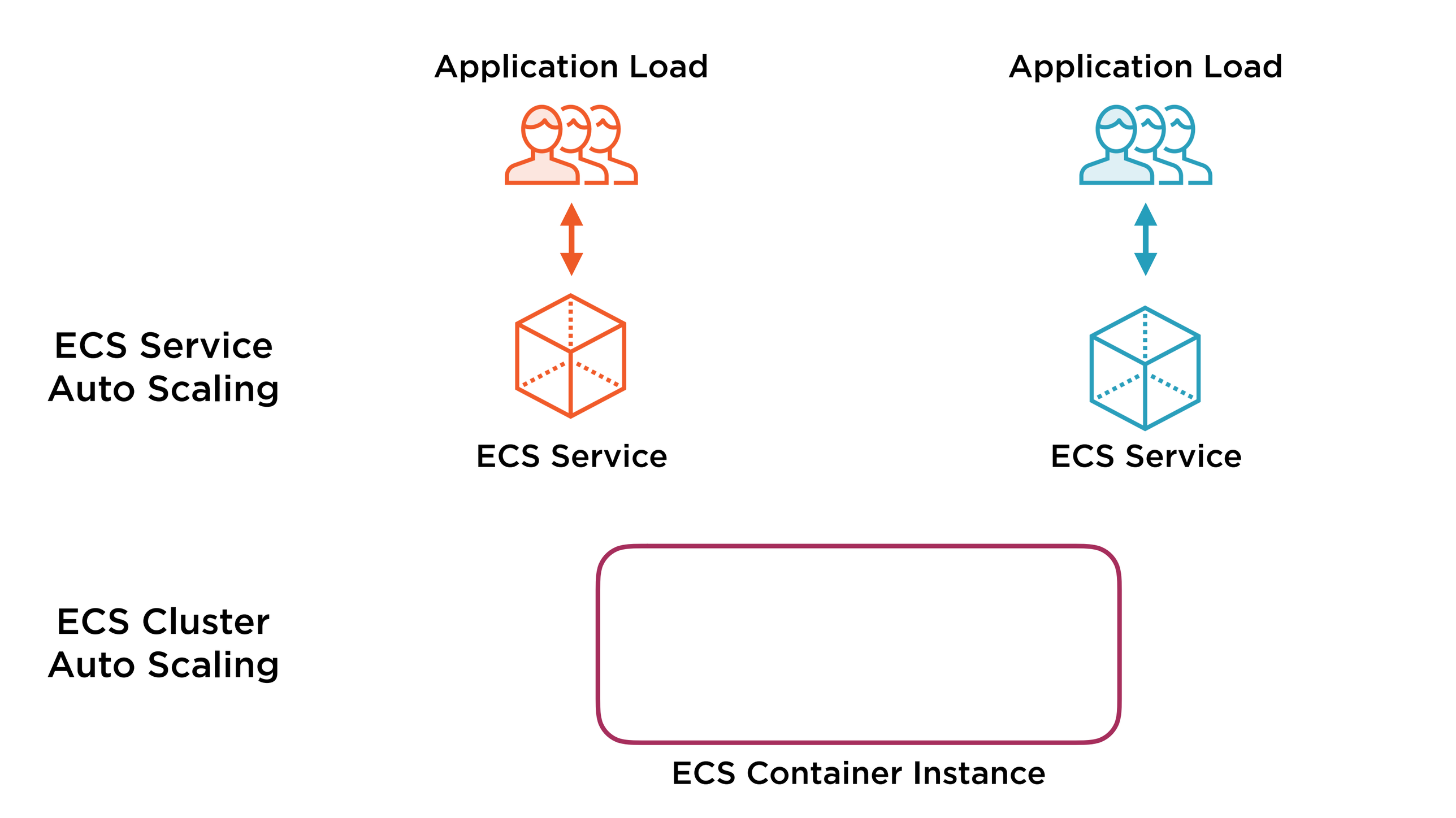
ECS Auto Scaling

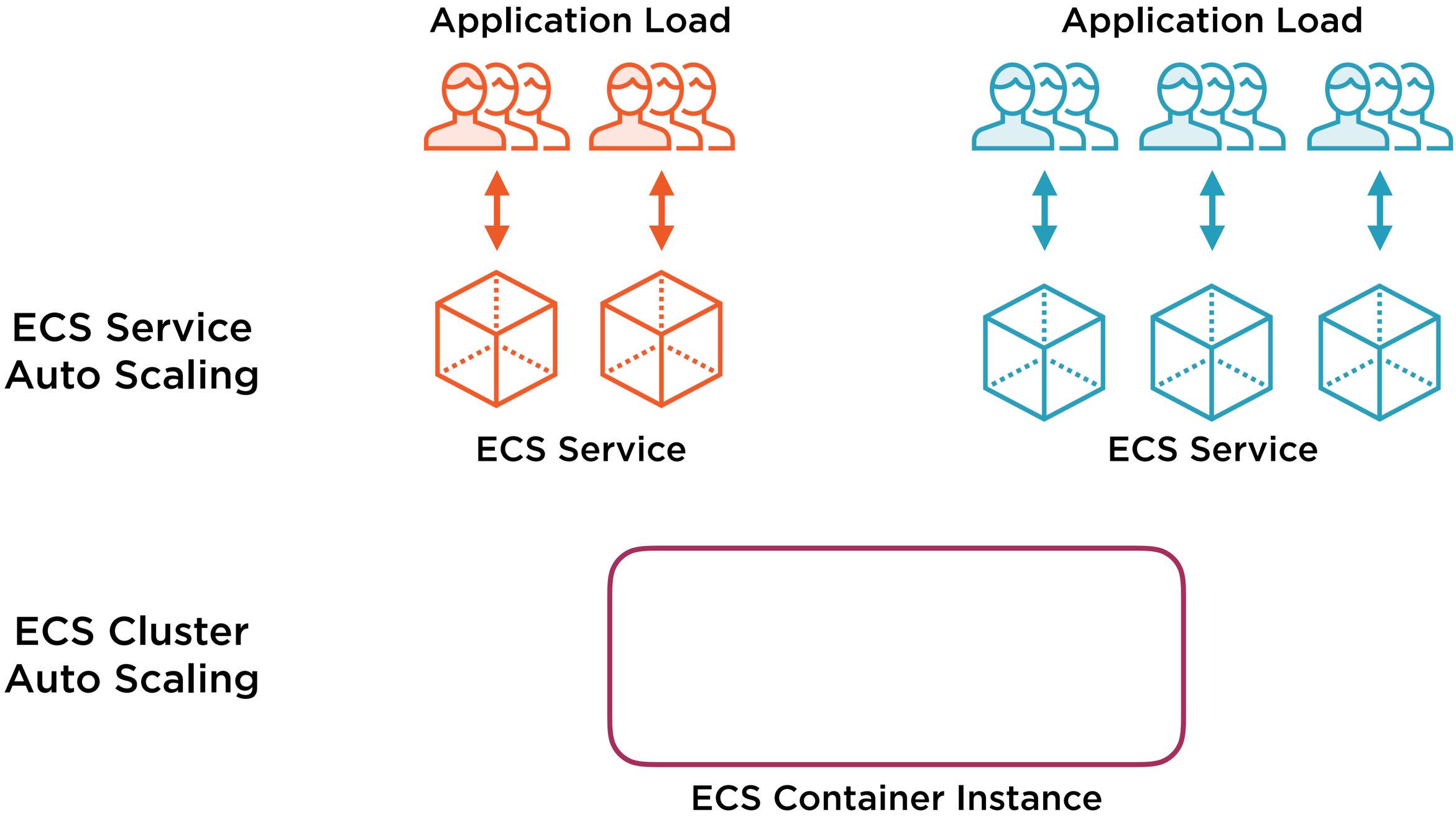
ECS Service Auto Scaling

Add bullets here

ECS Cluster Auto Scaling

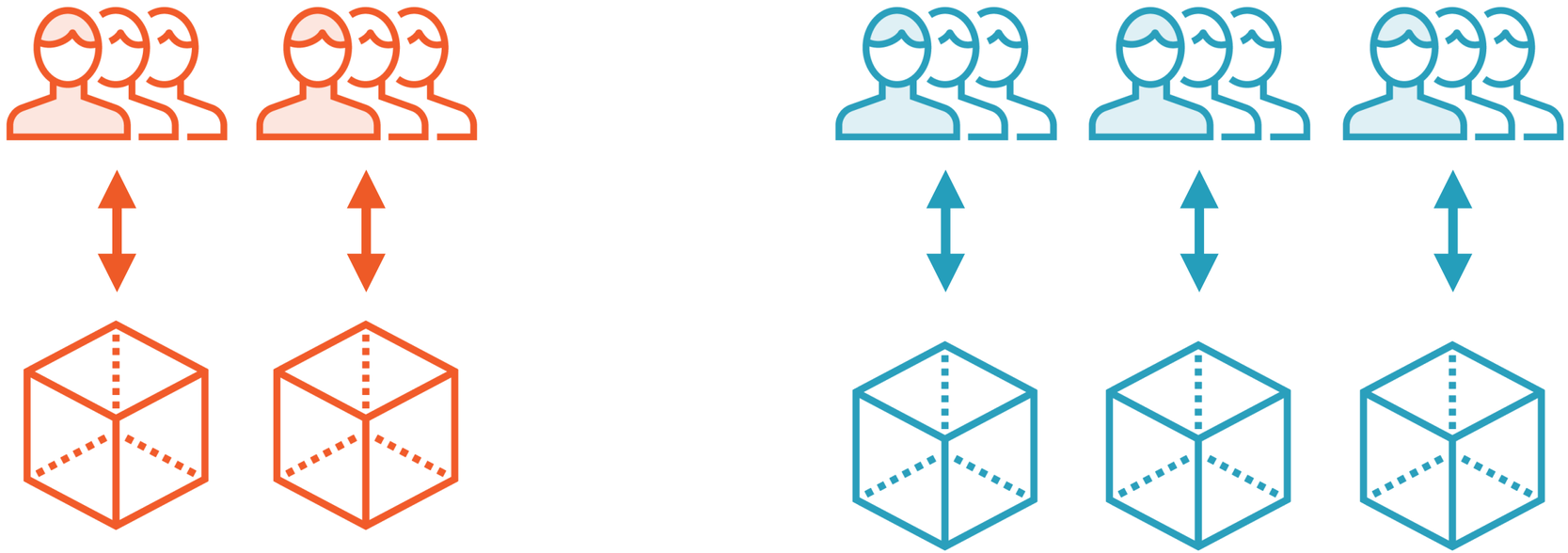
Add bullets here


Application Load
Application Load
ECS Service
Auto Scaling
ECS Service
ECS Service
ECS Cluster
Auto Scaling
ECS Container Instance


Application Load
Application Load
ECS Service Auto Scaling
ECS Service
ECS Service
ECS Cluster Auto Scaling
ECS Container Instance

Application Load

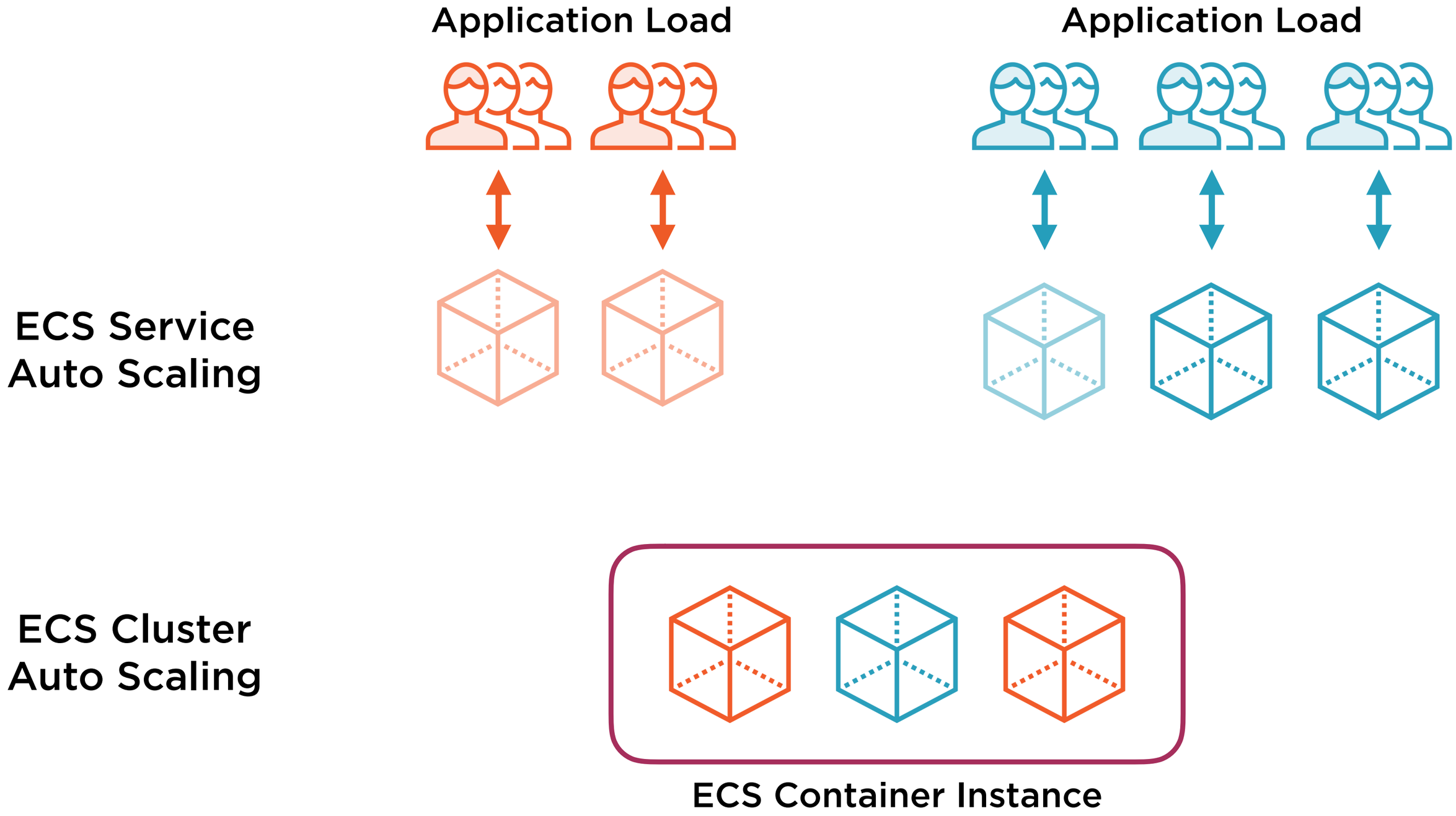
Application Load

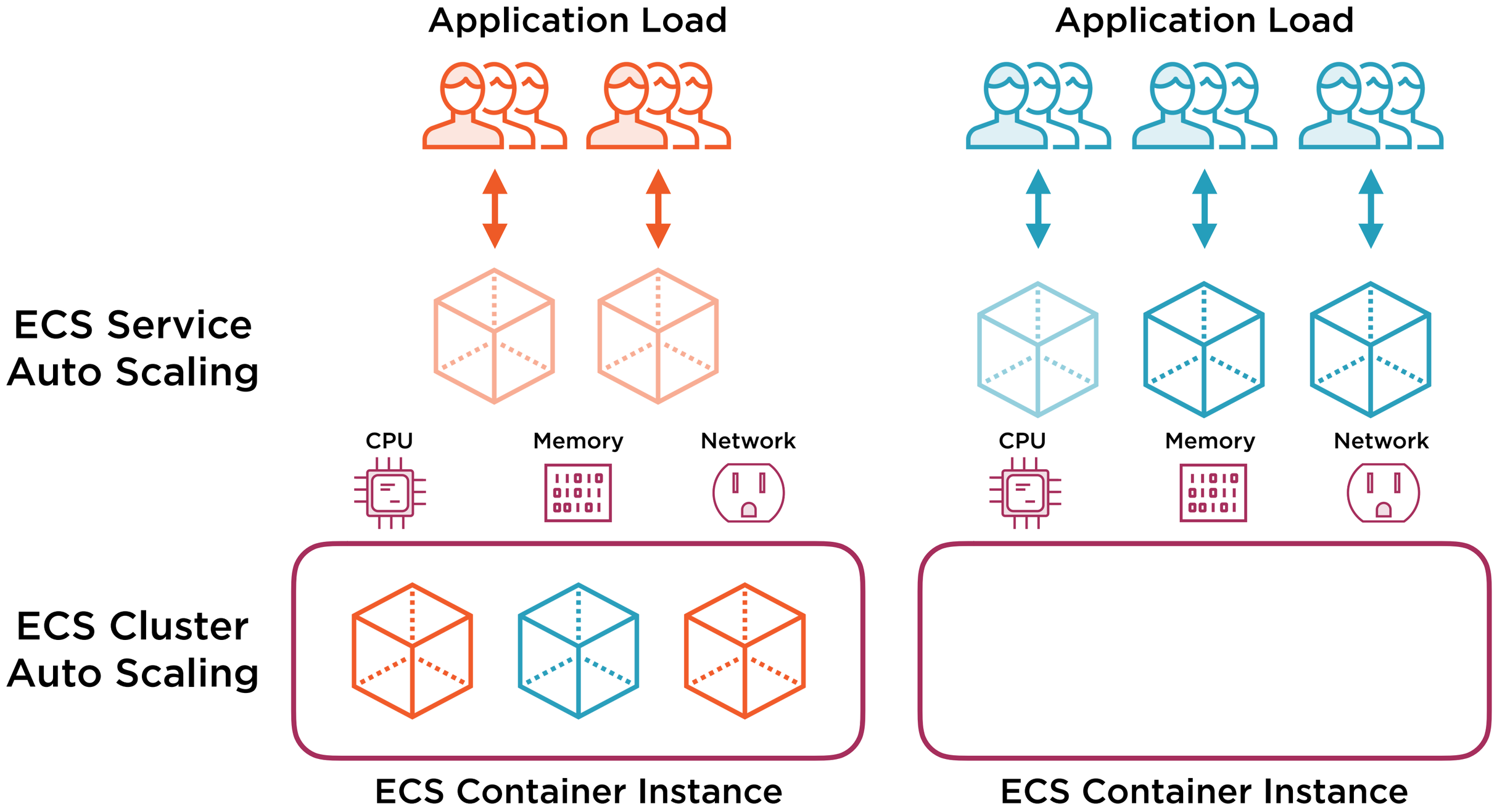


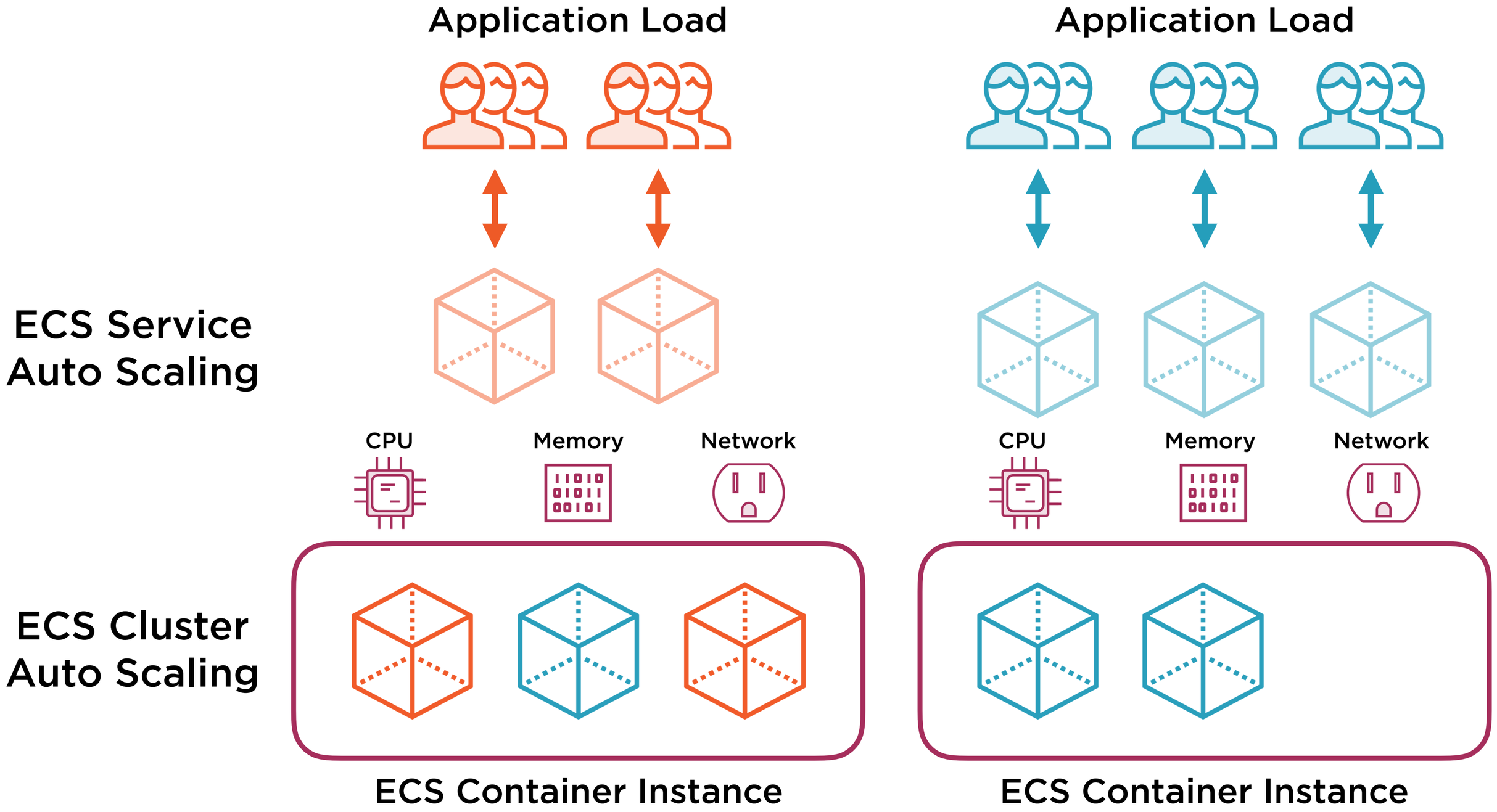
ECS Service
Auto Scaling

ECS Cluster
Auto Scaling

ECS Container Instance


Application Load
Application Load
ECS Service
Auto Scaling
ECS Cluster
Auto Scaling
ECS Container Instance

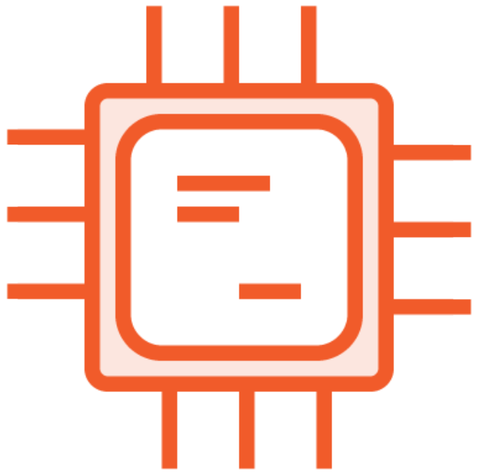

Application Load
Application Load
ECS Service
Auto Scaling
CPU
Memory
Network
CPU
Memory
Network
ECS Cluster
Auto Scaling
ECS Container Instance
ECS Container Instance


Application Load
Application Load
ECS Service
Auto Scaling
CPU
Memory
Network
CPU
Memory
Network
ECS Cluster
Auto Scaling
ECS Container Instance
ECS Container Instance

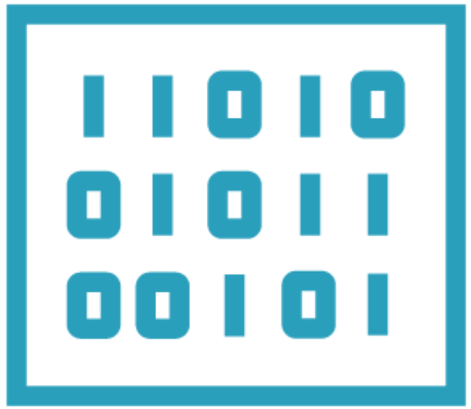
ECS Auto Scaling Challenges

Understanding ECS Resources

ECS Resource Types



CPU

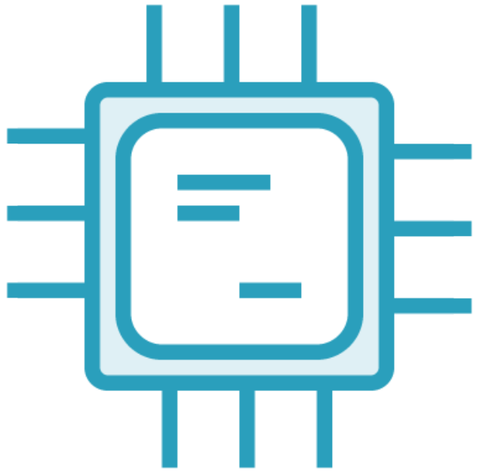


Memory

Network Ports

CPU Resources

CPU Reservations
CPU Units

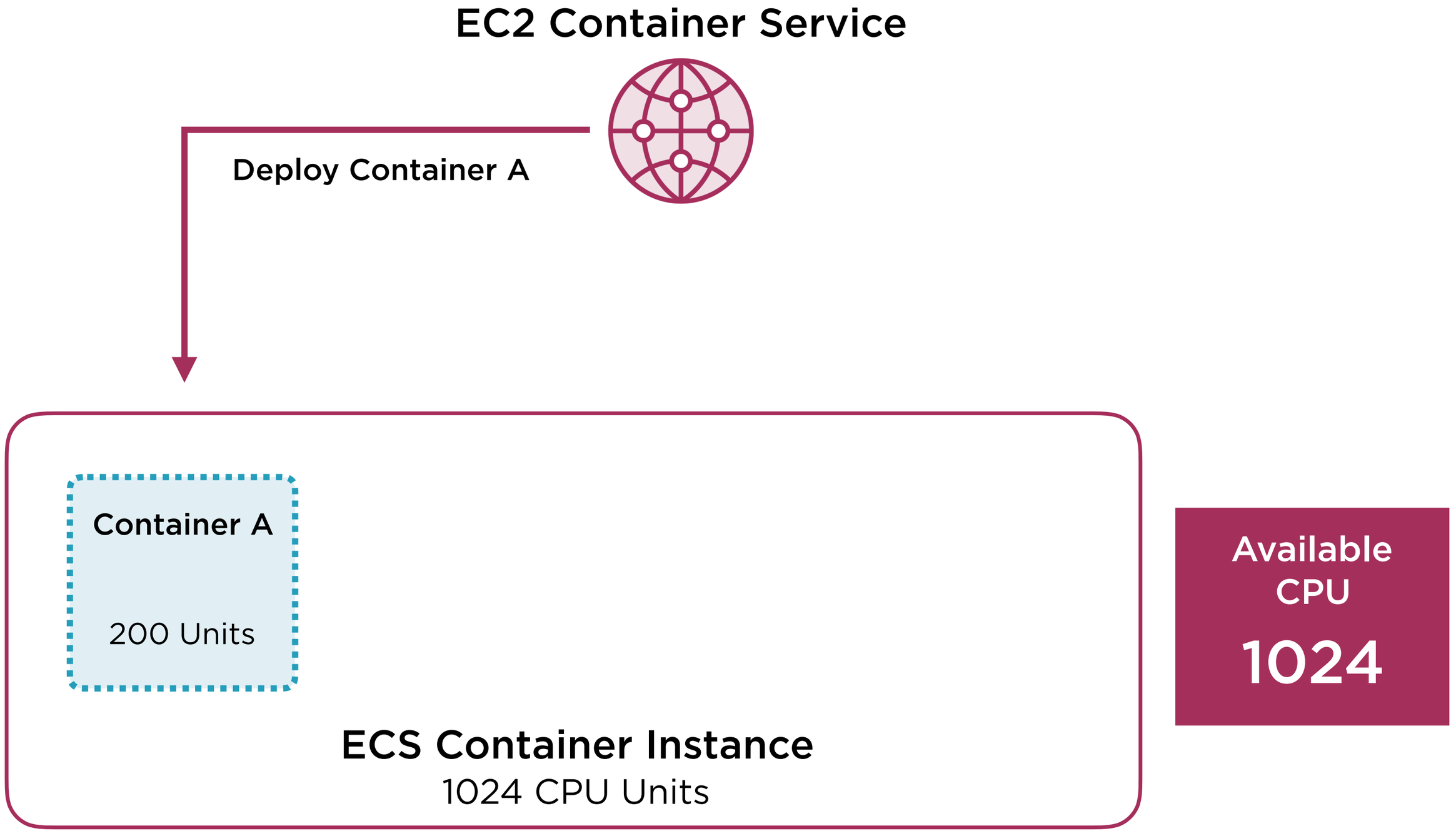


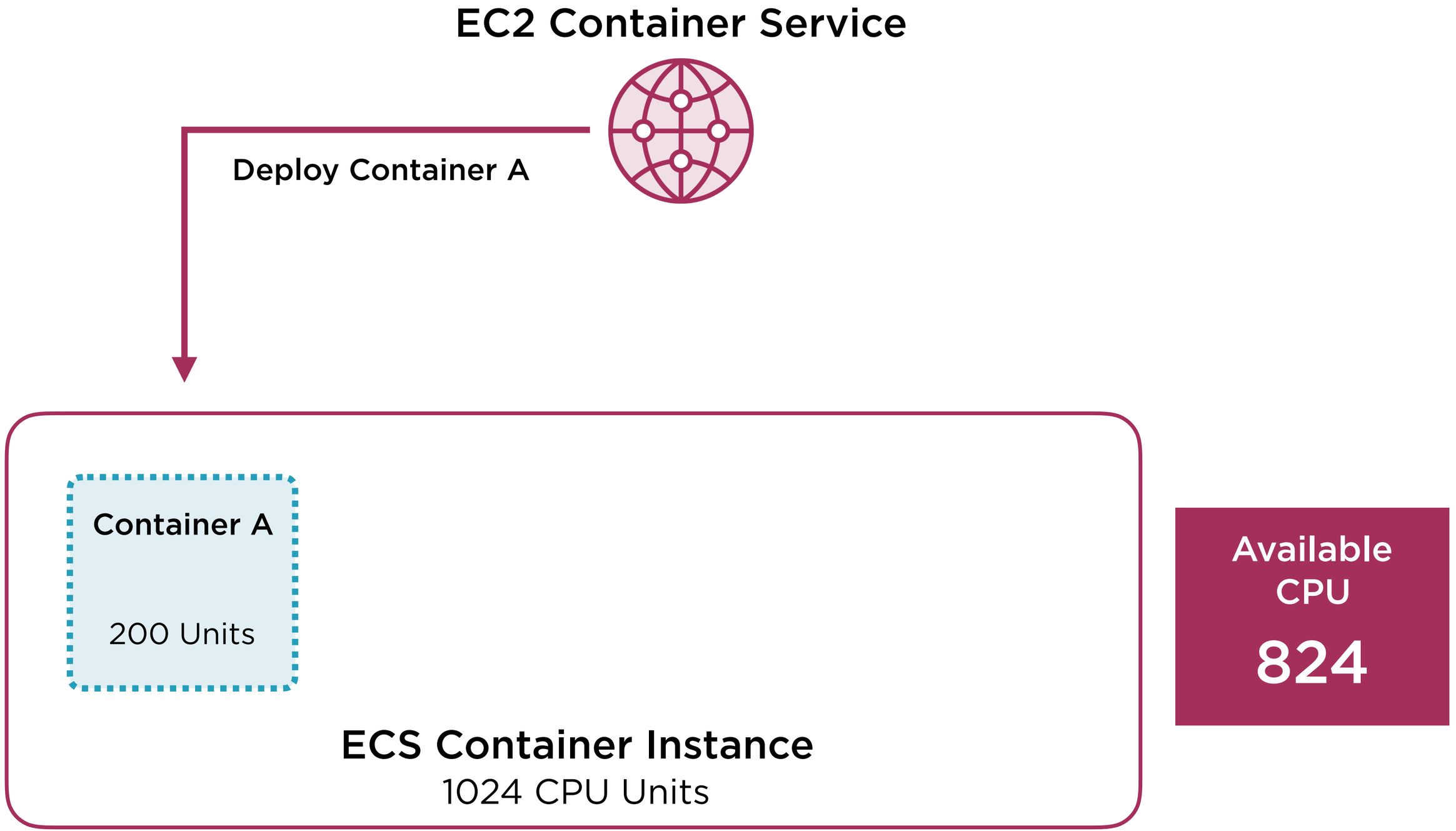
1024 Units per Core

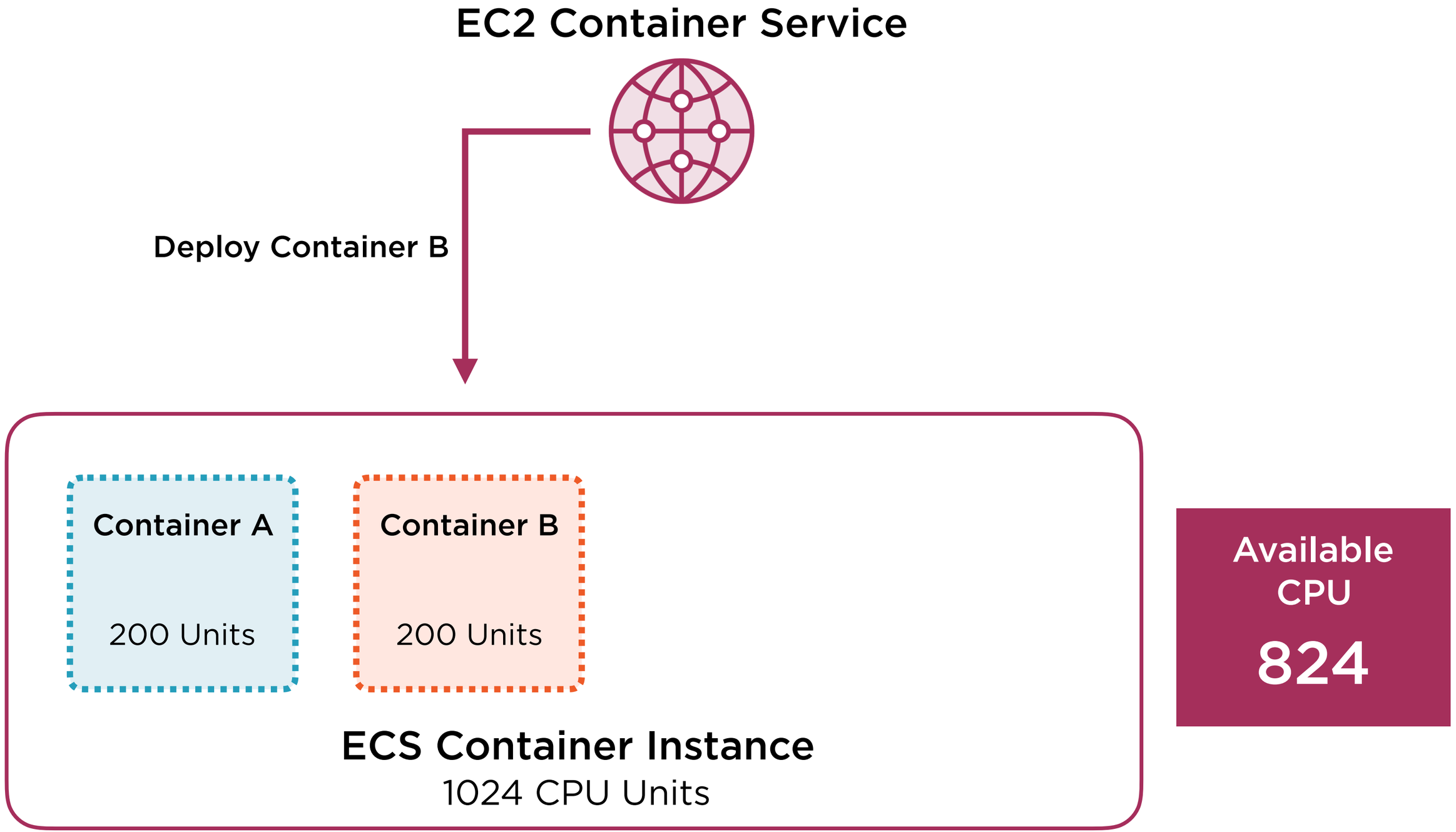
Used under CPU Load

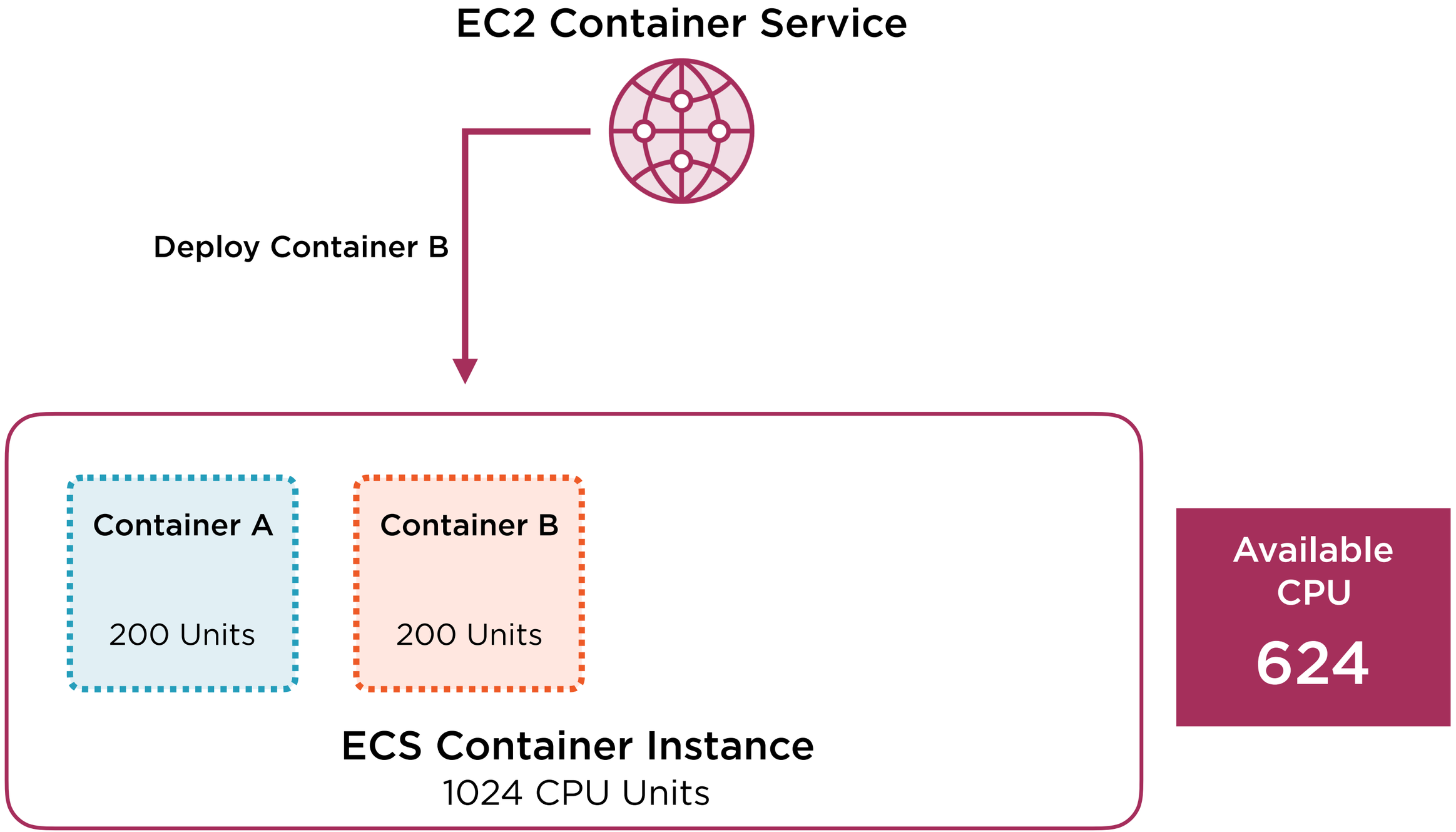
CPU Scheduling based upon CPU Units

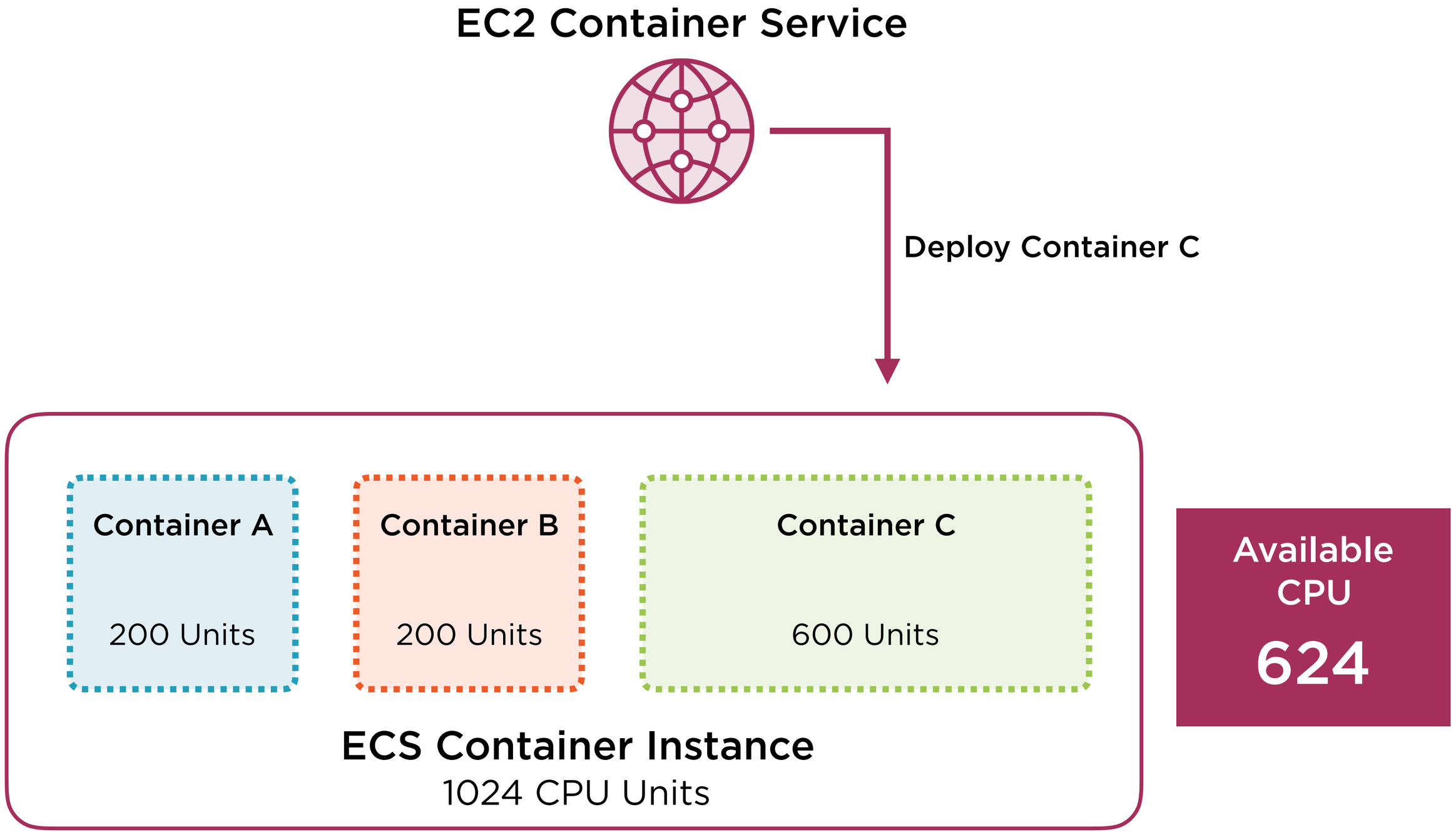
	Container A	Container B	Container C
Reservations	256 Units	256 Units	512 Units
Unit Ratio	256 / 1024	256 / 1024	512 / 1024
CPU Time	25%	25%	50%

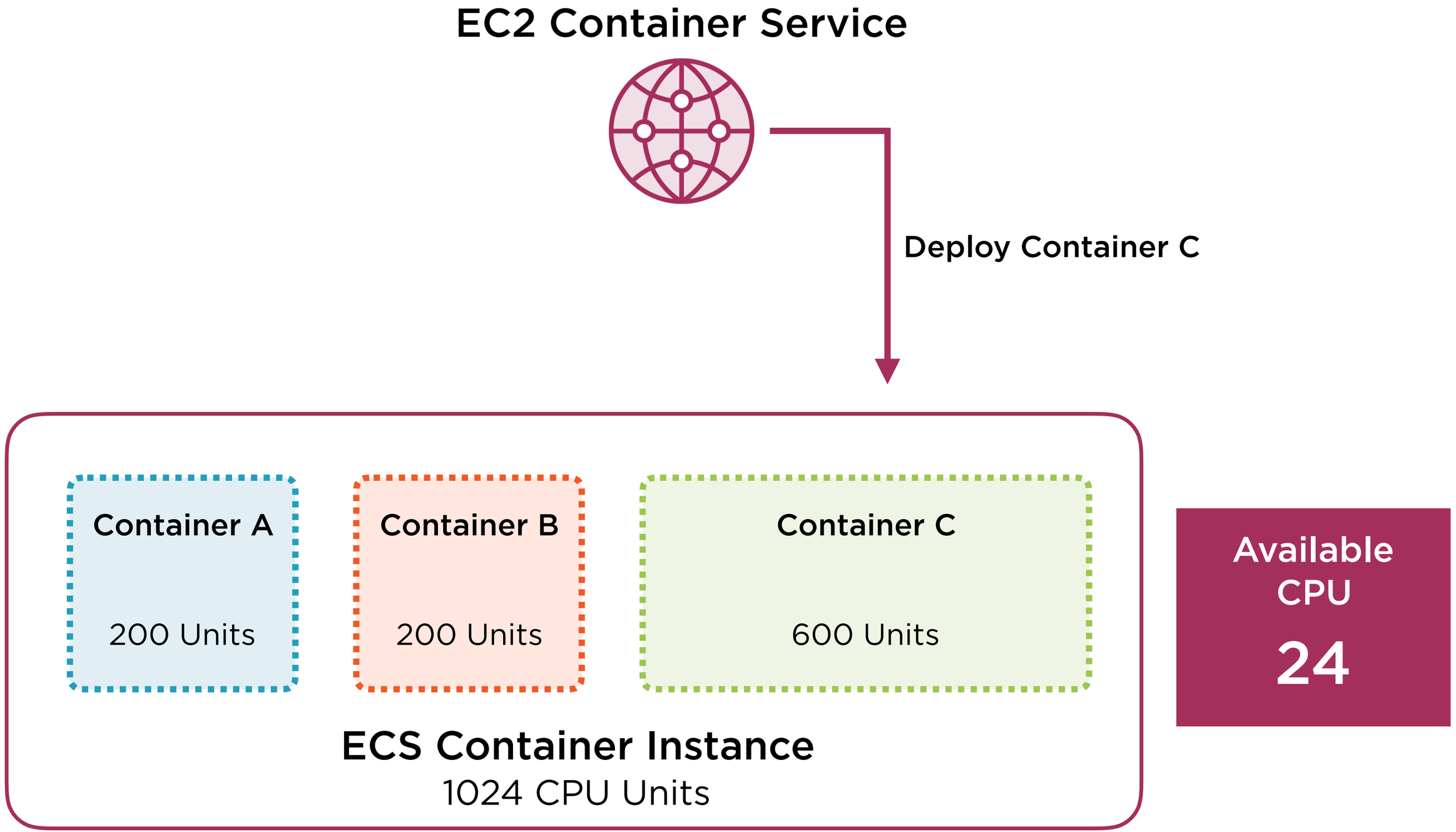

EC2 Container Service
Deploy Container A
Container A
200 Units
ECS Container Instance
1024 CPU Units
Available
CPU
1024

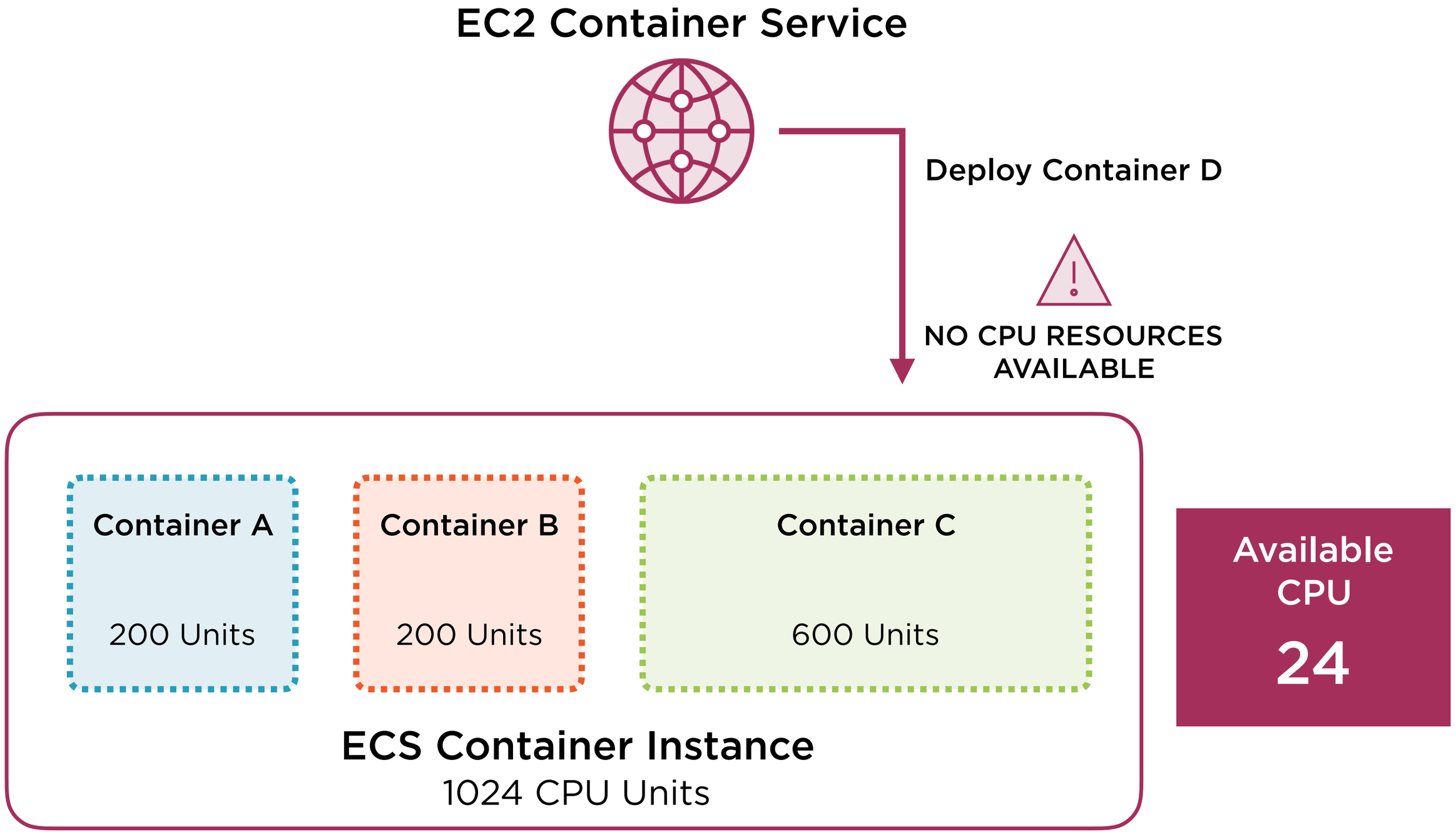

EC2 Container Service
Deploy Container A
Container A
200 Units
ECS Container Instance
1024 CPU Units
Available
CPU
824


EC2 Container Service
Deploy Container B
Container A
200 Units
Container B
200 Units
ECS Container Instance
1024 CPU Units
Available
CPU
824


EC2 Container Service
Deploy Container B
Container A
200 Units
Container B
200 Units
ECS Container Instance
1024 CPU Units
Available
CPU
624


EC2 Container Service
Deploy Container C
Container A
200 Units
Container B
200 Units
Container C
600 Units
ECS Container Instance
1024 CPU Units
Available CPU
624


EC2 Container Service
Deploy Container C
Container A
200 Units
Container B
200 Units
Container C
600 Units
ECS Container Instance
1024 CPU Units
Available CPU
24

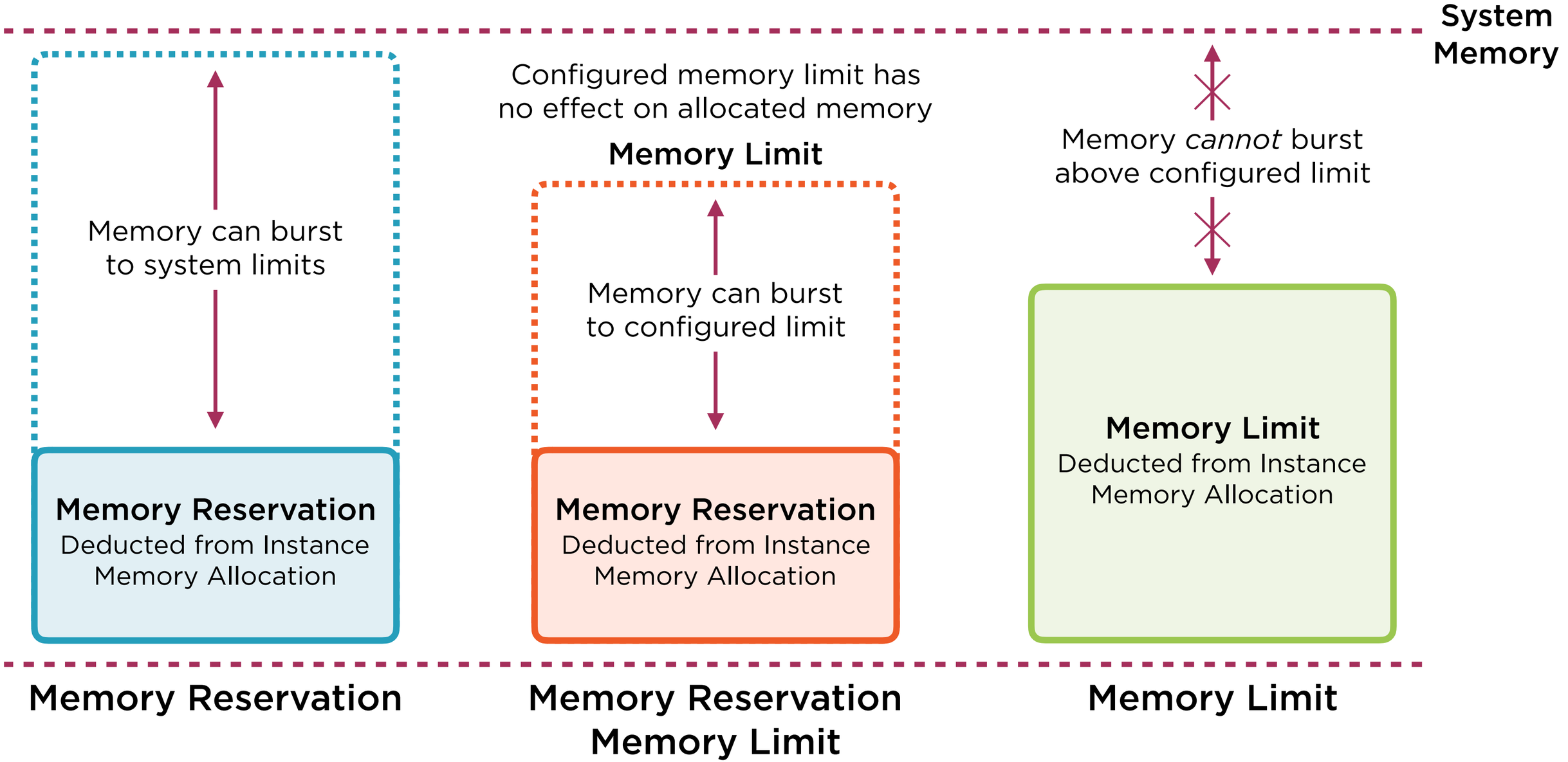

EC2 Container Service
Deploy Container D
NO CPU RESOURCES AVAILABLE
Container A
200 Units
Container B
200 Units
Container C
600 Units
ECS Container Instance
1024 CPU Units
Available CPU
24

Memory Resources

Memory Reservations

Memory Limits

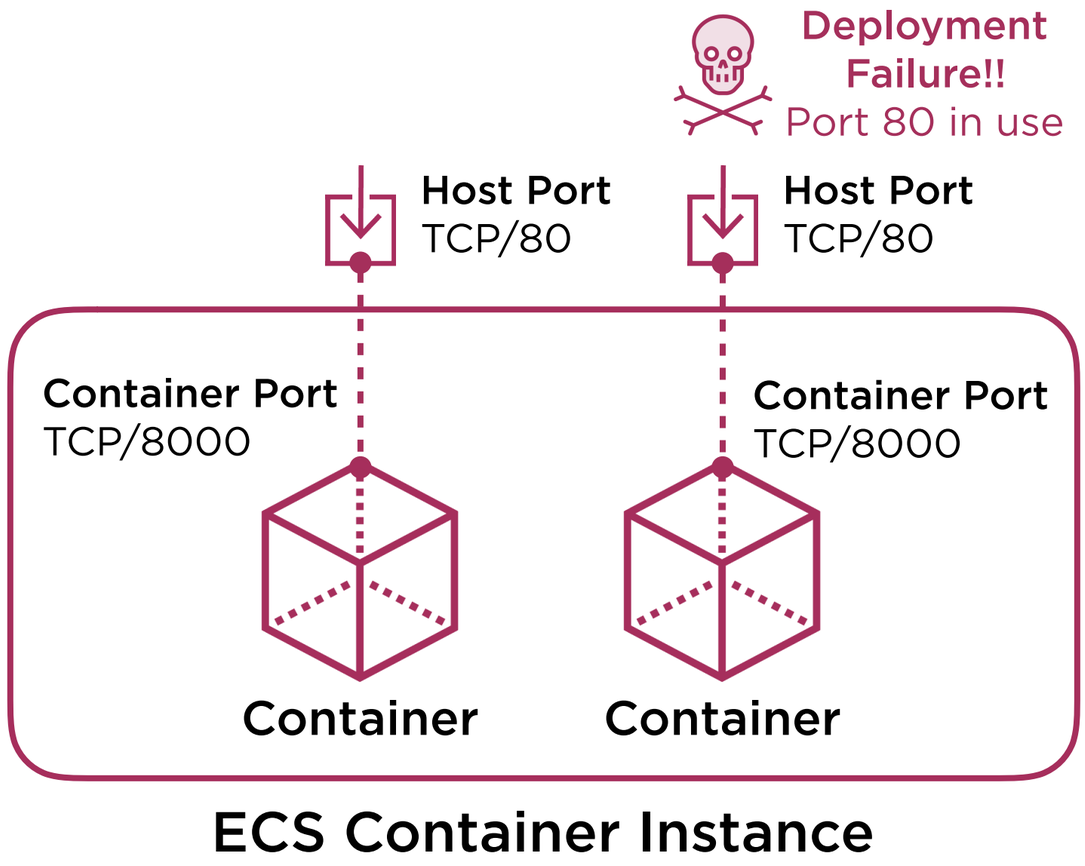
Memory Resources



Network Port Resources

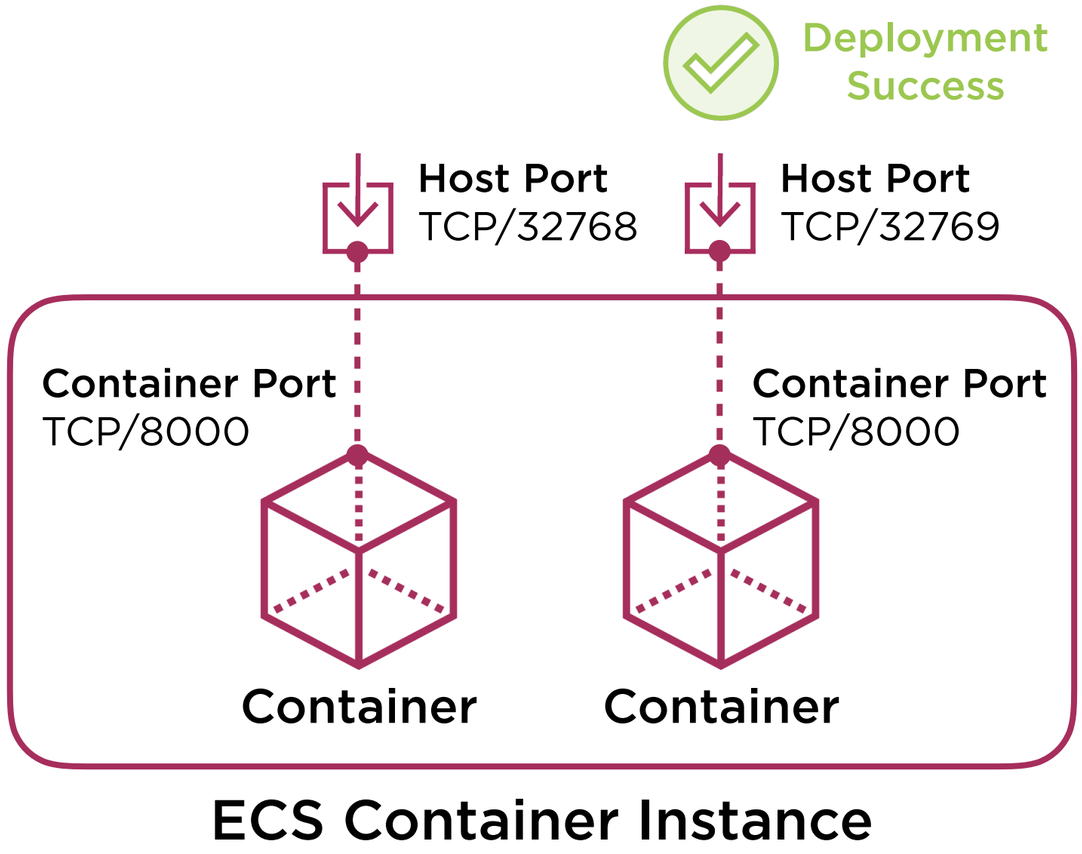
Static Port Mapping

Only one container per port per instance



Dynamic Port Mapping

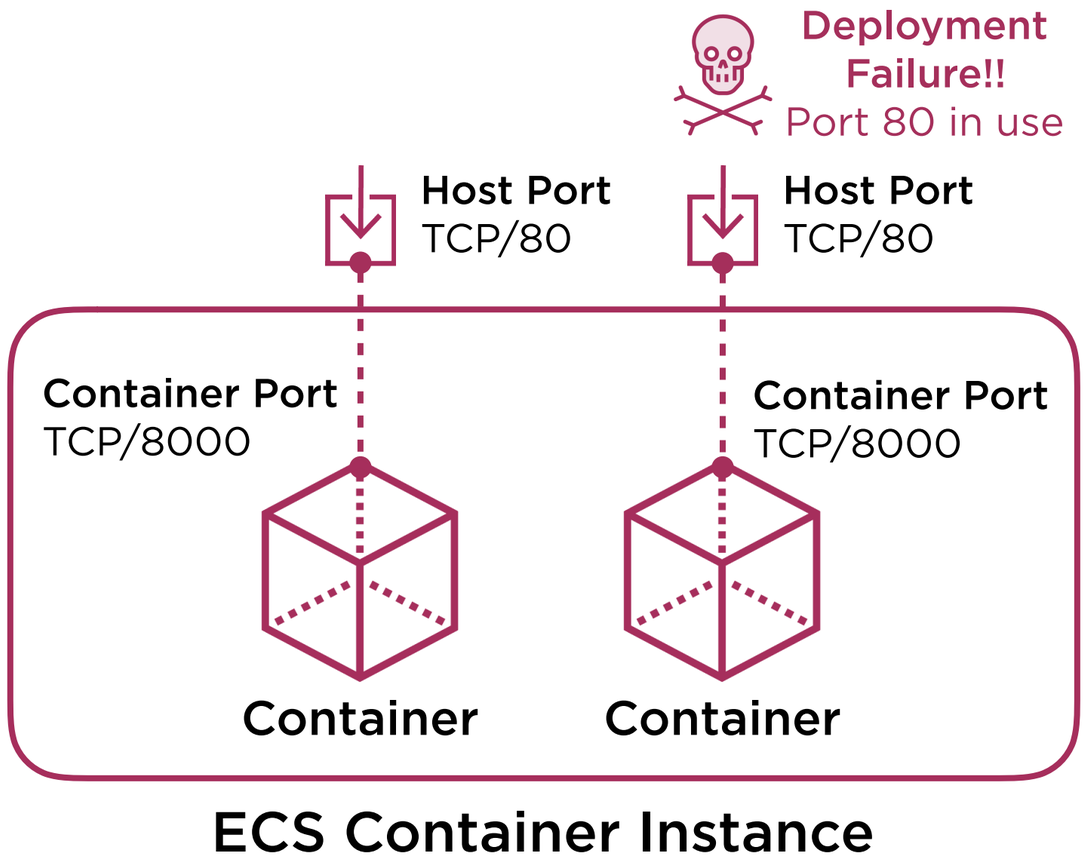
Multiple containers per port per instance



Dynamic Port Mapping

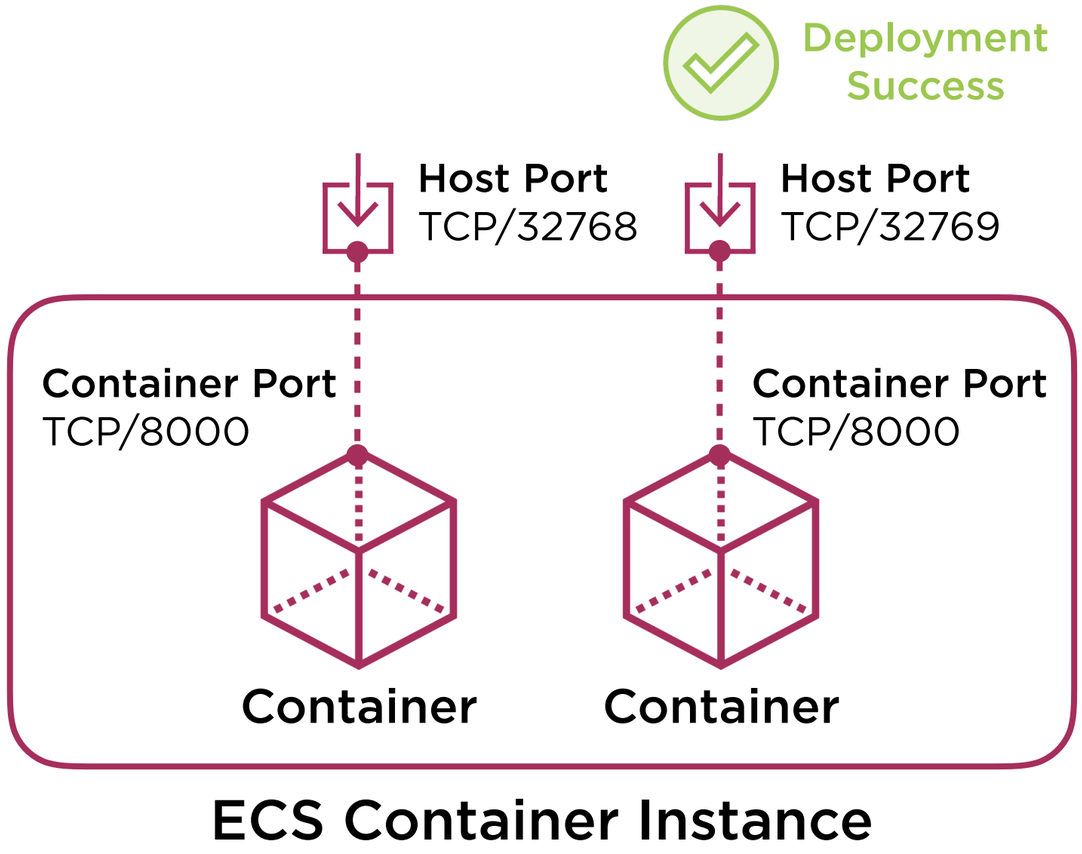
Static Port Mapping

Only one container per port per instance

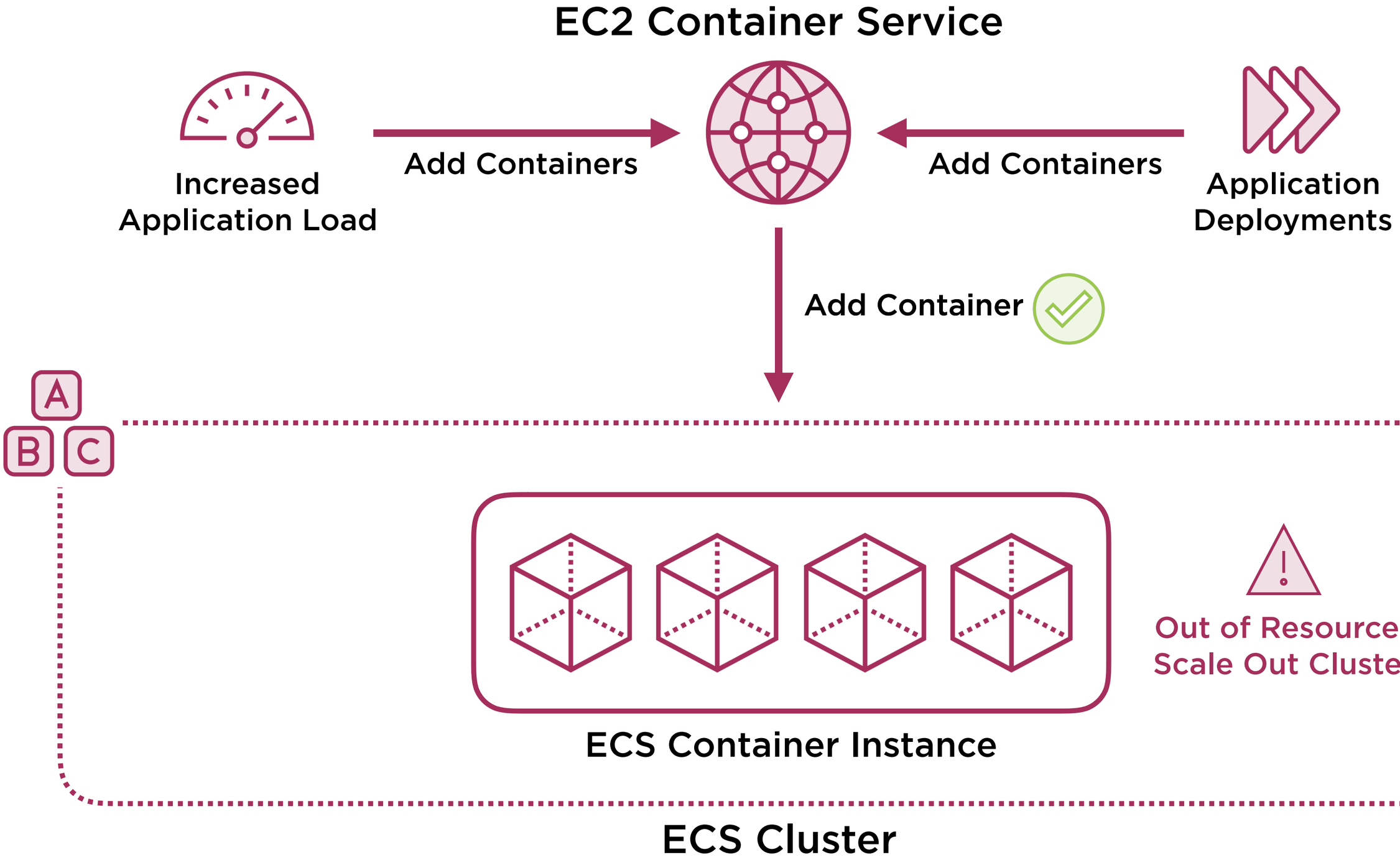


Dynamic Port Mapping

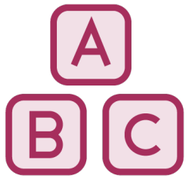
Multiple containers per port per instance

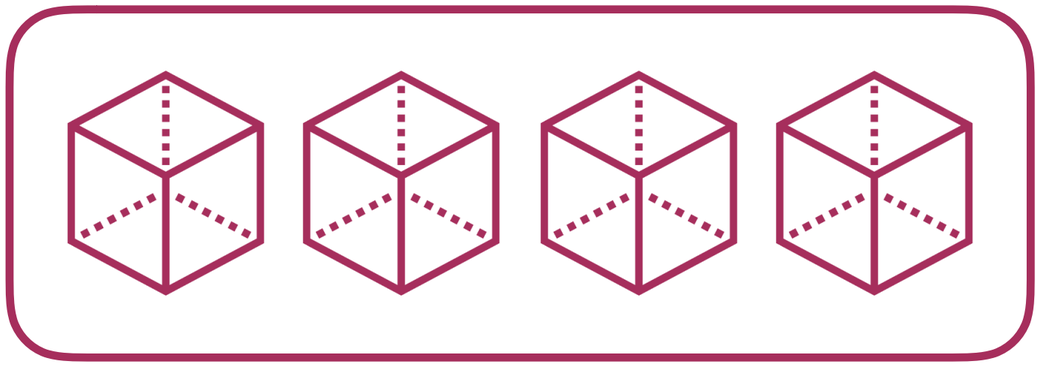


ECS Capacity Management


EC2 Container Service
Increased Application Load
Add Containers
Add Containers
Application Deployments
Add Container
A
B
C
ECS Container Instance
Out of Resources
Scale Out Cluster
ECS Cluster

EC2 Container Service

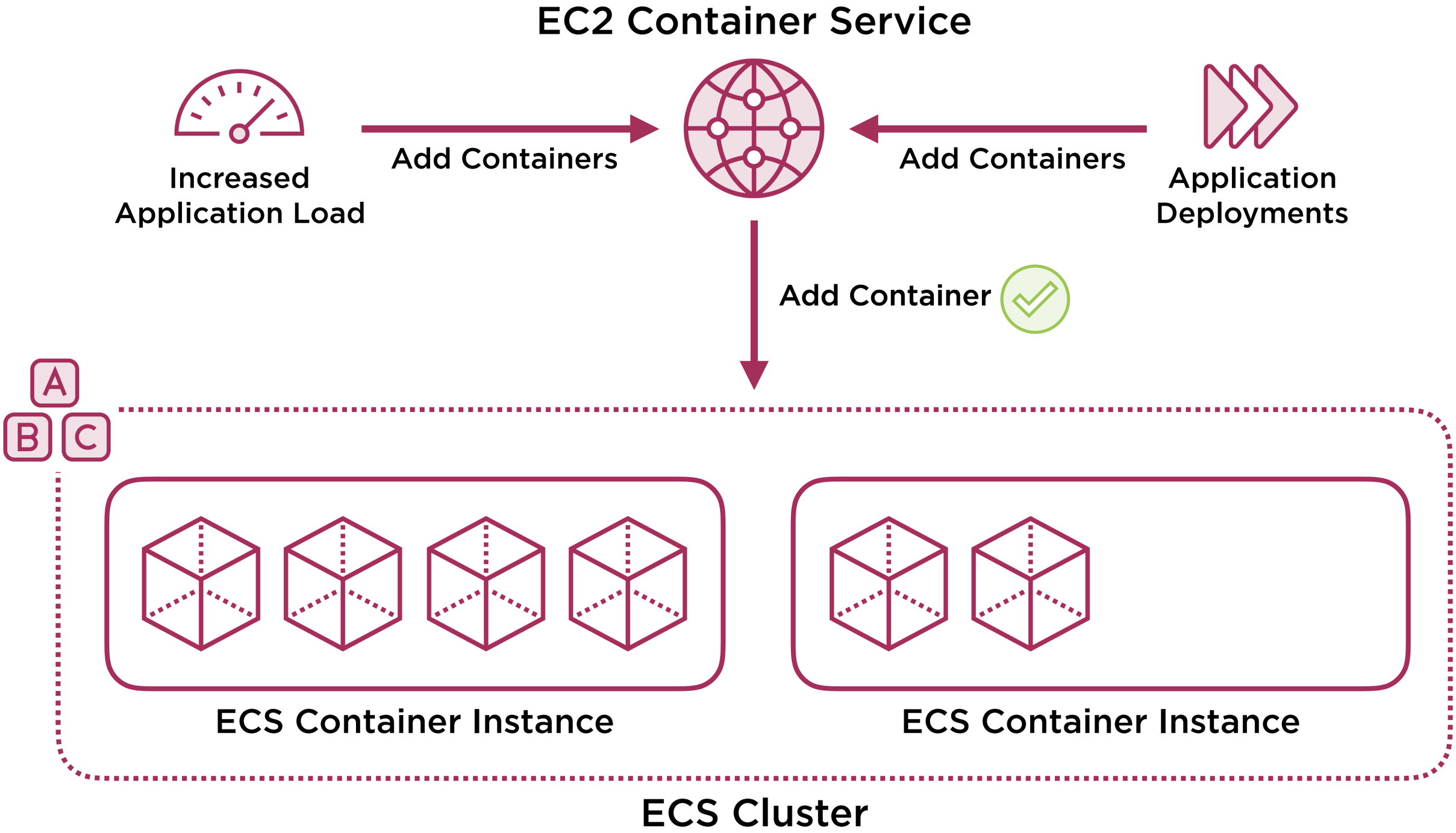




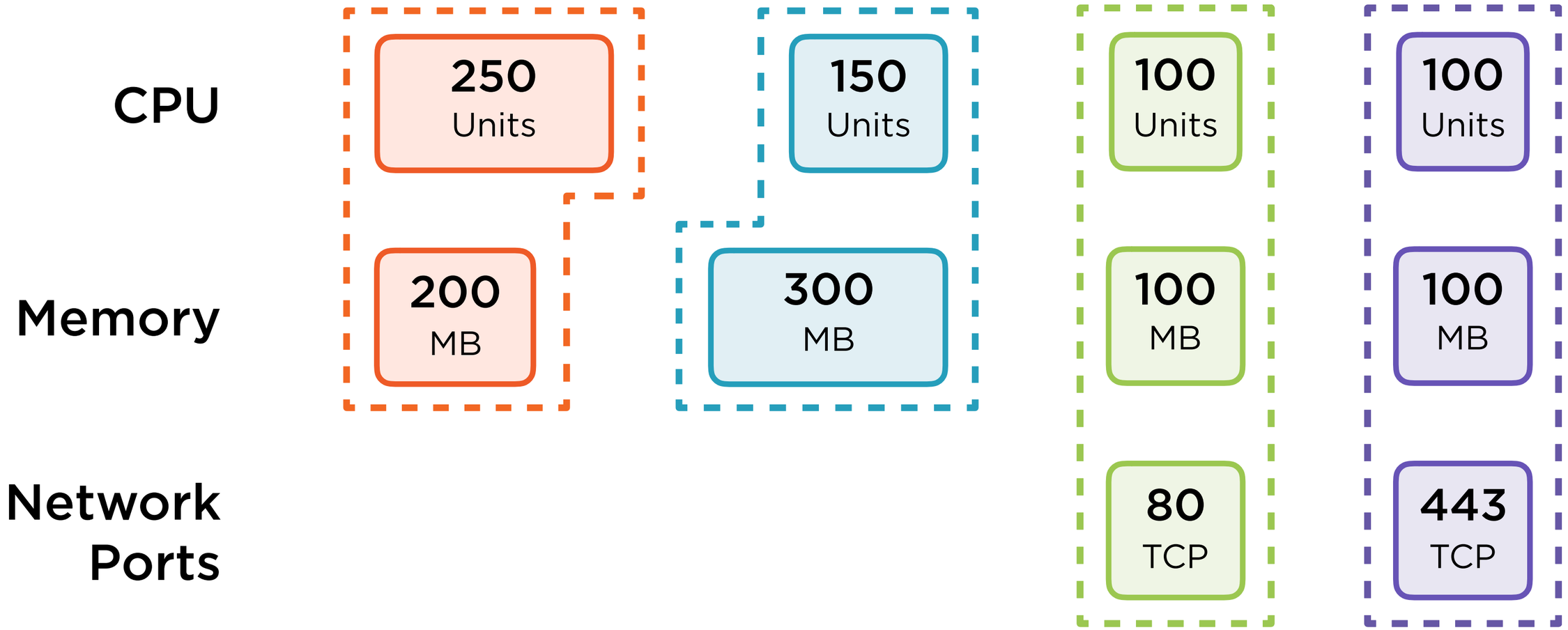
Out of Resources
Scale Out Cluster

ECS Container Instance

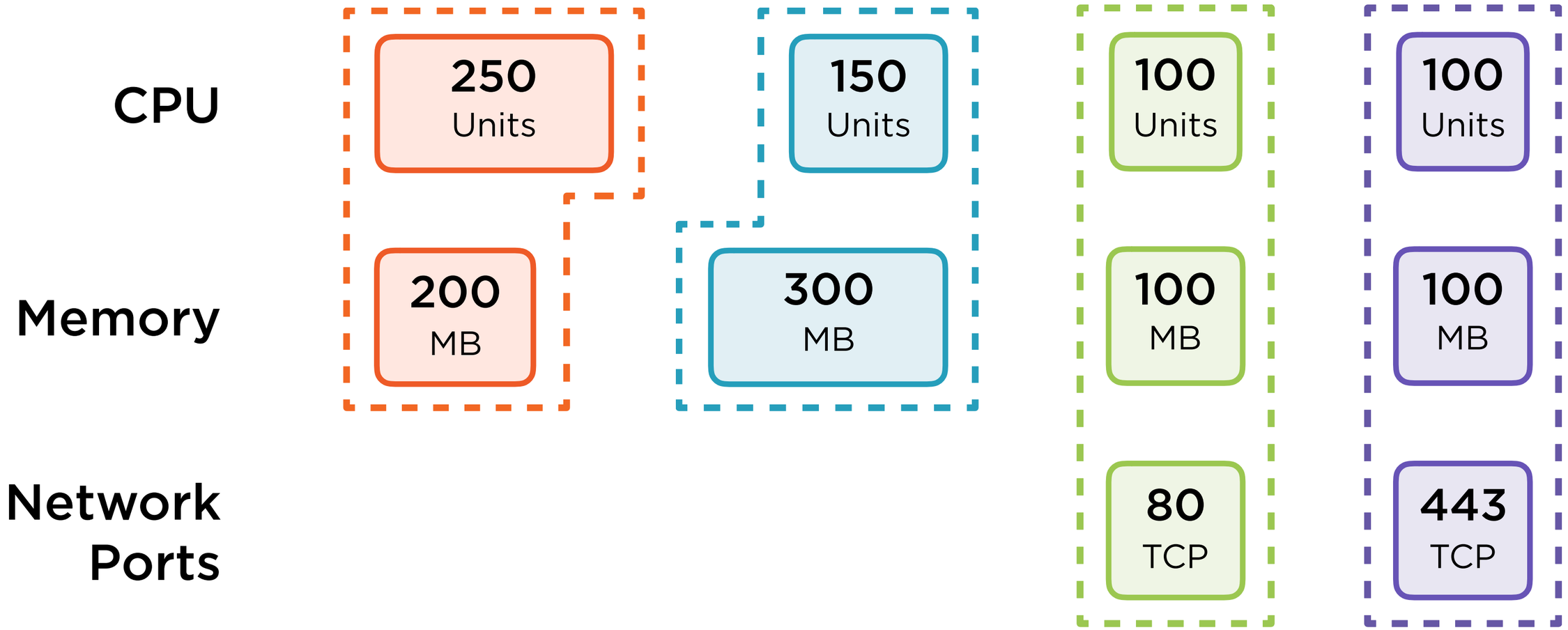
ECS Cluster

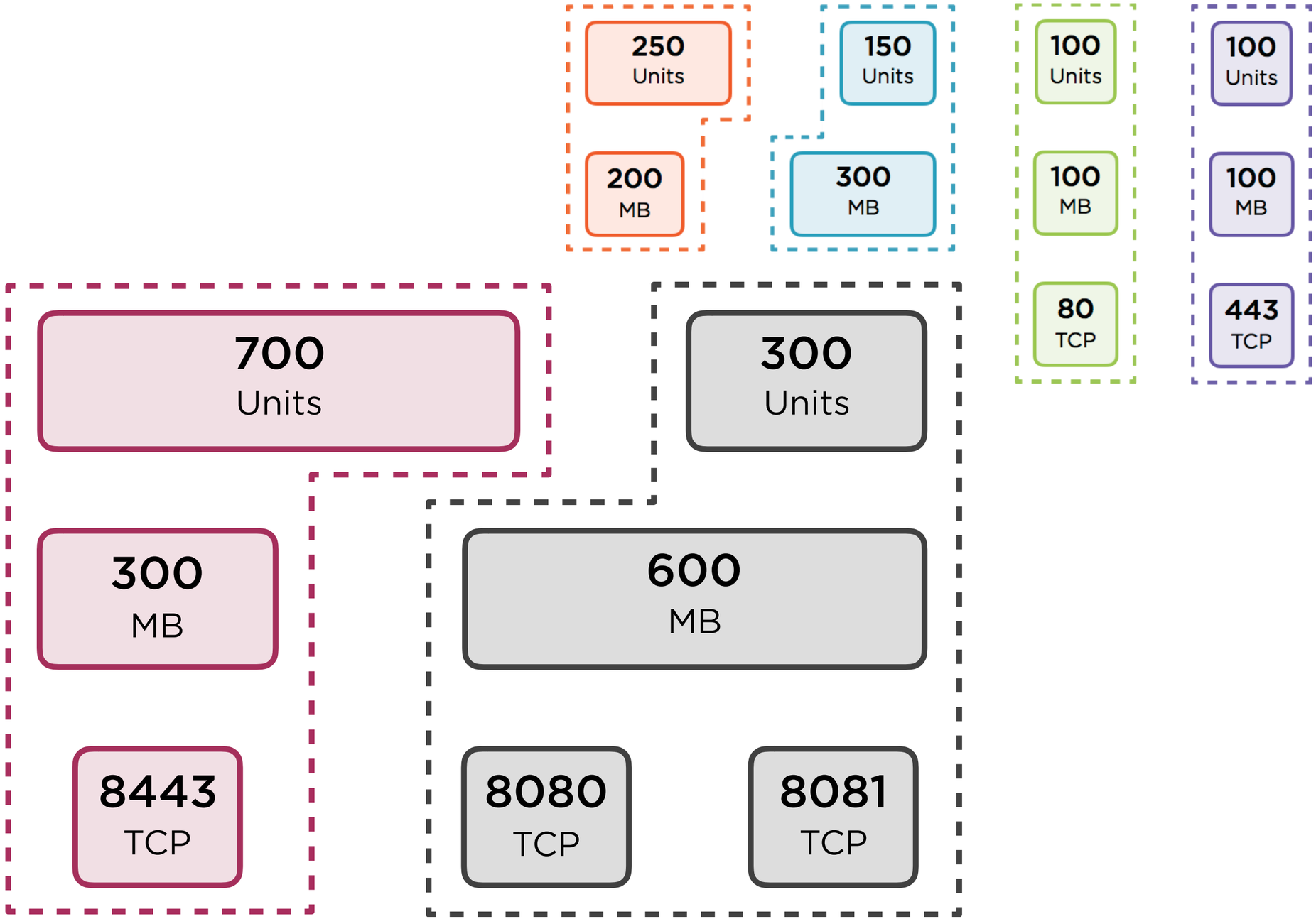

EC2 Container Service
Increased Application Load
Add Containers
Add Containers
Application Deployments
Add Container
A
B
C
ECS Container Instance
ECS Container Instance
ECS Cluster

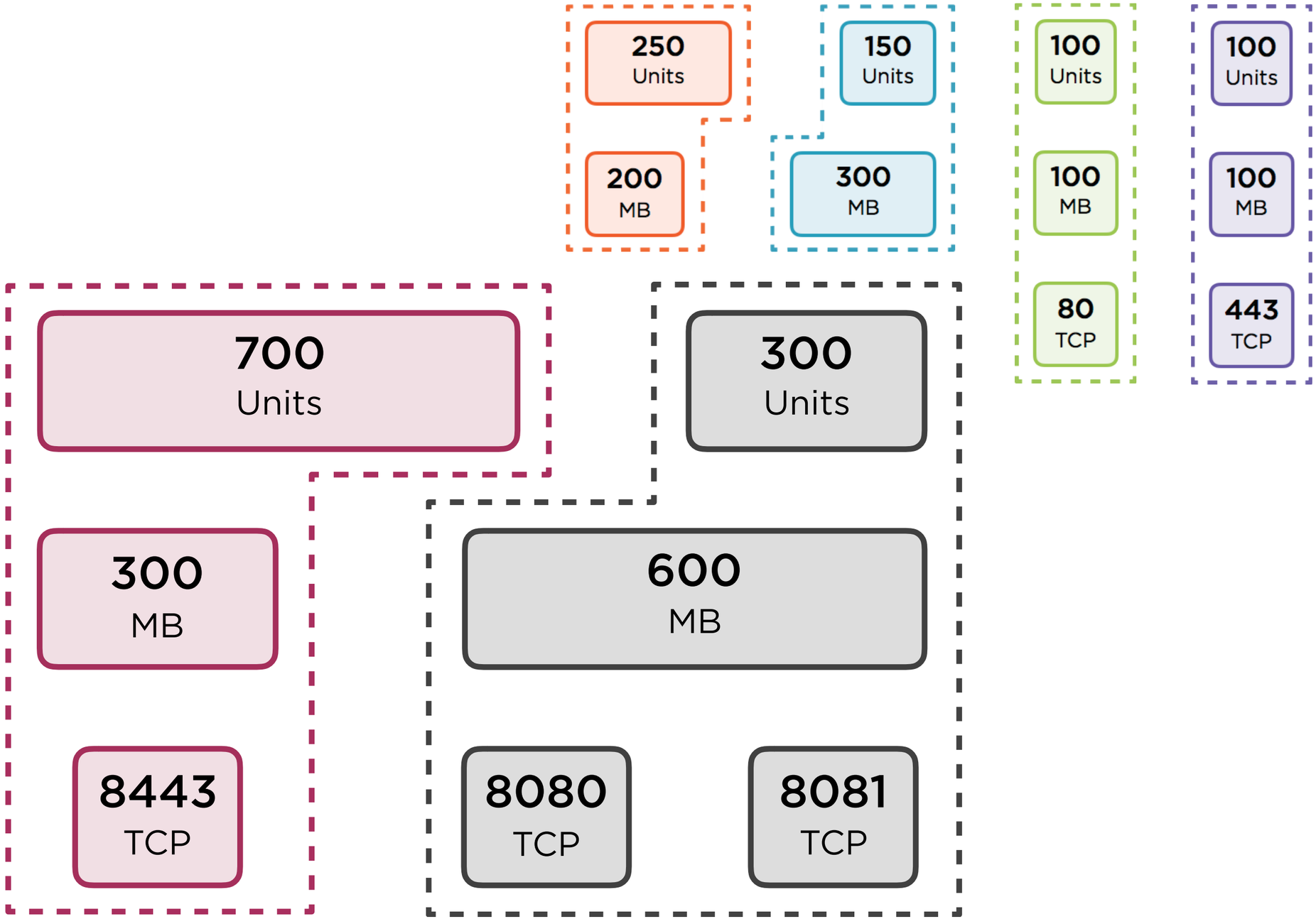
Dealing with Resources for Multiple Containers

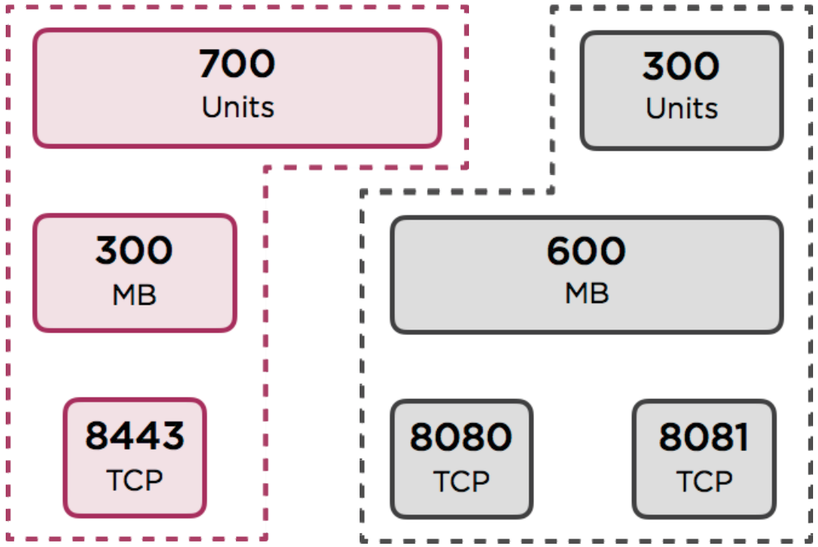


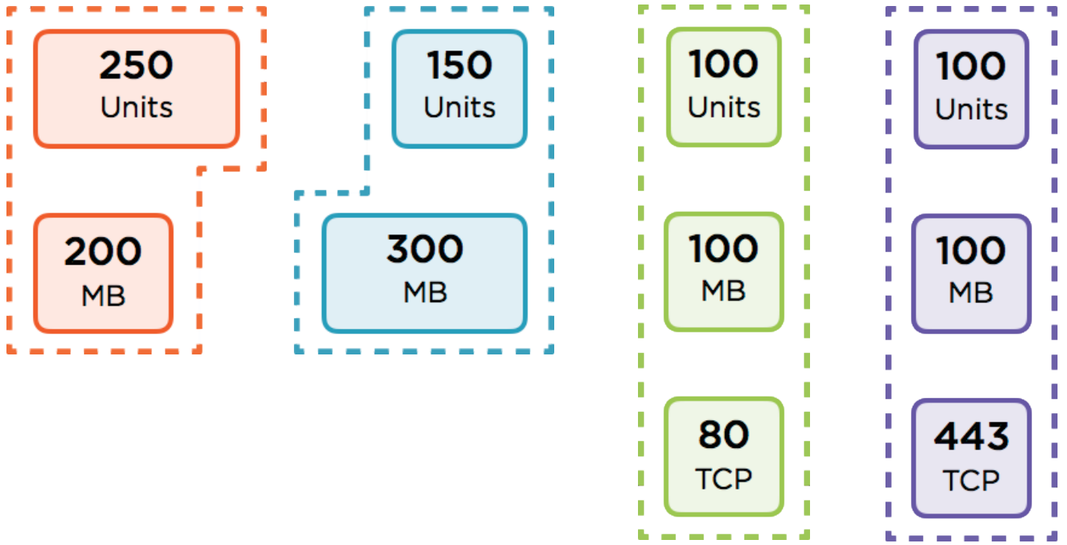
Dealing with Resources for Multiple Containers




250
Units
200
MB
150
Units
300
MB
100
Units
100
MB
80
TCP
100
Units
100
MB
443
TCP
700
Units
300
Units
300
MB
600
MB
8443
TCP
8080
TCP
8081
TCP


250
Units
200
MB
150
Units
300
MB
100
Units
100
MB
80
TCP
100
Units
100
MB
443
TCP
700
Units
300
Units
300
MB
600
MB
8443
TCP
8080
TCP
8081
TCP

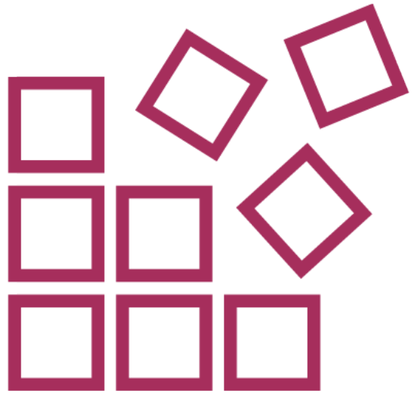




ECS Container Instance

ECS Container Instance

Calculating ECS Capacity



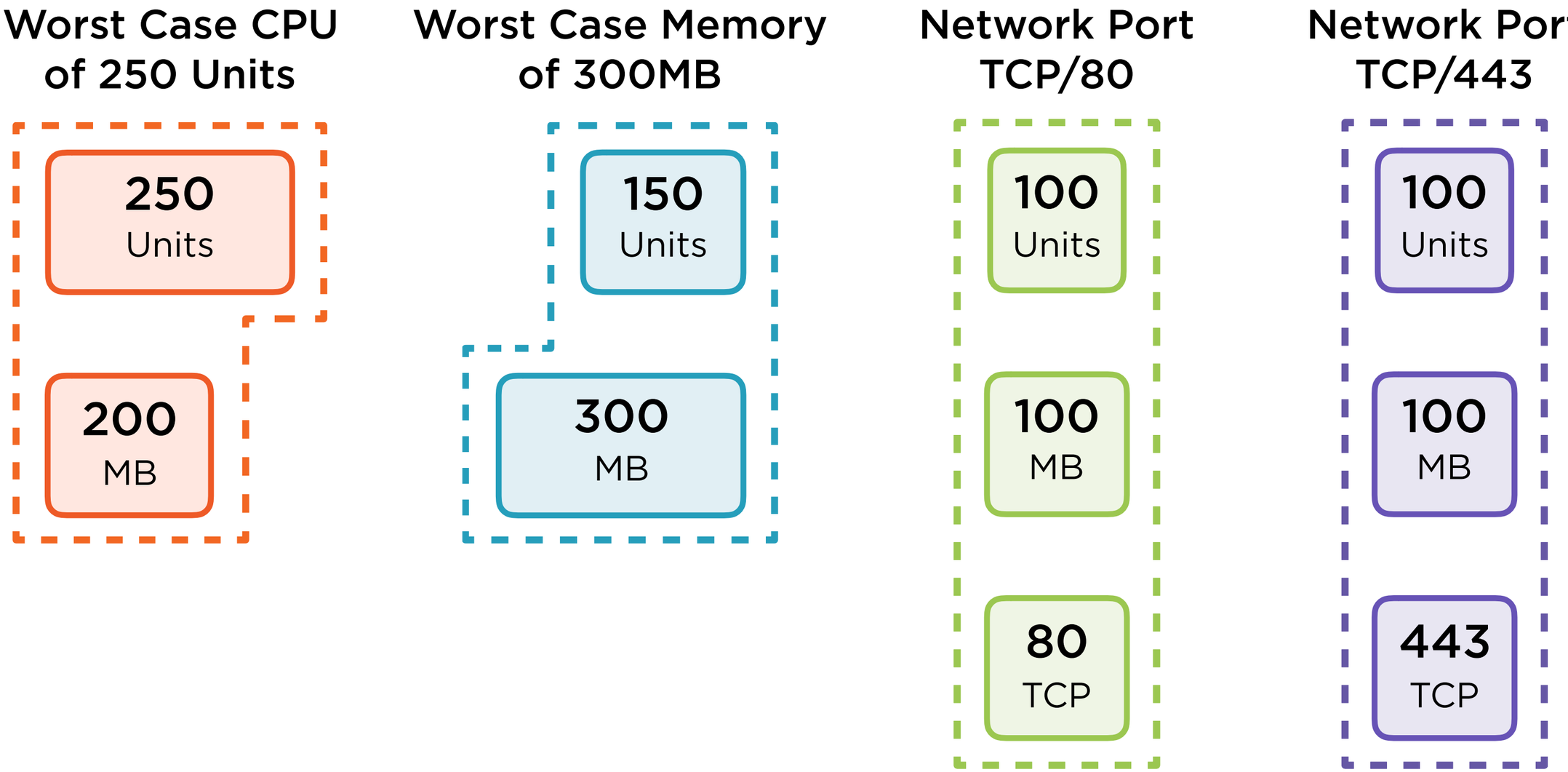
1. Calculate Individual Resource Capacity

- Determine container that requires largest amount of each resource
- Calculate current capacity in the cluster expressed in terms of containers
- E.g. I can currently schedule two of the containers that consume the most CPU in my cluster
- Repeat for each type of resource

2. Calculate Overall Capacity

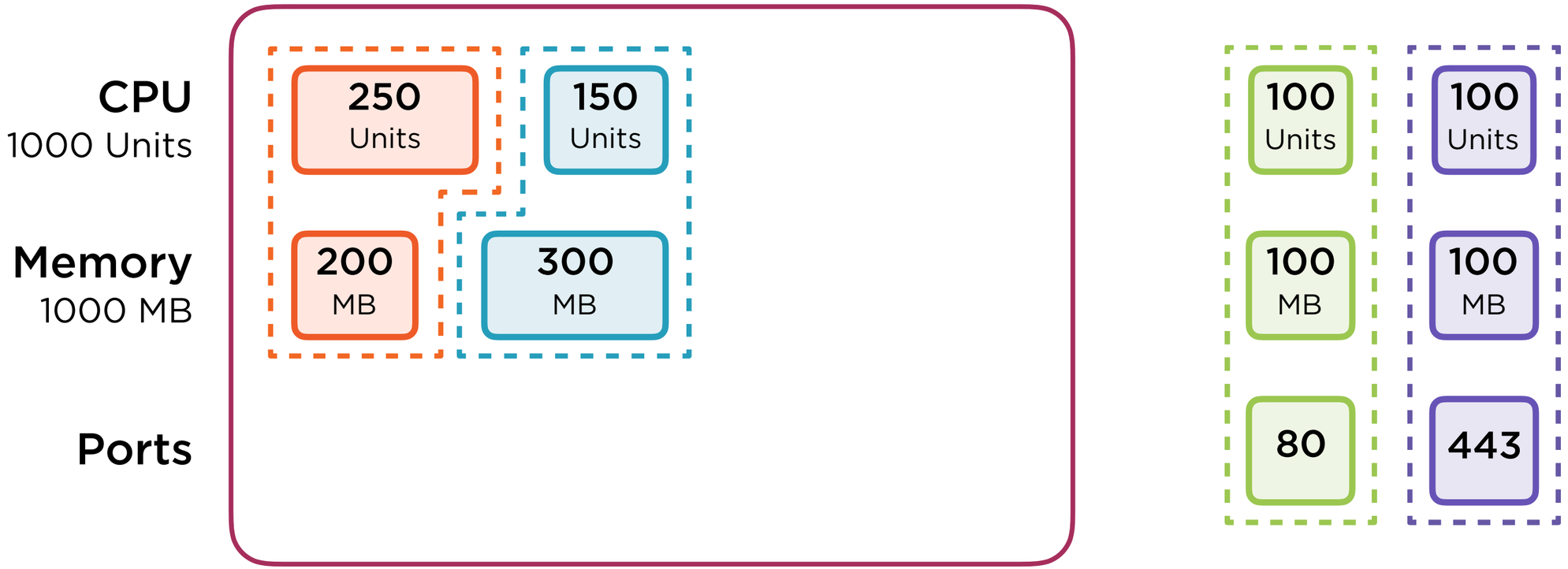
- Take the lowest value of each of the current individual resource capacities

Let's play Container Tetris!



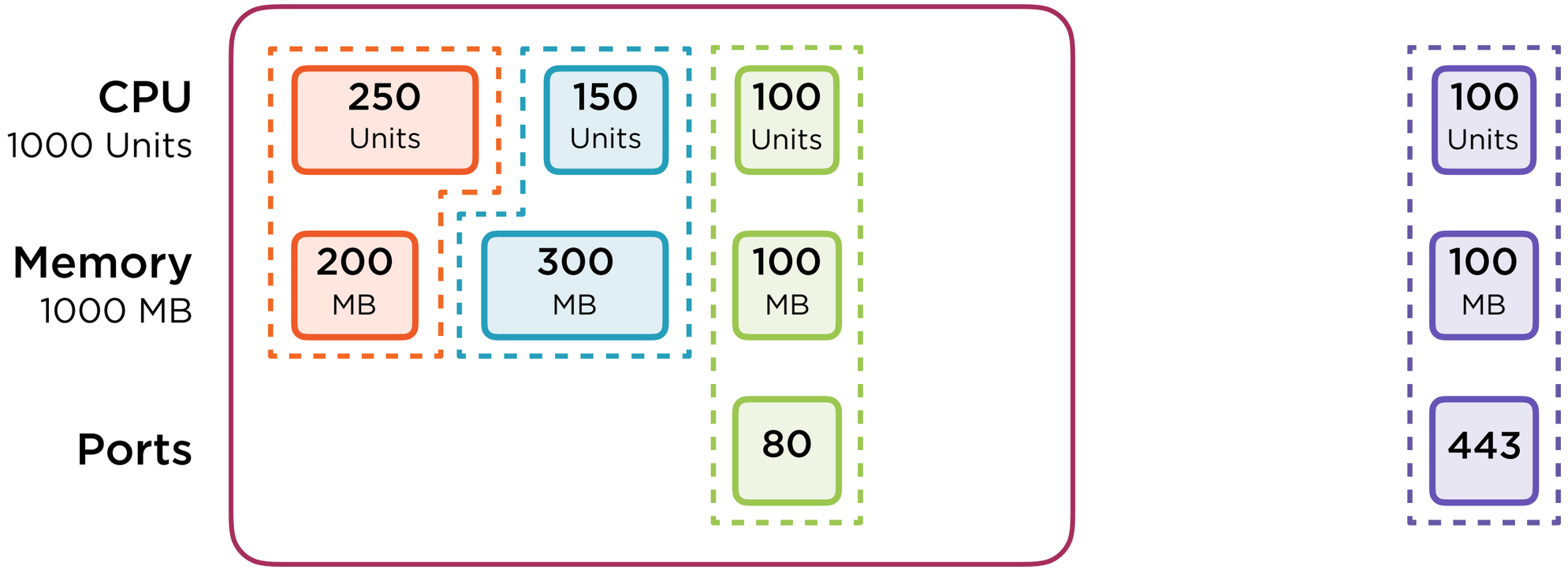
Scaling out ECS Clusters

Resource	Current Usage	Remaining Resources	Container Capacity
CPU	400	600	2 x Orange
Memory	500	500	1 x Blue
Port 80	0	1	1 x Green
Port 443	0	1	1 x Purple
		Minimum Capacity	1



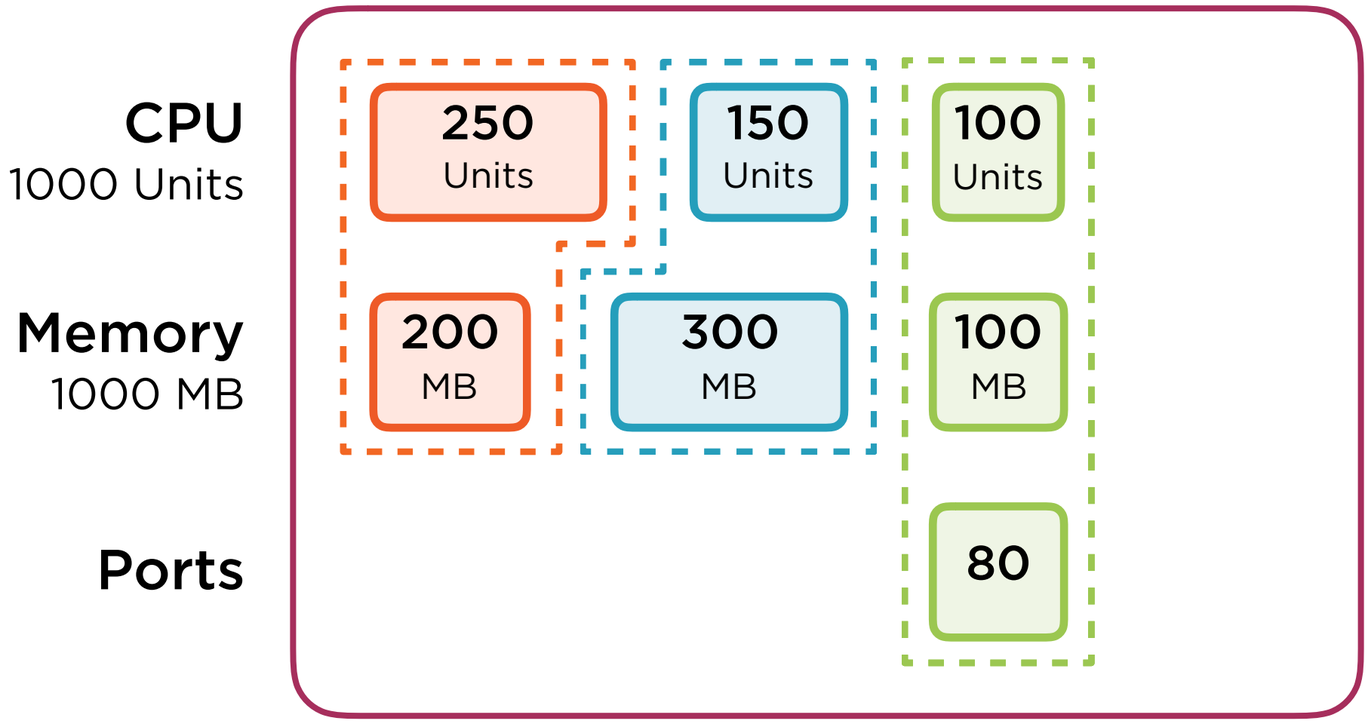
ECS Container Instance A

Resource	Current Usage	Remaining Resources	Container Capacity
CPU	500	500	2 x Orange
Memory	600	400	1 x Blue
Port 80	1	0	0 x Green
Port 443	0	1	1 x Purple
		Minimum Capacity	0



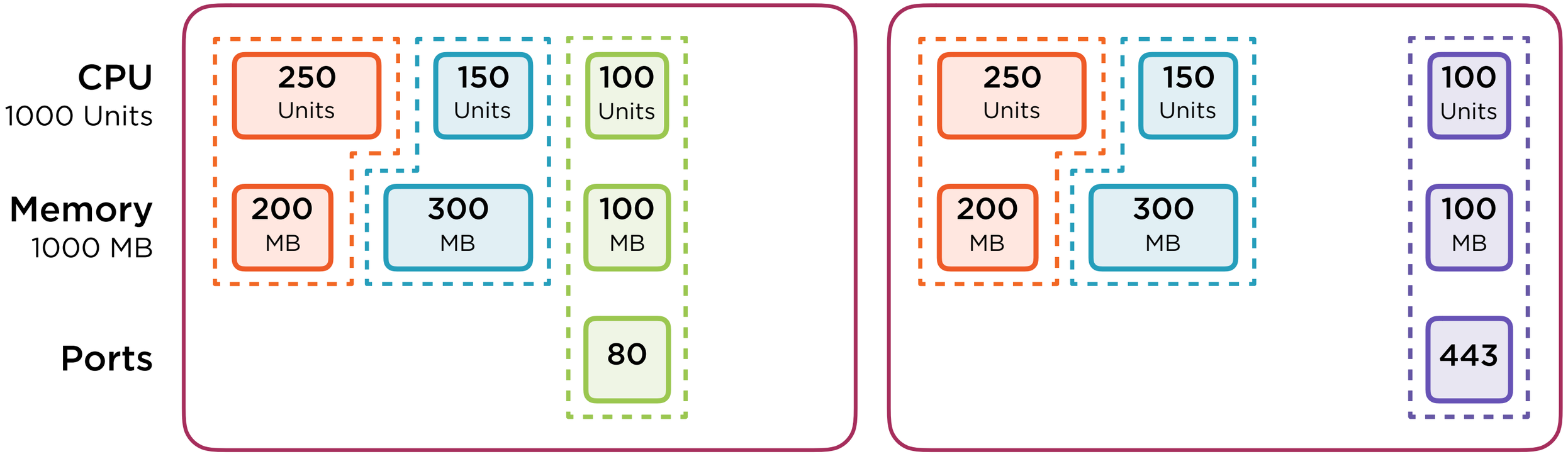
ECS Container Instance A

Resource	Current Usage	Remaining Resources	Container Capacity
CPU	500	500	2 x Orange
Memory	600	400	1 x Blue
Port 80	1	0	**0 x Green**
Port 443	0	1	1 x Purple
		Minimum Capacity	**0**

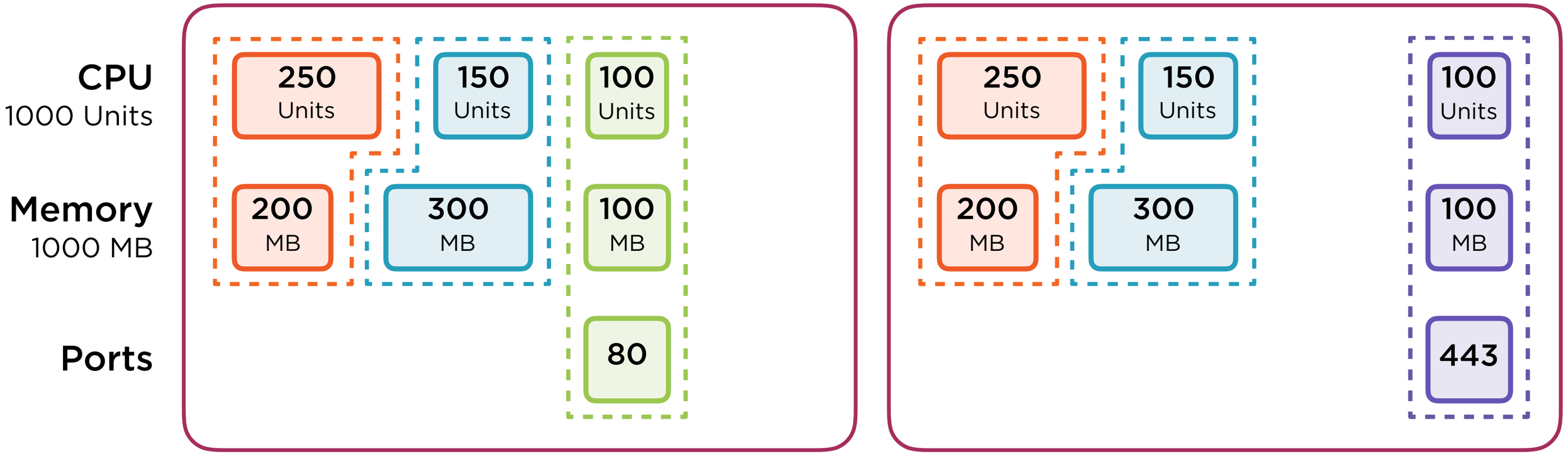


ECS Container Instance A

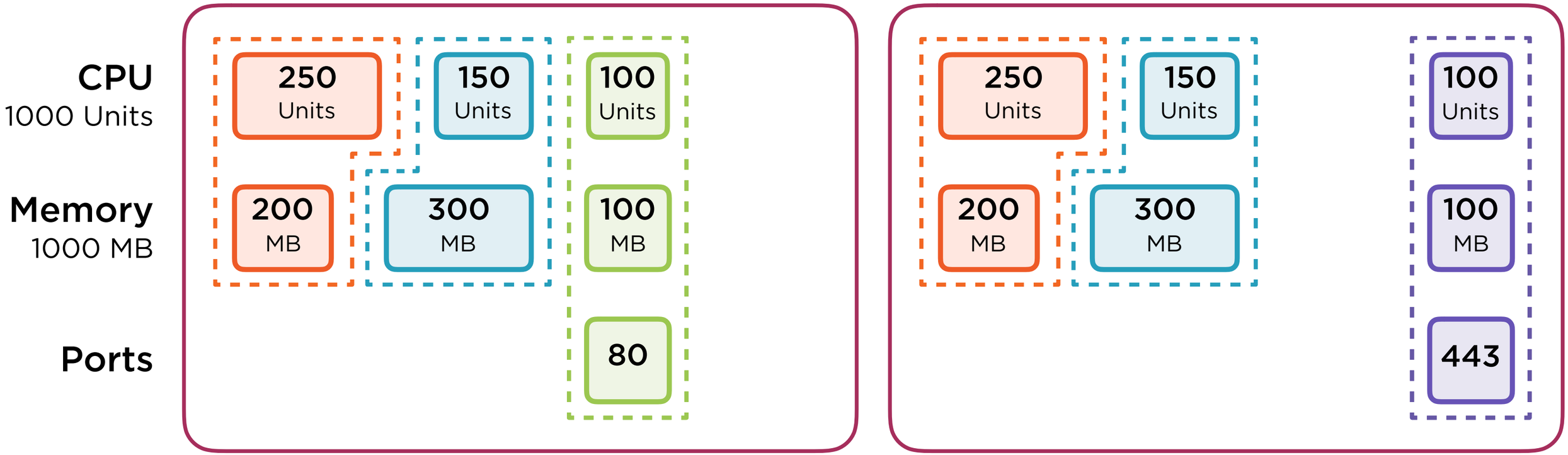
Resource	Current Usage	Remaining Resources	Container Capacity
CPU	500	500	2 x Orange
Memory	600	400	1 x Blue
Port 80	1	0	**0 x Green**
Port 443	0	1	1 x Purple
		Minimum Capacity	**0**



Resource	Current Usage	Remaining Resources	Container Capacity
CPU	500	500	2 x Orange
Memory	600	400	1 x Blue
Port 80	1	0	**0 x Green**
Port 443	0	1	1 x Purple
		Minimum Capacity	**0**

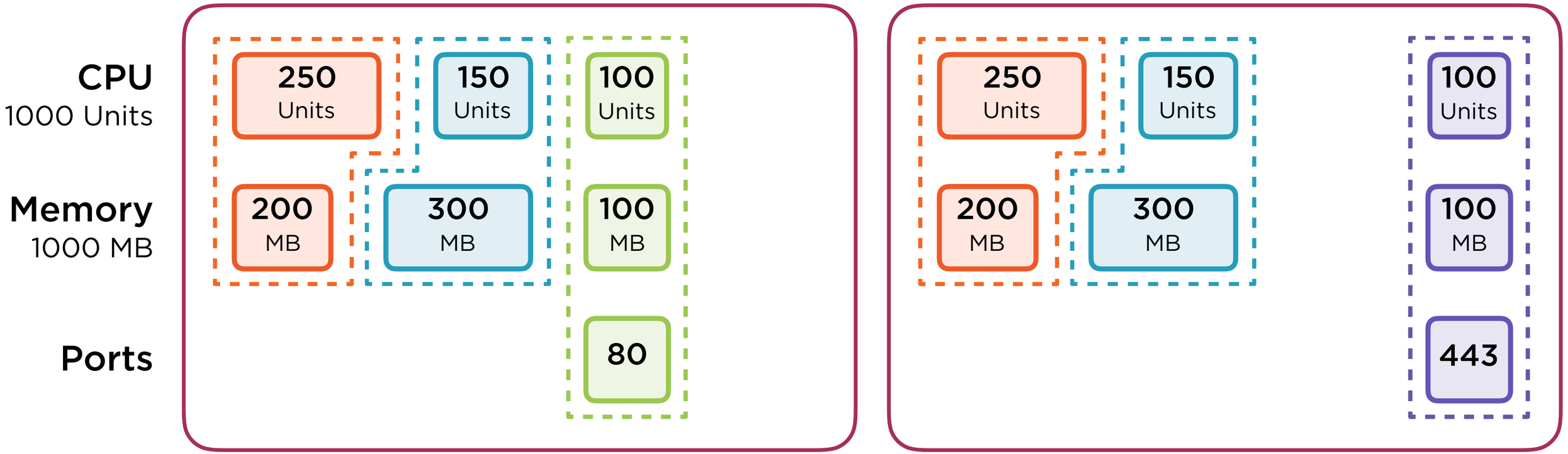


Resource	Current Usage		Remaining Resources		Container Capacity
CPU	500 (A)	**500 (B)**	500 (A)	**500 (B)**	**2 (A) + 2 (B) = 4**
Memory	600 (A)	**600 (B)**	400 (A)	**600 (B)**	**1 (A) + 1 (B) = 2**
Port 80	1 (A)	**0 (B)**	0 (A)	**1 (B)**	**0 (A) + 1 (B) = 1**
Port 443	0 (A)	**1 (B)**	1 (A)	**0 (B)**	**1 (A) + 0 (B) = 1**
			Minimum Capacity		**1**

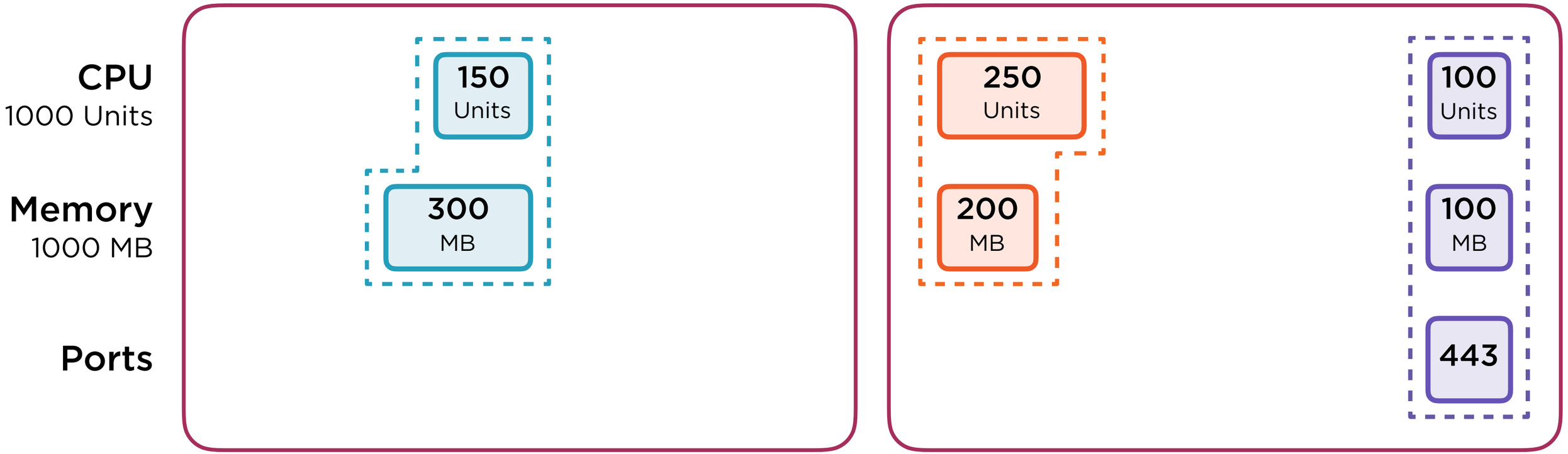


Scaling in ECS Clusters

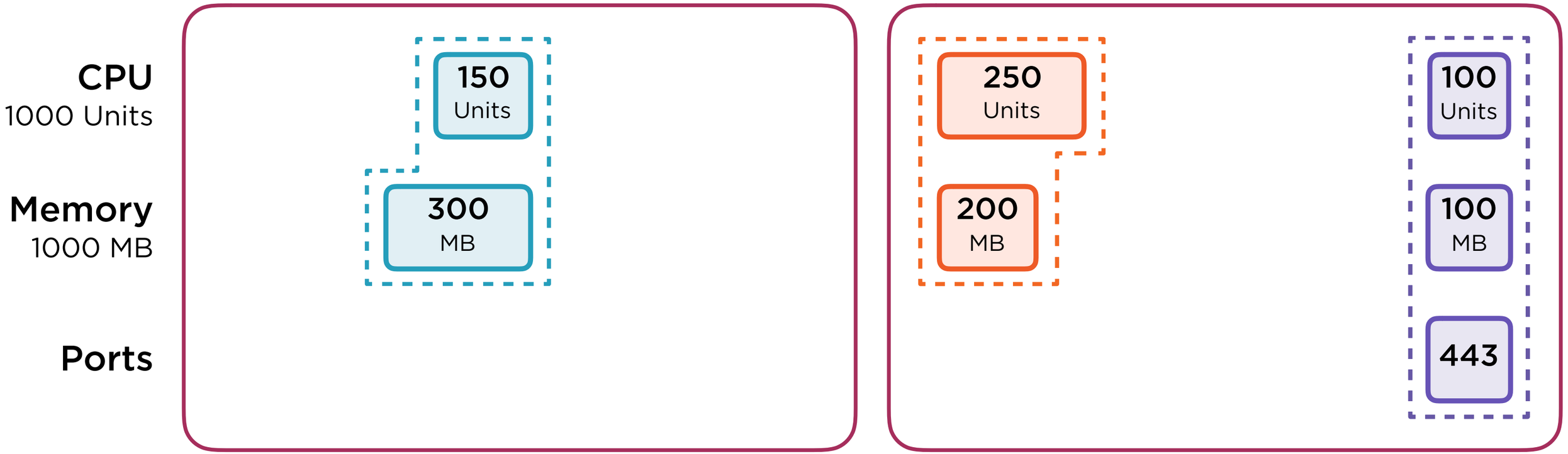
Resource	Current Usage		Remaining Resources		Container Capacity
CPU	500 (A)	500 (B)	500 (A)	500 (B)	2 (A) + 2 (B) = 4
Memory	600 (A)	600 (B)	400 (A)	600 (B)	1 (A) + 1 (B) = 2
Port 80	1 (A)	0 (B)	0 (A)	1 (B)	0 (A) + 1 (B) = 1
Port 443	0 (A)	1 (B)	1 (A)	0 (B)	1 (A) + 0 (B) = 1
			Minimum Capacity		1



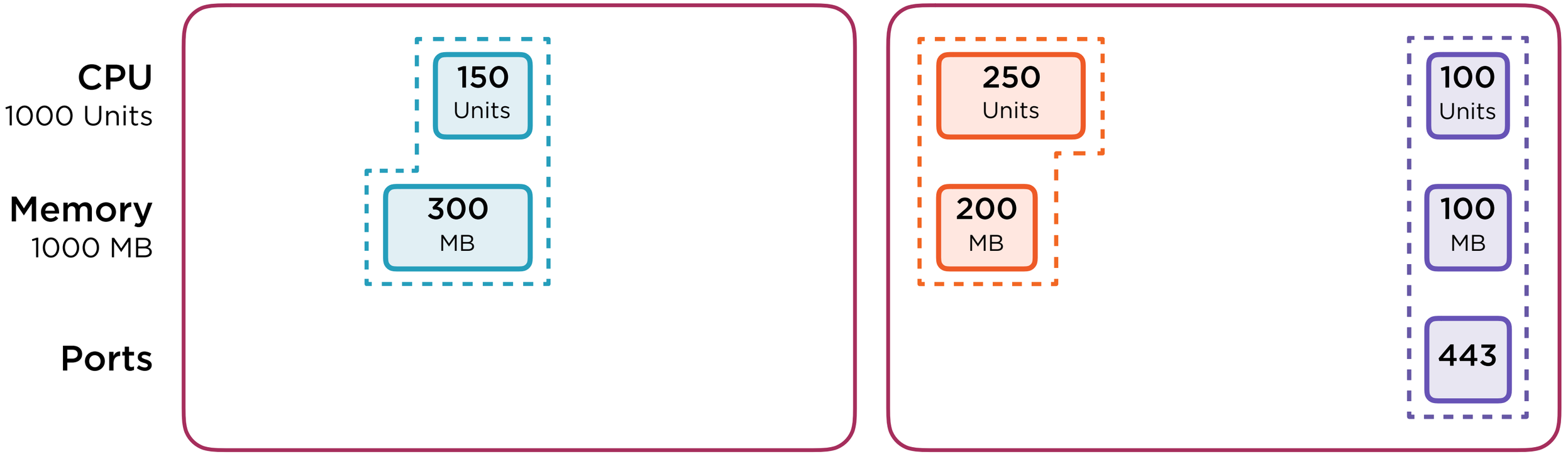
Resource	Current Usage		Remaining Resources		Container Capacity	Instance Capacity	Scale In Capacity
CPU	150 (A)	350 (B)	850 (A)	650 (B)	3 (A) + 2 (B) = 5	3	5 - 3 = 2
Memory	300 (A)	300 (B)	700 (A)	700 (B)	2 (A) + 2 (B) = 4	3	4 - 3 = 1
Port 80	0 (A)	0 (B)	1 (A)	1 (B)	1 (A) + 1 (B) = 2	1	2 - 1 = 1
Port 443	0 (A)	1 (B)	1 (A)	0 (B)	1 (A) + 0 (B) = 1	1	1 - 1 = 0
						Minimum Capacity	0



Resource	Current Usage		Remaining Resources		Container Capacity	Instance Capacity	Idle Capacity
CPU	150 (A)	350 (B)	850 (A)	650 (B)	3 (A) + 2 (B) = **5**	**3**	**5 / 3 = 1.67**
Memory	300 (A)	300 (B)	700 (A)	700 (B)	2 (A) + 2 (B) = **4**	**3**	**4 / 3 = 1.33**
Port 80	0 (A)	0 (B)	1 (A)	1 (B)	1 (A) + 1 (B) = **2**	**1**	**2 / 1 = 2.00**
Port 443	0 (A)	1 (B)	1 (A)	0 (B)	1 (A) + 0 (B) = **1**	**1**	**1 / 1 = 1.00**
						Idle Host Capacity	**1.00**

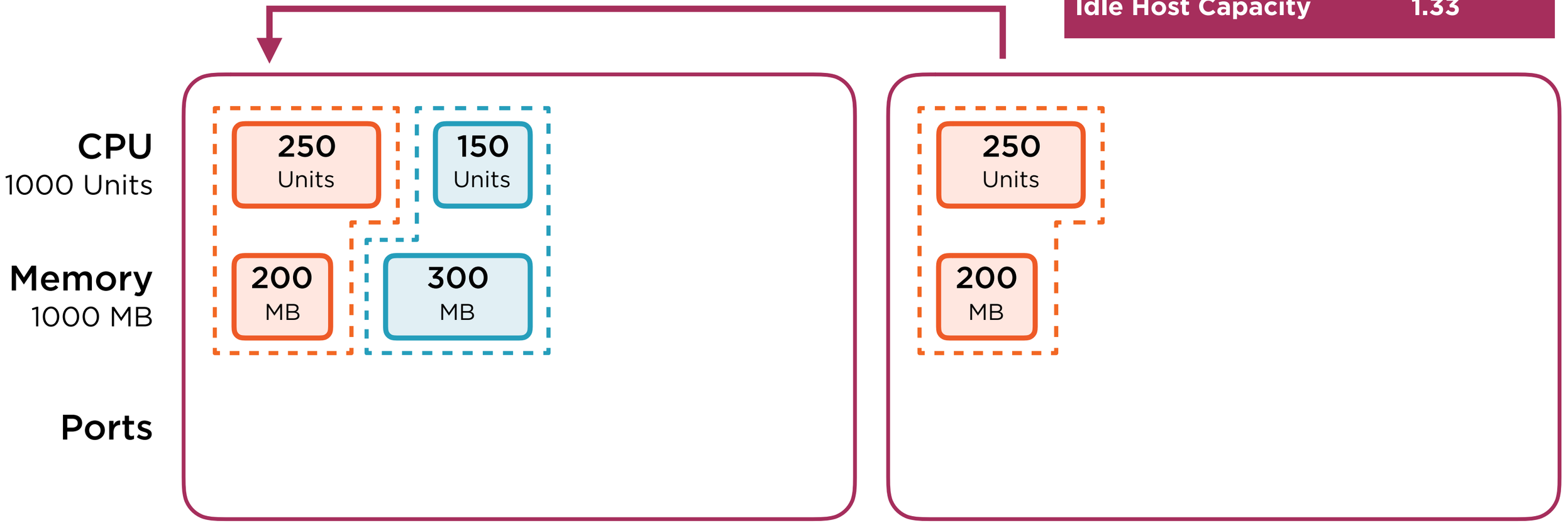


Resource	Current Usage		Remaining Resources		Container Capacity	Instance Capacity	Idle Capacity
CPU	150 (A)	350 (B)	850 (A)	650 (B)	3 (A) + 2 (B) = 5	3	5 / 3 = 1.67
Memory	300 (A)	300 (B)	700 (A)	700 (B)	2 (A) + 2 (B) = 4	3	4 / 3 = 1.33
Port 80	0 (A)	0 (B)	1 (A)	1 (B)	1 (A) + 1 (B) = 2	1	2 / 1 = 2.00
Port 443	0 (A)	1 (B)	1 (A)	0 (B)	1 (A) + 0 (B) = 1	1	1 / 1 = 1.00
						Idle Host Capacity	1.00

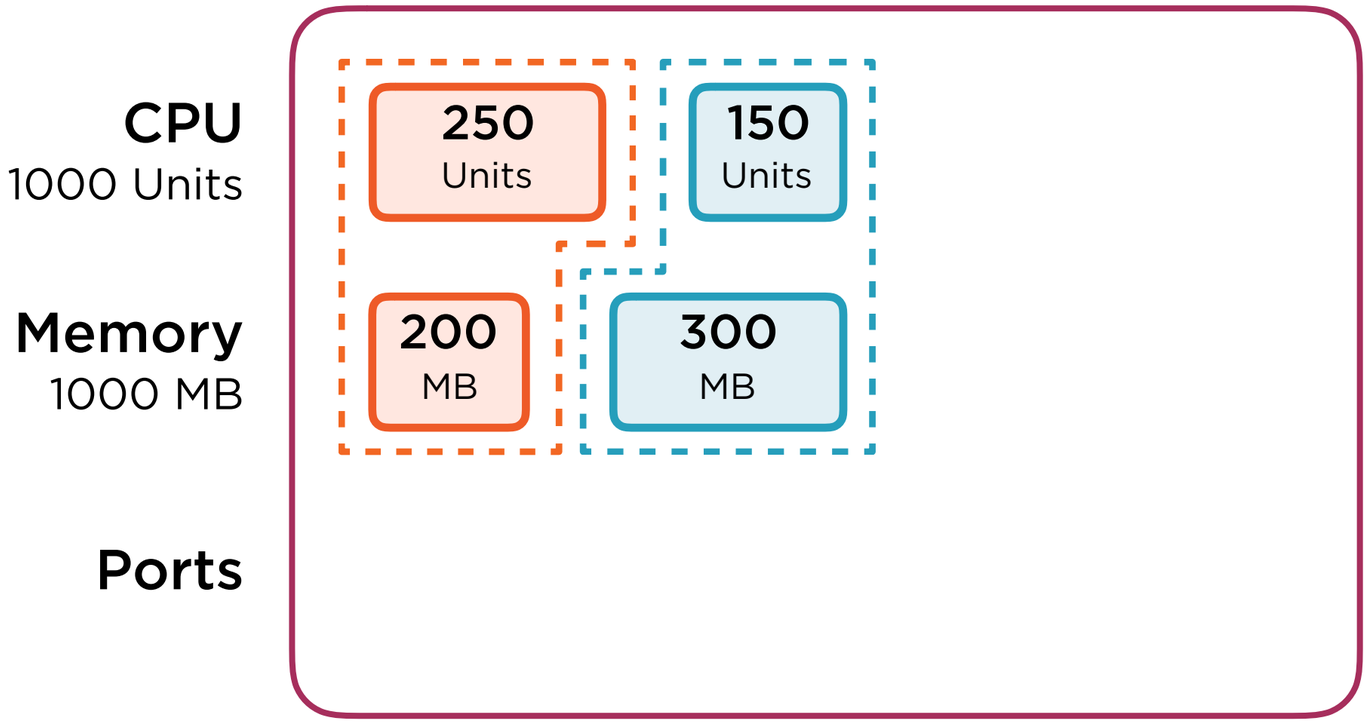


Resource	Current Usage		Remaining Resources		Container Capacity	Instance Capacity	Idle Capacity
CPU	150 (A)	250 (B)	850 (A)	750 (B)	3 (A) + 3 (B) = 6	3	6 / 3 = 2.00
Memory	300 (A)	200 (B)	700 (A)	800 (B)	2 (A) + 2 (B) = 4	3	**4 / 3 = 1.33**
Port 80	0 (A)	0 (B)	1 (A)	1 (B)	1 (A) + 1 (B) = 2	1	2 / 1 = 2.00
Port 443	**0 (A)**	**0 (B)**	**1 (A)**	**1 (B)**	**1 (A) + 1 (B) = 2**	**1**	**2 / 1 = 2.00**

Idle Host Capacity	1.33



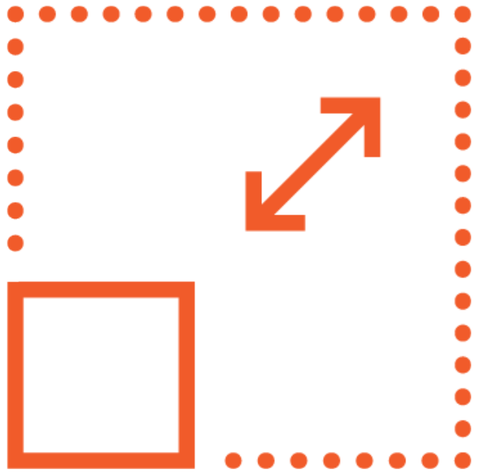
Resource	Current Usage		Remaining Resources		Container Capacity	Instance Capacity	Idle Capacity
CPU	150 (A)	250 (B)	850 (A)	750 (B)	3 (A) + 3 (B) = 6	3	6 / 3 = 2.00
Memory	300 (A)	200 (B)	700 (A)	800 (B)	2 (A) + 2 (B) = 4	3	**4 / 3 = 1.33**
Port 80	0 (A)	0 (B)	1 (A)	1 (B)	1 (A) + 1 (B) = 2	1	2 / 1 = 2.00
Port 443	0 (A)	0 (B)	1 (A)	1 (B)	1 (A) + 1 (B) = 2	1	2 / 1 = 2.00
						Idle Host Capacity	**1.33**



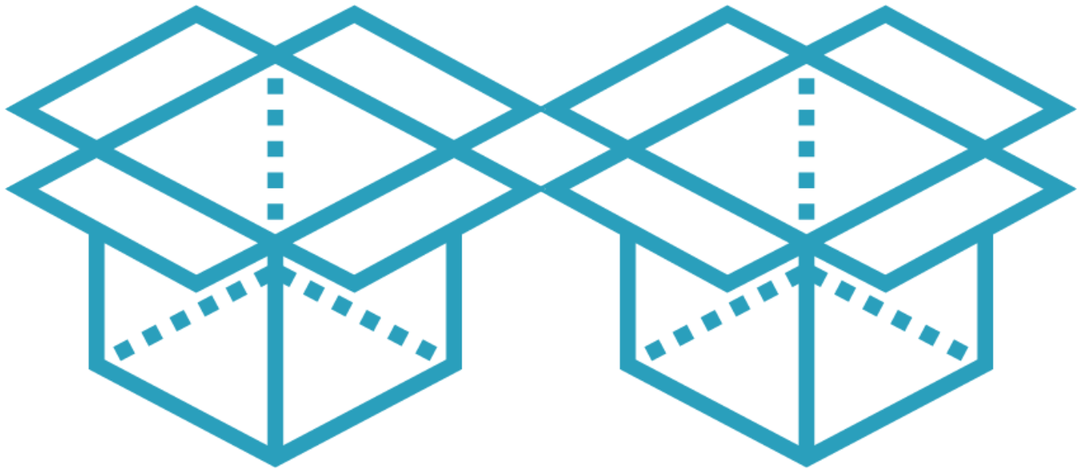
ECS Container Instance A

Auto Scaling Solution Overview

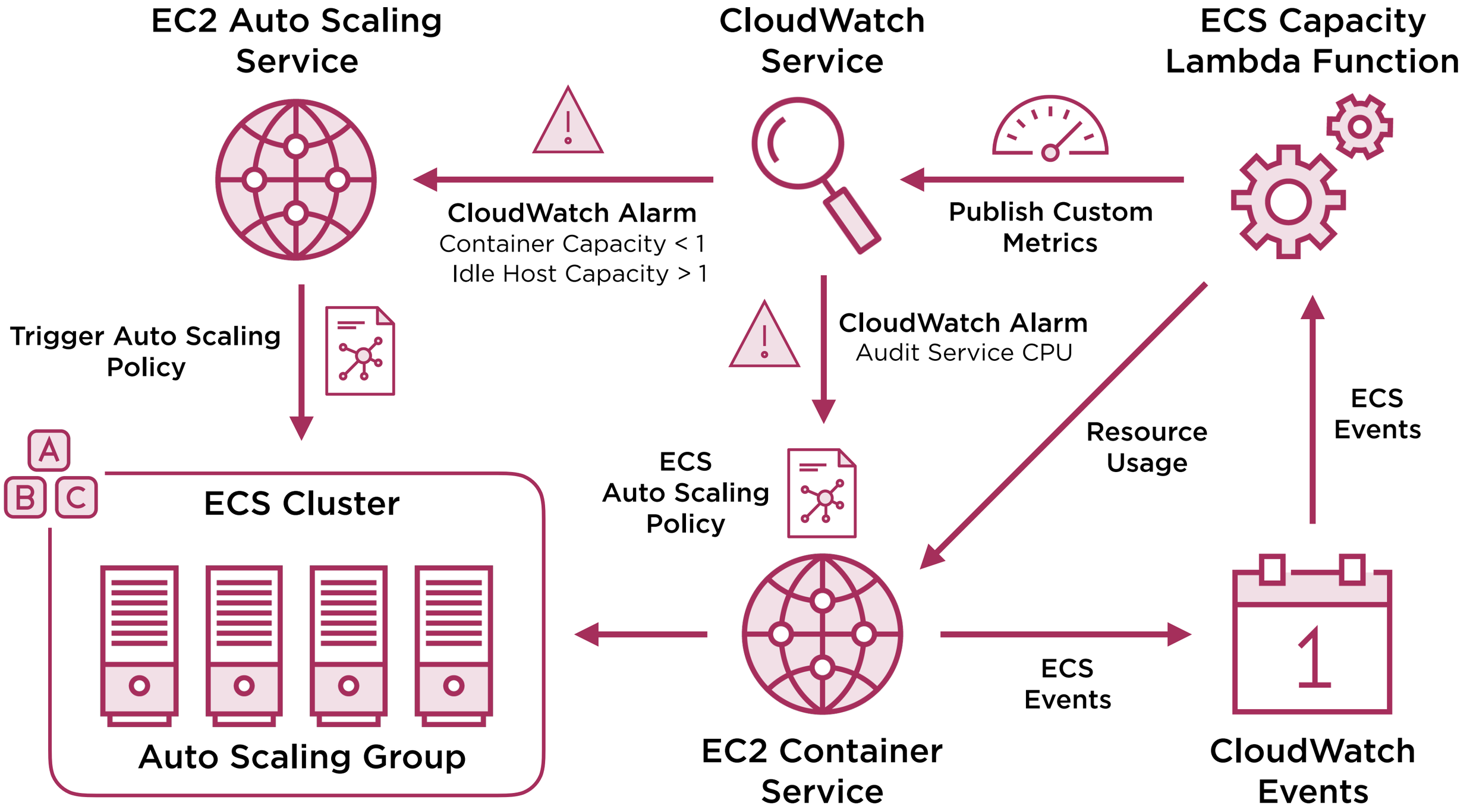
ECS Capacity Summary



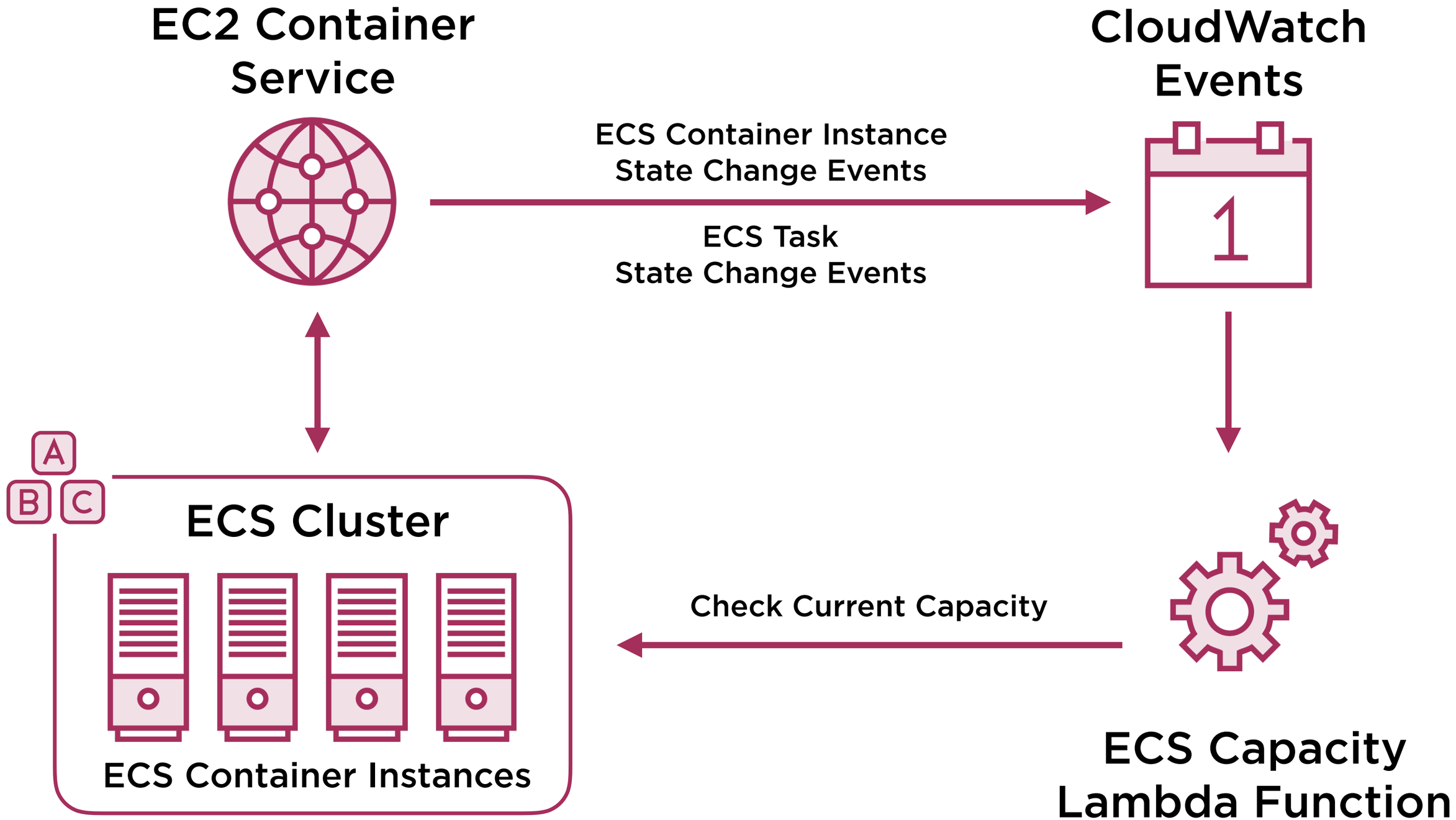
Container Capacity
Scale out when < 1



Idle Host Capacity
Scale in when > 1


EC2 Auto Scaling Service
CloudWatch Service
ECS Capacity Lambda Function
CloudWatch Alarm
Container Capacity < 1
Idle Host Capacity > 1
Publish Custom Metrics
Trigger Auto Scaling Policy
CloudWatch Alarm
Audit Service CPU
ECS Events
Resource Usage
ECS Auto Scaling Policy
A
B
C
ECS Cluster
Auto Scaling Group
EC2 Container Service
ECS Events
1
CloudWatch Events

Understanding ECS Events

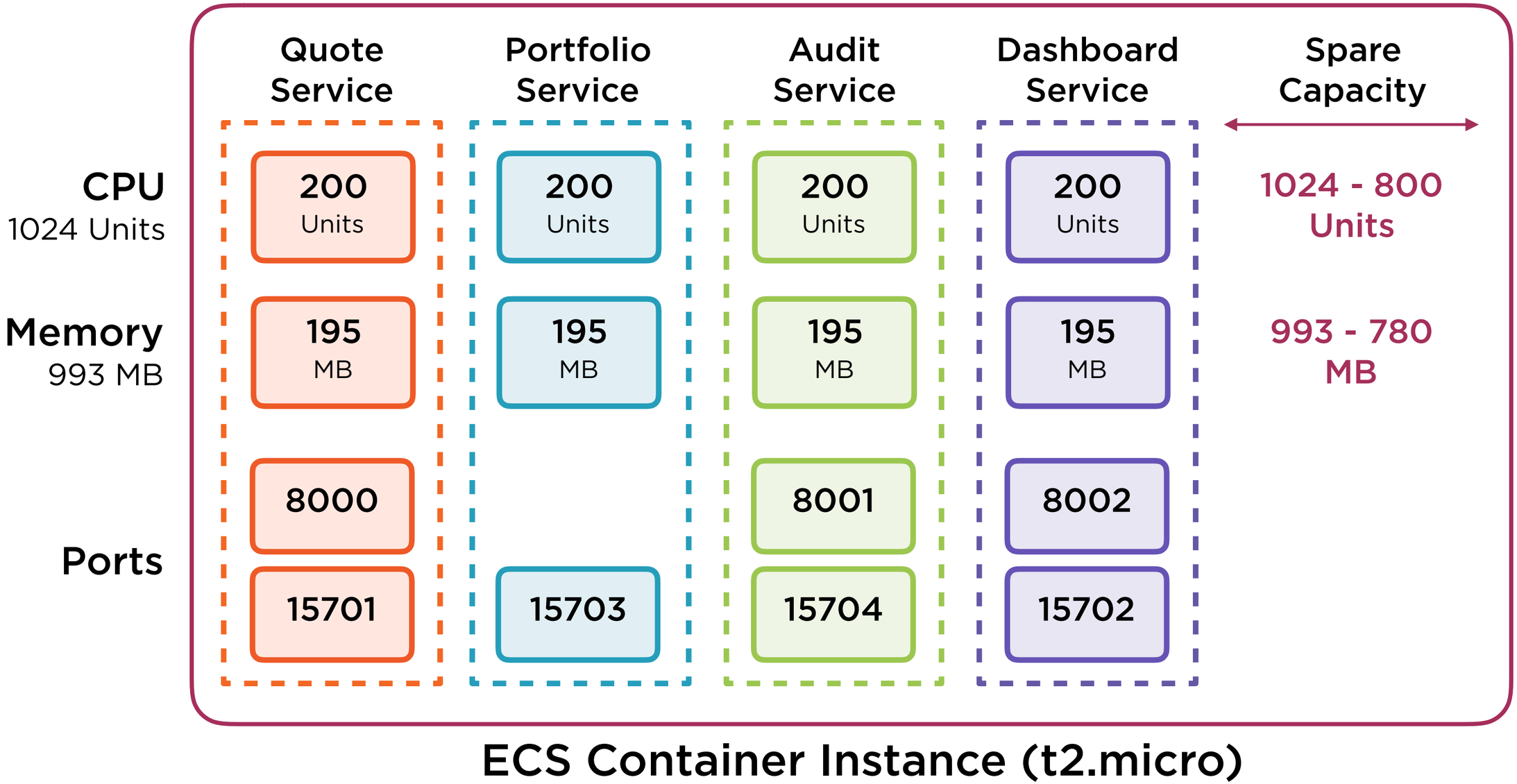

EC2 Container Service
CloudWatch Events
ECS Container Instance State Change Events
ECS Task State Change Events
1
A
B
C
ECS Cluster
ECS Container Instances
Check Current Capacity
ECS Capacity Lambda Function

Creating a Capacity Manager Lambda Function

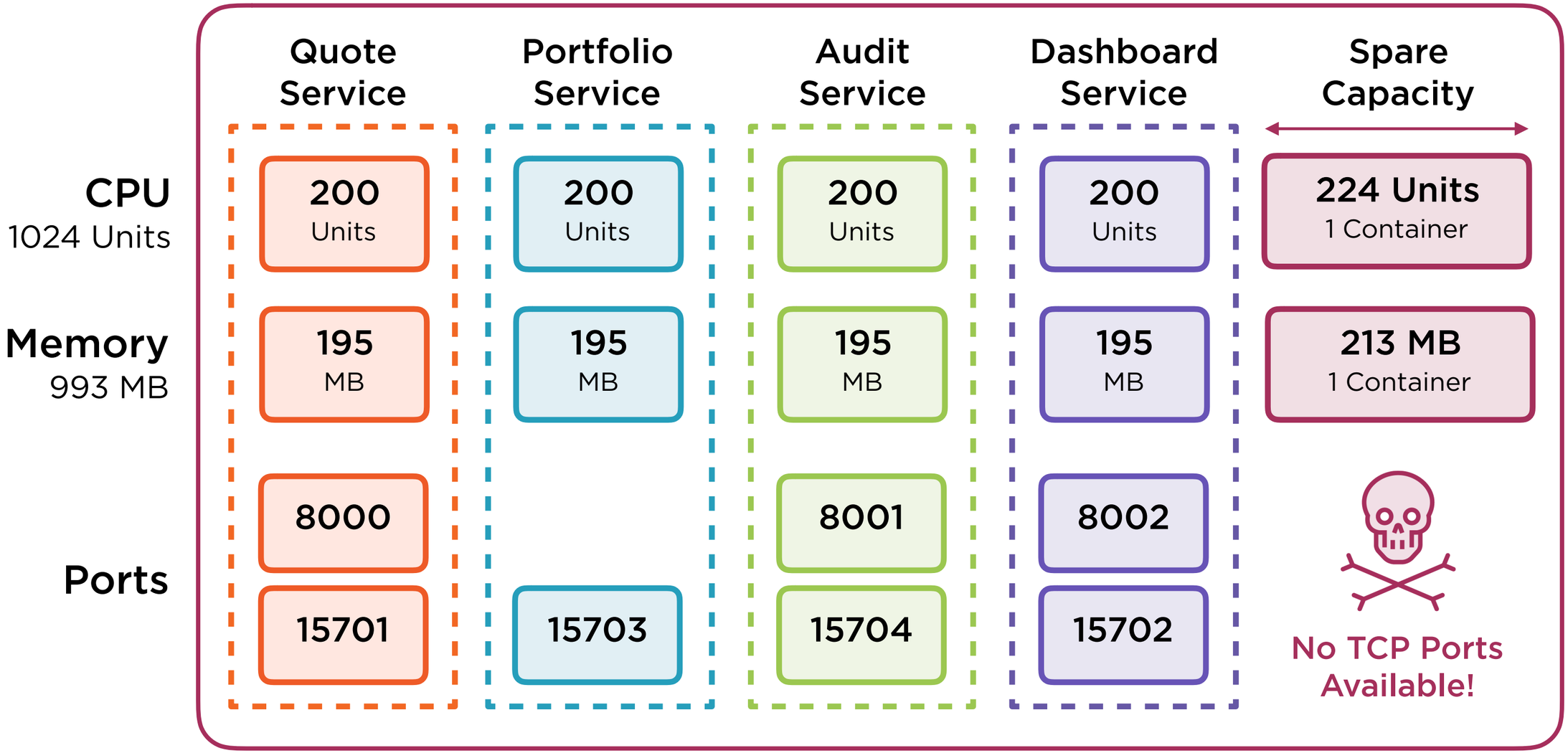
Calculating ECS Container Instance Capacity

Testing the ECS Capacity Manager

Current Microtrader Utilization



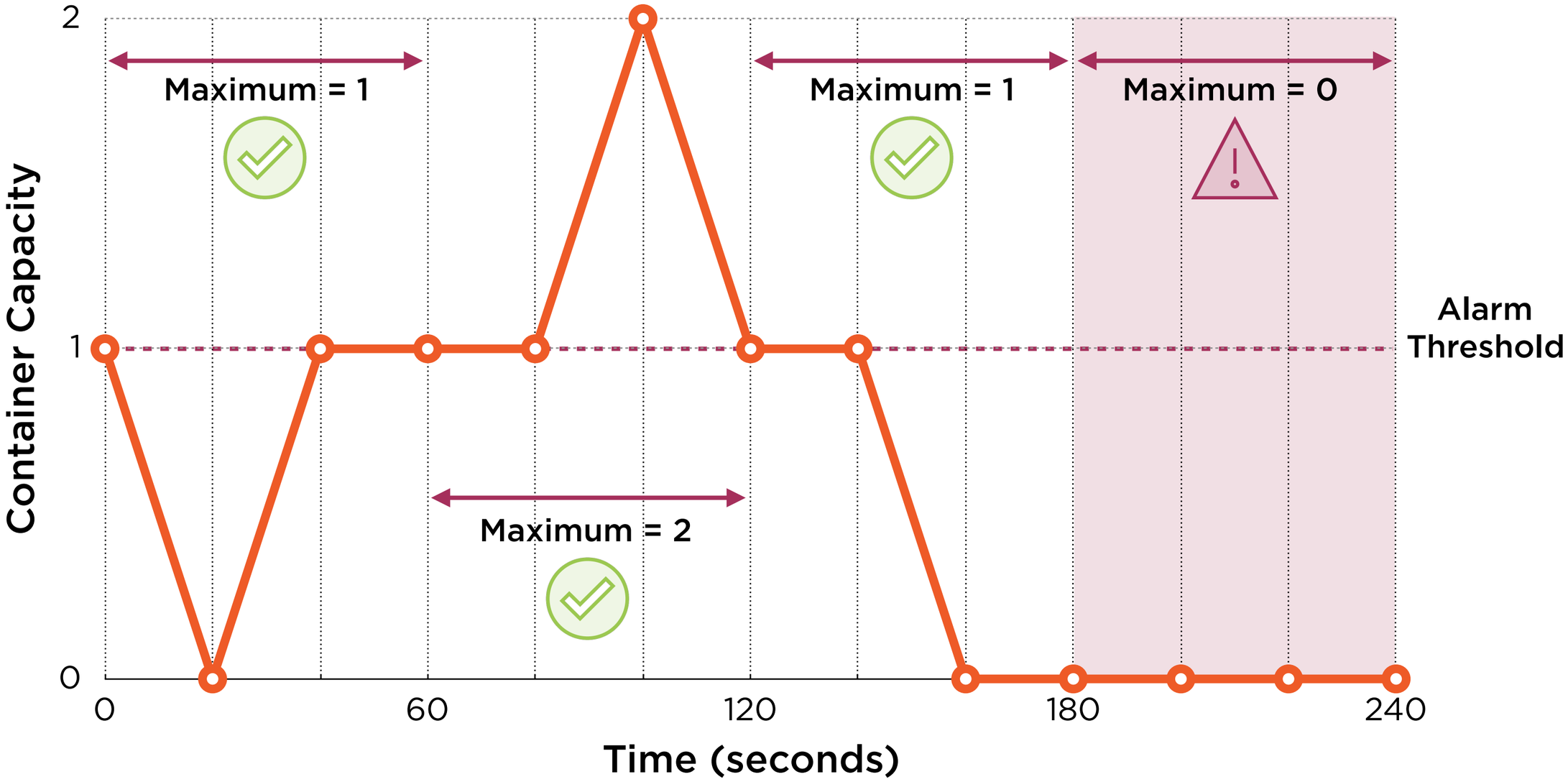
Current Microtrader Utilization



Publishing CloudWatch Custom Metrics

Creating CloudWatch Alarms

Choosing CloudWatch Statistics



Creating EC2 Auto Scaling Policies

Managing ECS Capacity using CloudFormation

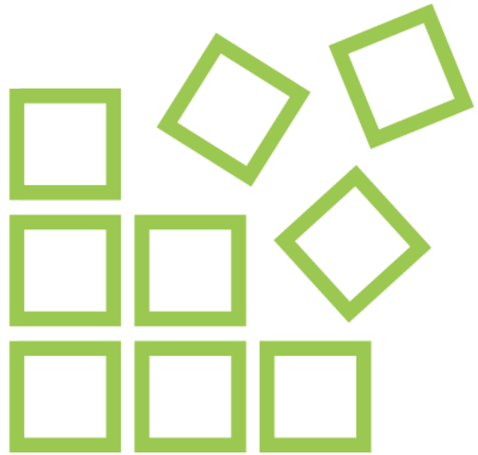
EC2 Auto Scaling using CloudFormation

Configuring ECS Autoscaling

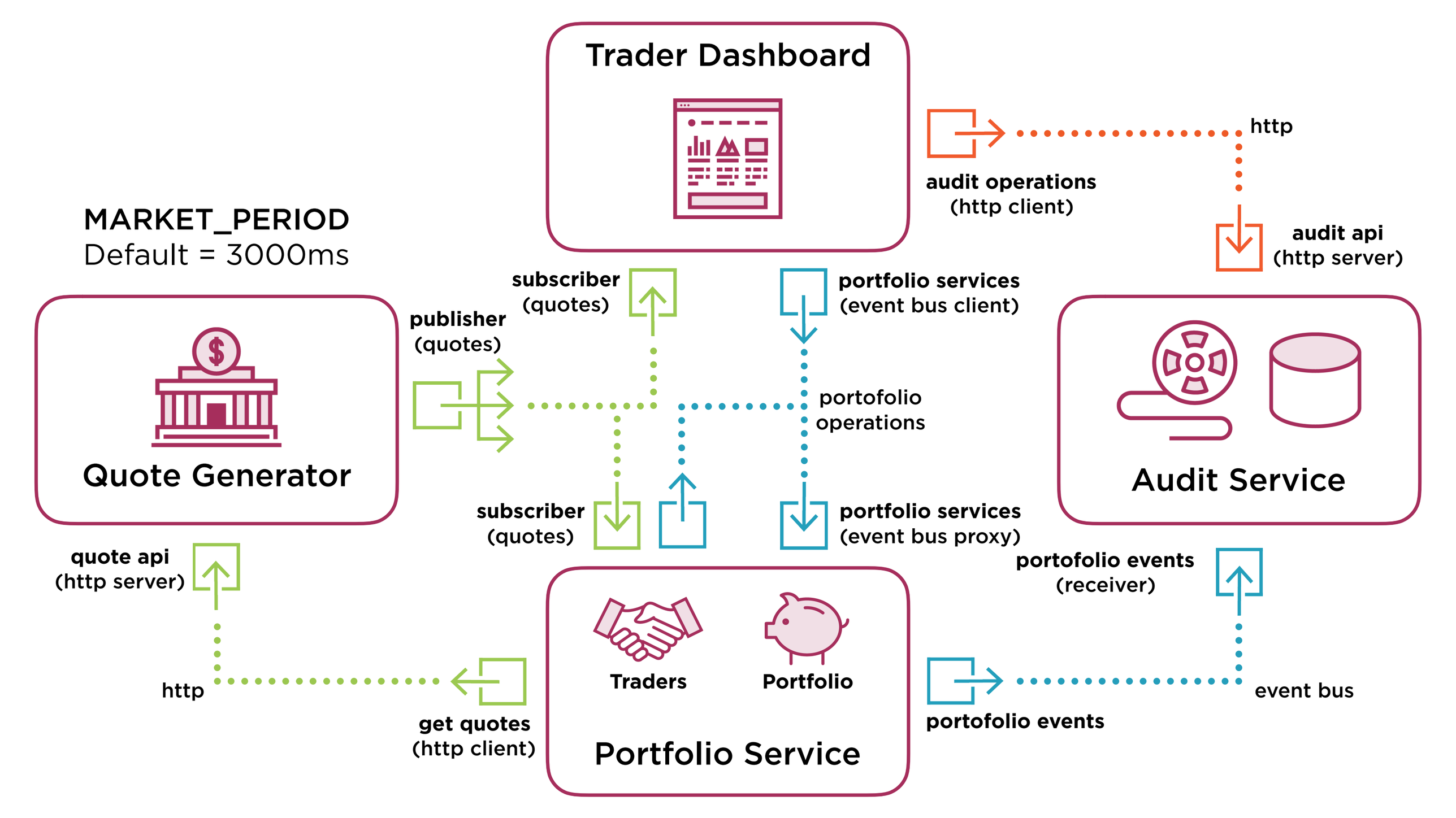
ECS Auto Scaling

Applied at an
ECS Service level

Choose Application
Specific Metrics



Add/Remove
Containers


Trader Dashboard
audit operations
(http client)
http
audit api
(http server)
MARKET_PERIOD
Default = 3000ms
subscriber
(quotes)
portfolio services
(event bus client)
publisher
(quotes)
portofolio
operations
Quote Generator
Audit Service
subscriber
(quotes)
portfolio services
(event bus proxy)
quote api
(http server)
portofolio events
(receiver)
Traders
Portfolio
http
get quotes
(http client)
portofolio events
event bus
Portfolio Service

ECS Auto Scaling using CloudFormation

Summary

Auto Scaling ECS Applications

- ECS Auto Scaling
  - Application specific
  - Adds/subtracts containers for your ECS services
- EC2 Auto Scaling
  - Needs to ensure we have spare capacity for additional containers
  - Needs to scale in when we have too much container capacity

Summary

Calculating ECS Capacity

- Consider the worst case capacity for each type of resource
  - Memory
  - CPU
  - Network Ports
- Take the lowest of each resource capacity calculation
- Container Capacity is used to scale out when < 1
- Idle Host Capacity is used to scale in when > 1

Summary

ECS Auto Scaling Solution

- ECS Capacity Lambda function
  - Triggered by ECS container instance state change events
- Custom CloudWatch metrics
  - ContainerCapacity
  - IdleHostCapacity
- CloudWatch alarms trigger EC2 Auto Scaling policies
- ECS Auto Scaling policies triggered by application-specific metrics and increase or decrease container counts

Avoid static network port mappings!