

Note: This is a hands-on lab/exercise, so looking at various JSON files/commands and editing values in <angle brackets> may be required (i.e. this is not a fully automated soln, but a step by step guide)

Base ECS Setup

(files highlighted here are under folder “/ecs-base”)

1. Create Task Definition

```
aws ecs create-task-definition --cli-input-json file:///app1-td.json
aws ecs create-task-definition --cli-input-json file:///app2-td.json
aws ecs create-task-definition --cli-input-json file:///app3-td.json
```

2. Create Service Discovery Namespace

```
aws servicediscovery create-private-dns-namespace \
  --name test.local \
  --vpc <vpc-id>
```

3. Obtain <namespace-id>

```
aws servicediscovery list-namespaces --filters
Name="NAME",Values="test.local",Condition="EQ"
```

4. Create Service Discovery Service

```
aws servicediscovery create-service \
  --name app1-svc \
  --namespace-id <namespace-id> \
  --dns-config
"NamespaceId=<namespace-id>,RoutingPolicy=MULTIVALUE,DnsRecords=[{Type=
A,TTL=60}]"

aws servicediscovery create-service \
  --name app2-svc \
  --namespace-id <namespace-id> \
  --dns-config
"NamespaceId=<namespace-id>,RoutingPolicy=MULTIVALUE,DnsRecords=[{Type=
A,TTL=60}]"

aws servicediscovery create-service \
  --name app3-svc \
  --namespace-id <namespace-id> \
  --dns-config
"NamespaceId=<namespace-id>,RoutingPolicy=MULTIVALUE,DnsRecords=[{Type=
A,TTL=60}]"
```

5. Obtain Service Registry Arn for app1-svc, app2-svc, app3-svc

```
aws servicediscovery list-services --filters
Name="NAMESPACE_ID",Values="<namespace-id>",Condition="EQ"
```

6. Create ECS Service

```
aws ecs create-service --service-name app1-svc --cli-input-json
file://app1-svc.json
aws ecs create-service --service-name app2-svc --cli-input-json
file://app2-svc.json
aws ecs create-service --service-name app3-svc --cli-input-json
file://app3-svc.json
```

7. Use **ecs exec** to see if app1 can talk to app2 and app3, etc.

```
aws ecs execute-command --cluster default \
--task <task-id> \
--region <region> \
--container app1 \
--interactive \
--command "/bin/sh"
```

AppMesh Implementation

(files highlighted here are under folder “/appmesh-implementation”)

1. Create mesh

```
aws appmesh create-mesh --cli-input-json file://mesh.json
```

2. Create virtual nodes (VN)

```
aws appmesh create-virtual-node --cli-input-json file://app1-vn.json
aws appmesh create-virtual-node --cli-input-json file://app2-vn.json
aws appmesh create-virtual-node --cli-input-json file://app3-vn.json
```

3. Create virtual services (VS)

```
aws appmesh create-virtual-service --cli-input-json
file://app1-vs.json
aws appmesh create-virtual-service --cli-input-json
file://app2-vs.json
aws appmesh create-virtual-service --cli-input-json
file://app3-vs.json
```

4. Update ECS Task Definition with AppMesh Integration

- a. This adds envoy proxy container
- b. And proxy configuration on the task definition

```
aws ecs register-task-definition --cli-input-json
file://app1-td-envoy.json
aws ecs register-task-definition --cli-input-json
file://app2-td-envoy.json
aws ecs register-task-definition --cli-input-json
file://app3-td-envoy.json
```

5. Update ECS Service with new task definition

```
aws ecs update-service --cli-input-json file://app3-svc-update.json
aws ecs update-service --cli-input-json file://app3-svc-update.json
aws ecs update-service --cli-input-json file://app3-svc-update.json
```

6. Use **ecs exec** to see if app1 can talk to app2 and app3, etc.

```
aws ecs execute-command --cluster default \
--task <task-id> \
--region <region> \
--container app1 \
--interactive \
--command "/bin/sh"
```

Q: Why can't app1 talk to app2 or app3?

A: Virtual node service backends

7. Update Virtual Node configuration

```
aws appmesh update-virtual-node --generate-cli-skeleton >
app1-vn-update.json
```

Exposing App Mesh Virtual Services to outside world (outside of mesh)

(files highlighted here are under folder "/vgw-implementation")

1. Create NLB (Network Load Balancer) - *Recommended over ALB*

```
aws elbv2 create-load-balancer --name my-load-balancer --type network
--subnets <subnet-1> <subnet-2> <subnet-3>
```

2. Create Target Group

```
aws elbv2 create-target-group --name envoy-targets --protocol TCP
--port 80 --vpc-id <vpc-id>
```

3. Create LoadBalancer Listener

```
aws elbv2 create-listener --load-balancer-arn <load-balancer-arn> \
--protocol TCP --port 80 \
--default-actions Type=forward,TargetGroupArn=<target-group-arn>
```

4. Create Virtual Gateway

```
aws appmesh create-virtual-gateway --cli-input-json file://vgw.json
```

5. Create Virtual Gateway Route (Gateway Route)

```
aws appmesh create-gateway-route --cli-input-json file://vgwr.json
```

6. Create service discovery for VGW (optional)

```
aws servicediscovery create-service \
--name vgw-envoy-service \
--namespace-id <namespace-id> \
--dns-config
"NamespaceId=<namespace-id>,RoutingPolicy=MULTIVALUE,DnsRecords=[{Type=A,TTL=60}]"
```

7. Create VGW Envoy Task Definition

```
aws ecs register-task-definition --cli-input-json
file://vgw-envoy-td.json
```

8. Create VGW Envoy Service

```
aws ecs register-task-definition --cli-input-json
file://vgw-envoy-svc.json
```

ECS Console Screen Grabs (VN TD Creation)

Service integration

AWS App Mesh is a service mesh based on the Envoy proxy that makes it easy to monitor and control microservices. App Mesh standardizes how your microservices communicate, giving you end-to-end visibility and helping to ensure high-availability for your applications. To enable App Mesh integration, complete the following fields and then choose **Apply** which will auto-configure the proxy configuration. [Learn more](#)

Enable App Mesh integration ☒

Mesh name

AppMesh endpoints ☒ Virtual node ☐ Virtual gateway

Application container name

Virtual node name

Virtual node port

Envoy image

ECS Console Screen Grabs (VGW TD Creation)

Service integration

AWS App Mesh is a service mesh based on the Envoy proxy that makes it easy to monitor and control microservices. App Mesh standardizes how your microservices communicate, giving you end-to-end visibility and helping to ensure high-availability for your applications. To enable App Mesh integration, complete the following fields and then choose **Apply** which will auto-configure the proxy configuration. [Learn more](#)

Enable App Mesh integration ☒

Mesh name

AppMesh endpoints ☐ Virtual node
☒ Virtual gateway

Virtual gateway name 

Virtual gateway port

Envoy image

Apply

Apply App Mesh Configuration



Clicking "Confirm" will make the following changes to your task and container definitions:

- Add new "envoy" container to existing container definitions
- Remove application container `{"label": "Application container name", "value": ""}`
- Disable proxy configuration

Cancel

Confirm