Deep Dive into Container Networking

Anirudh Aithal
Senior Software Development Engineer
Amazon Web Services – Container Services

AWS re:Invent

aws

Agenda

Container networking use cases

Task networking in Amazon Elastic Container Service (Amazon ECS)

Task networking in Fargate

Pod networking in Amazon Elastic Container Service for Kubernetes (Amazon EKS)

1. Discovery

Use-case - Discovery

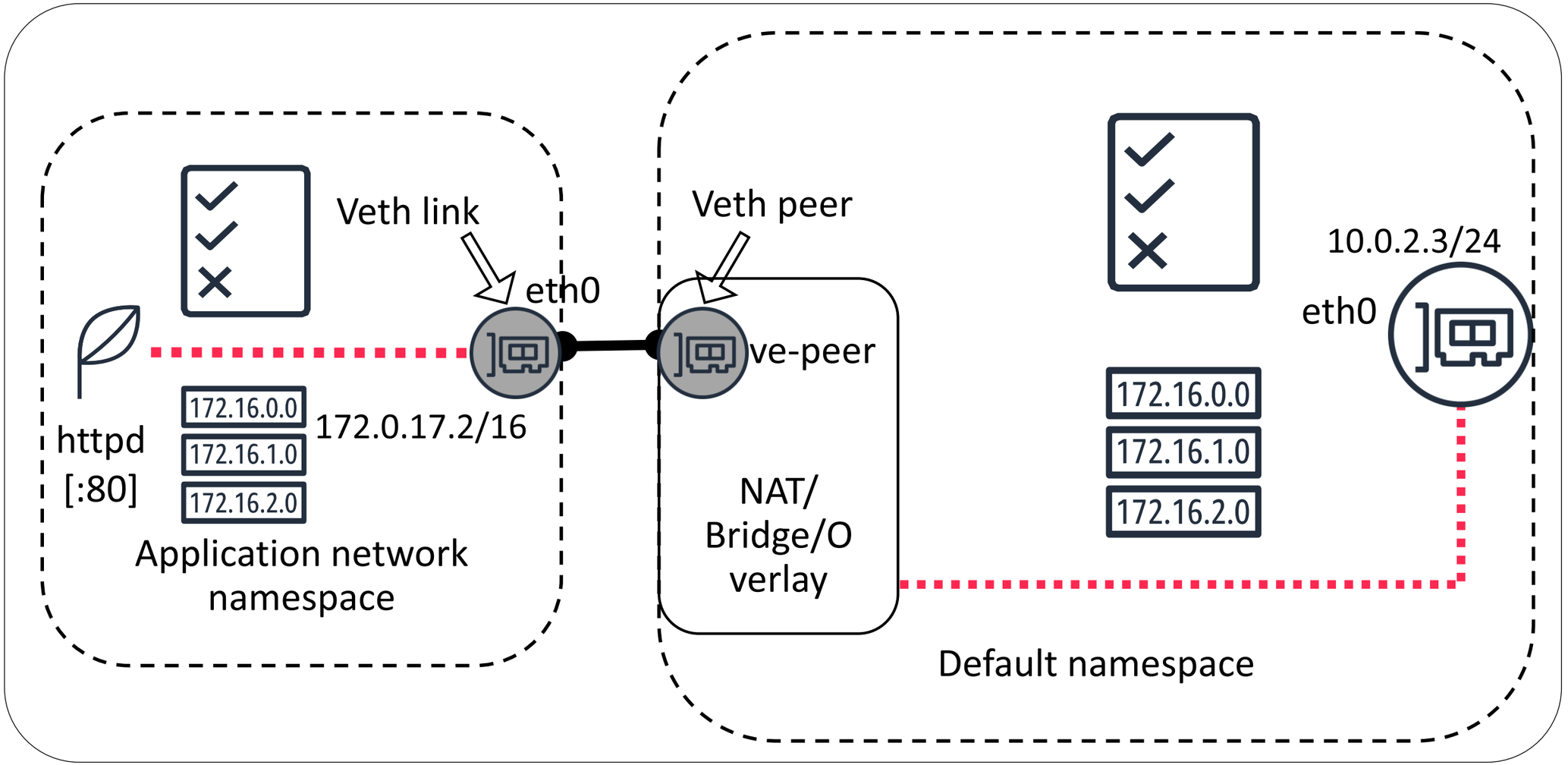
Within an instance: Alice deploys an application that consists of multiple containers, which are deployed together (colocated on an instance). Container-A wants to communicate with Container-B using Container-B's name.

Across instances: Bob has two microservices Service-A & Service-B. Service-A wants to call remote APIs supported by Service-B. Service-B should be reachable by containers under Service-A.

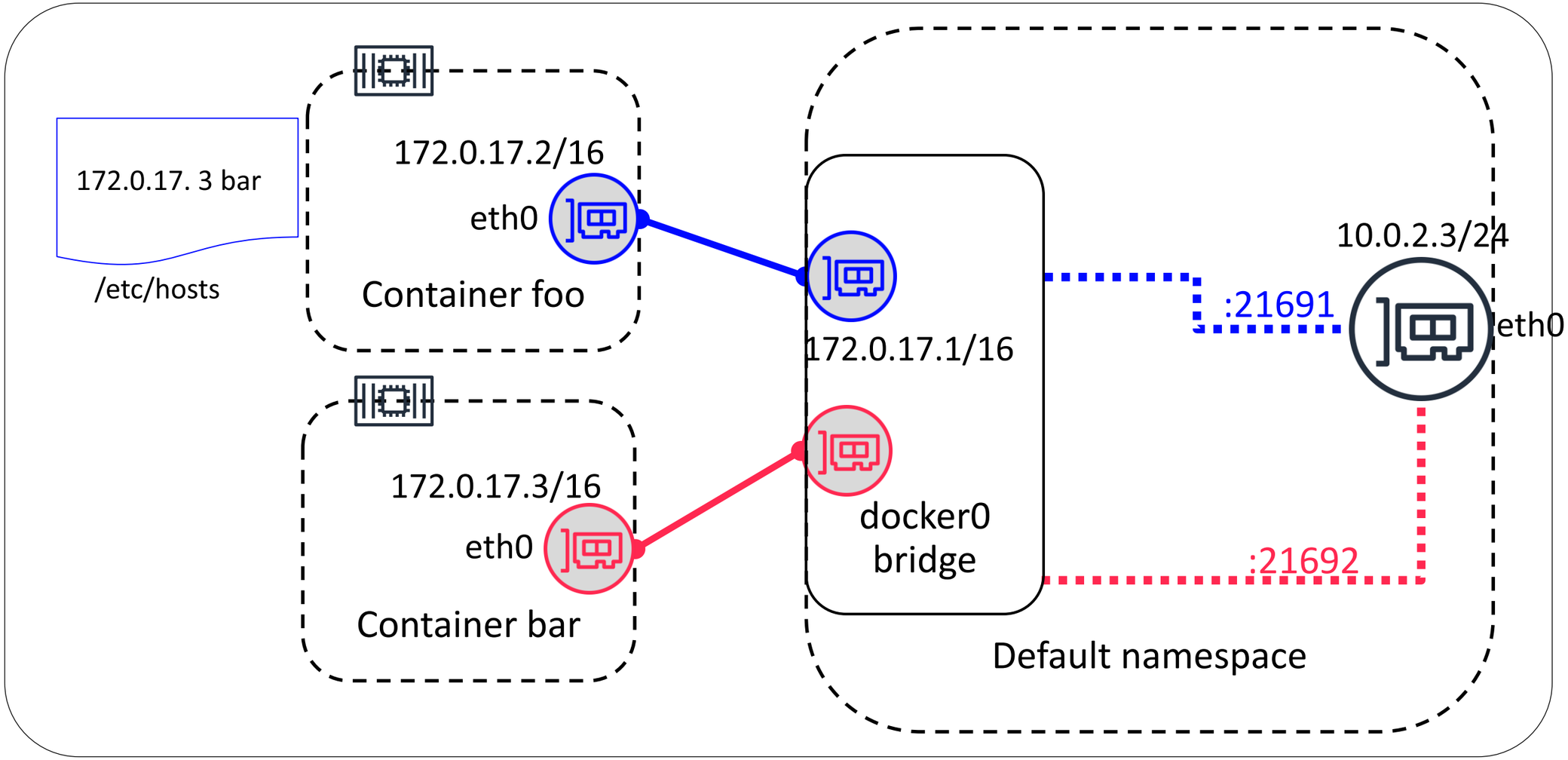
Consistent addressing: Cara creates service-1 using container-A, which exposes port 80. Cara should be able to run multiple instances of service-1 on same container instance without worrying about port conflicts.

VPC integration: Deepak runs his own DNS server in his VPC. He wants his applications to get domain names and IPv4 addresses from this domain.

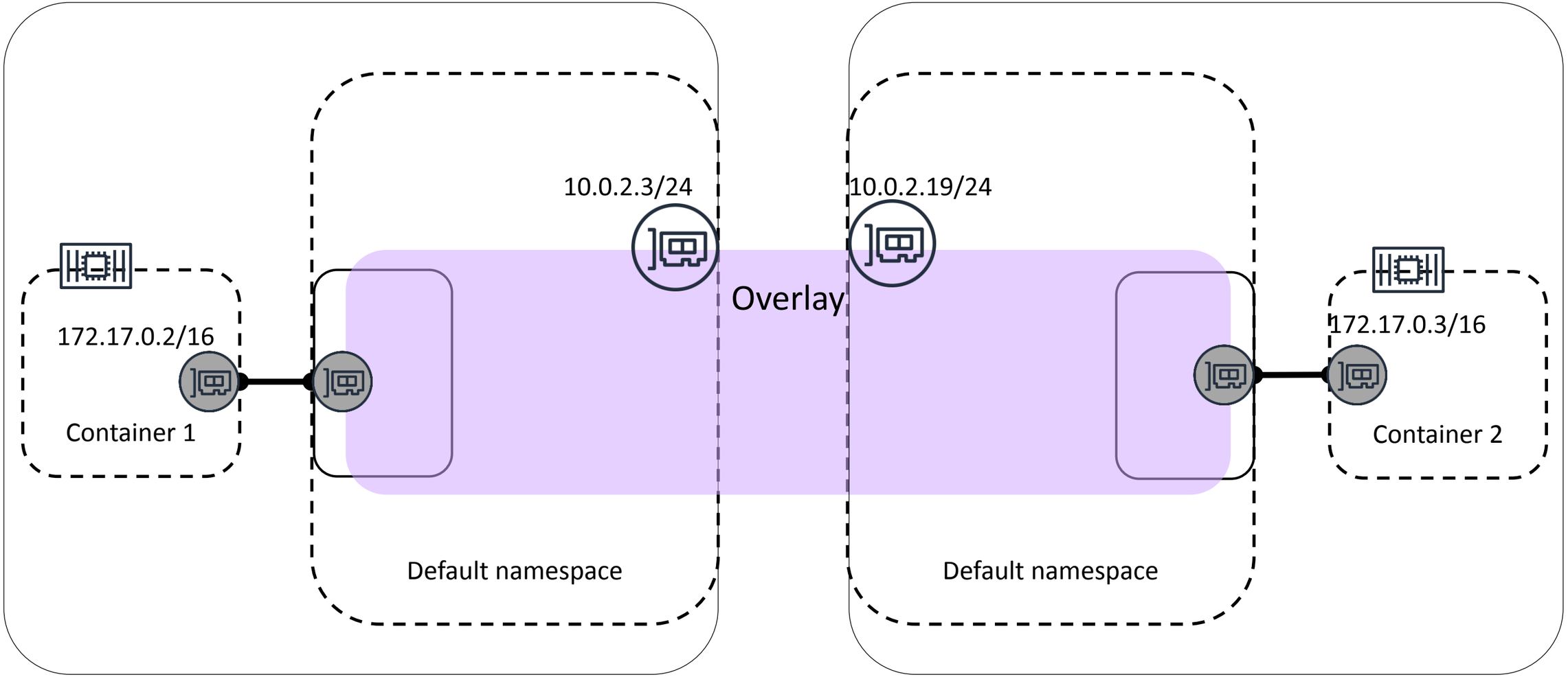
Discovery – Docker bridge



Discovery – Docker bridge + links



Discovery – Overlay



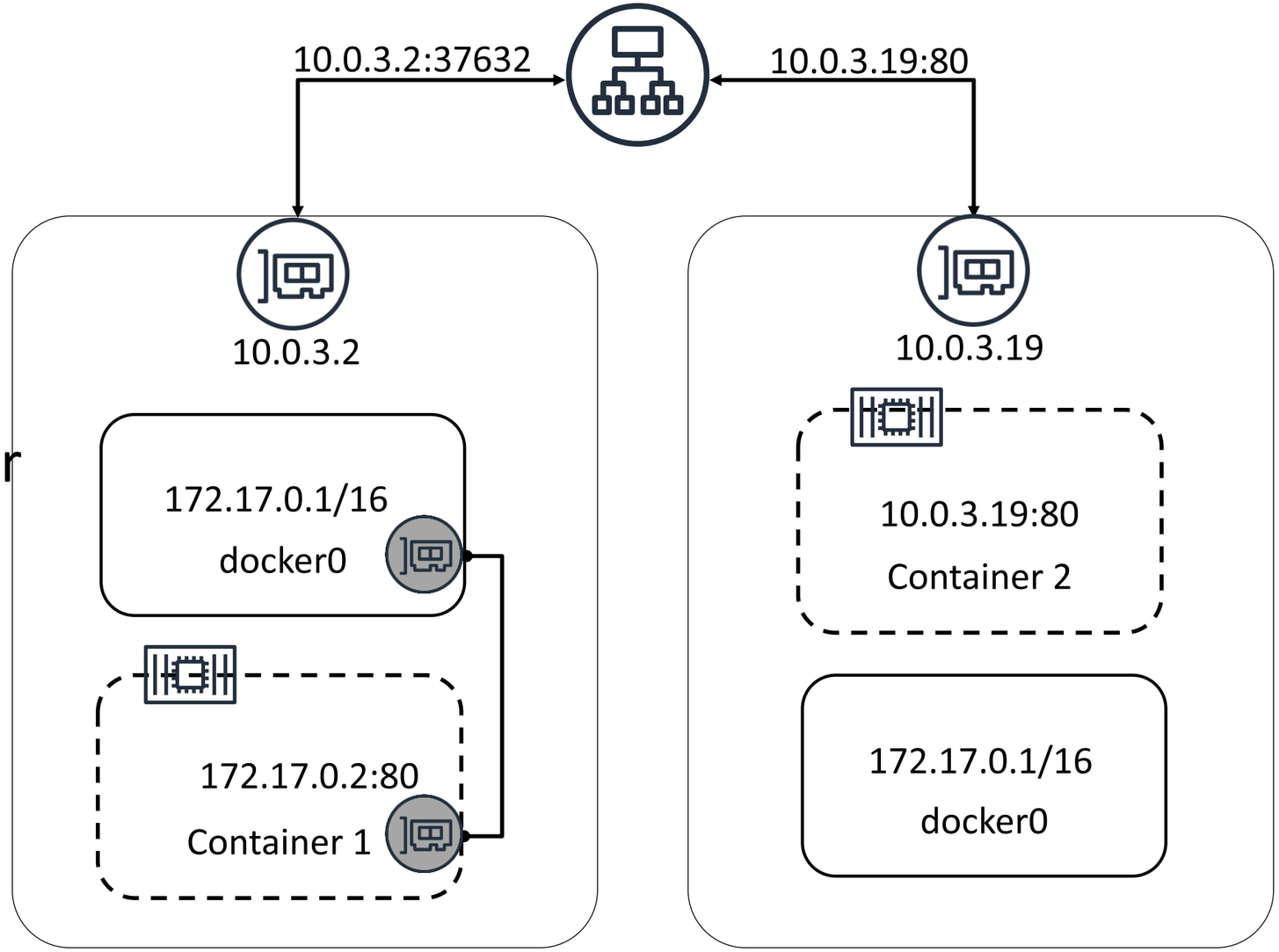
Discovery—Load balancers (LBs) or Service Discovery

LBs distribute application traffic across multiple targets

Orchestrators (Amazon ECS, k8s, and others) automatically register applications as

targets to LBs and Route53

Application health checks can be used as target health checks



Application LB (ALB) or Network LB (NLB)?

TLS needs to terminate on the host? – NLB

Need HTTP Host and Path Based Routing? – ALB

Application uses non HTTP protocol? – NLB

Demo

1. Discovery

2. Security

AWS re:Invent

aws

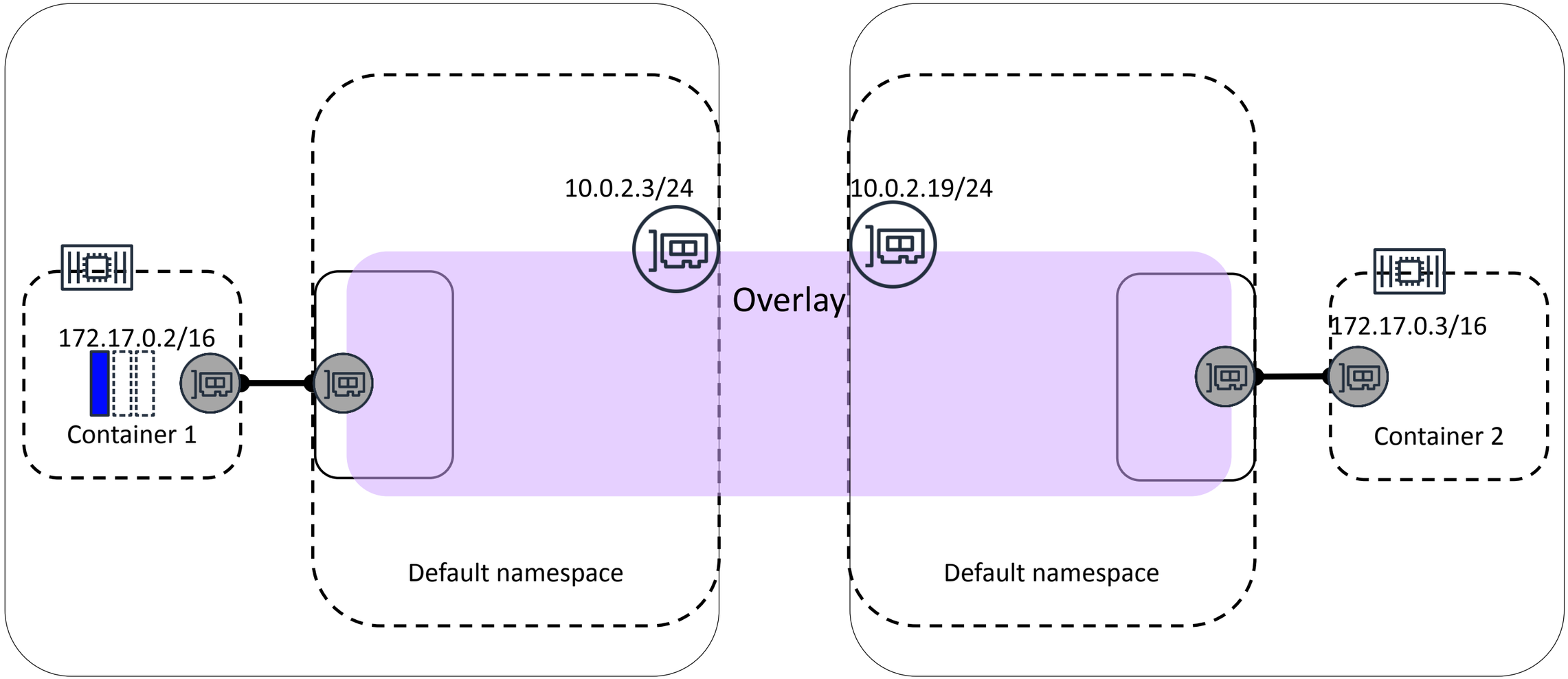
Use-case - Security

Access control: Elizabeth wants to ensure that none of the applications hosted on the infrastructure she maintains expose port 22 to the internet.

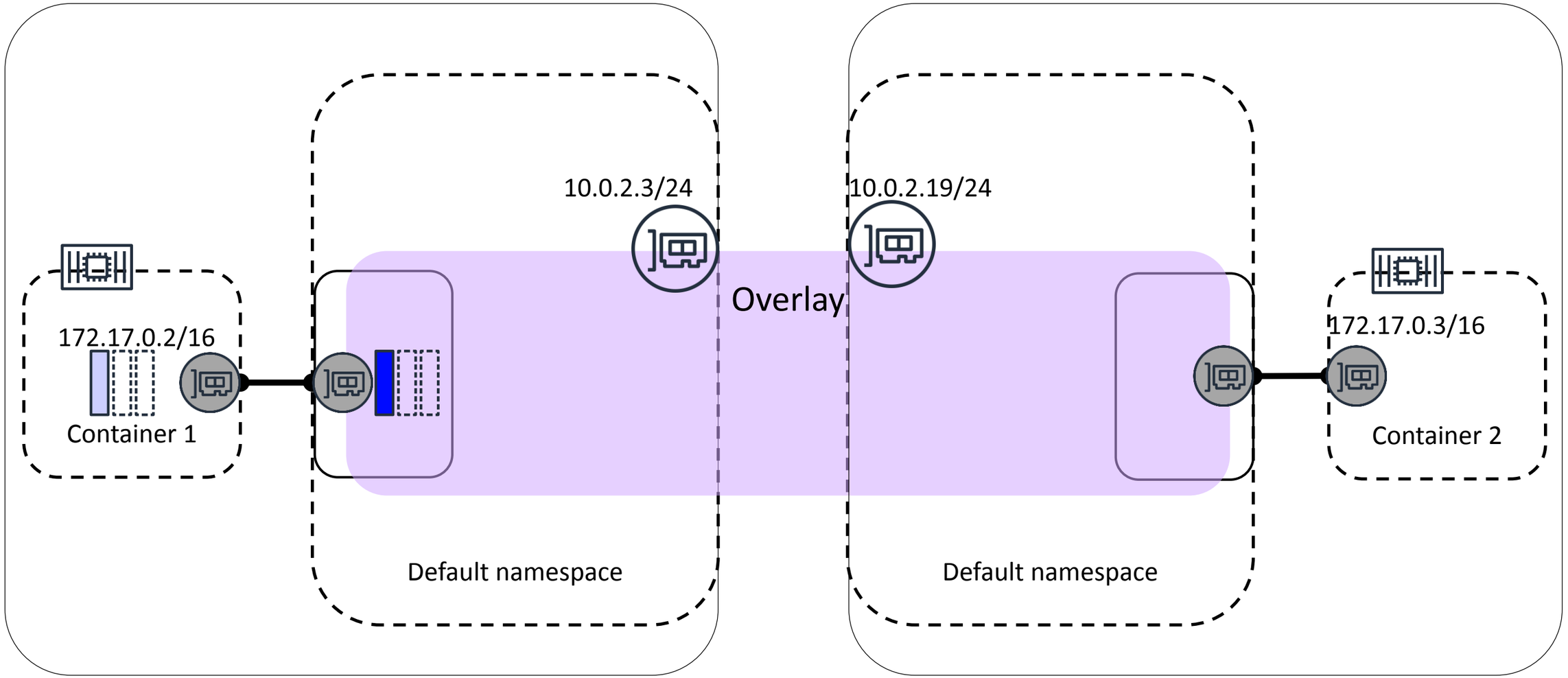
Isolation: Feng hosts applications from multiple teams in his organization. He has to ensure that network traffic from applications belonging to multiple teams are isolated from each other (including the network interfaces that are used).

Access control, VPC integration: Grace's security team has compiled a list of network policies using security groups and VPC ACLs. She wants to ensure that these are enforced on all of the applications she deploys.

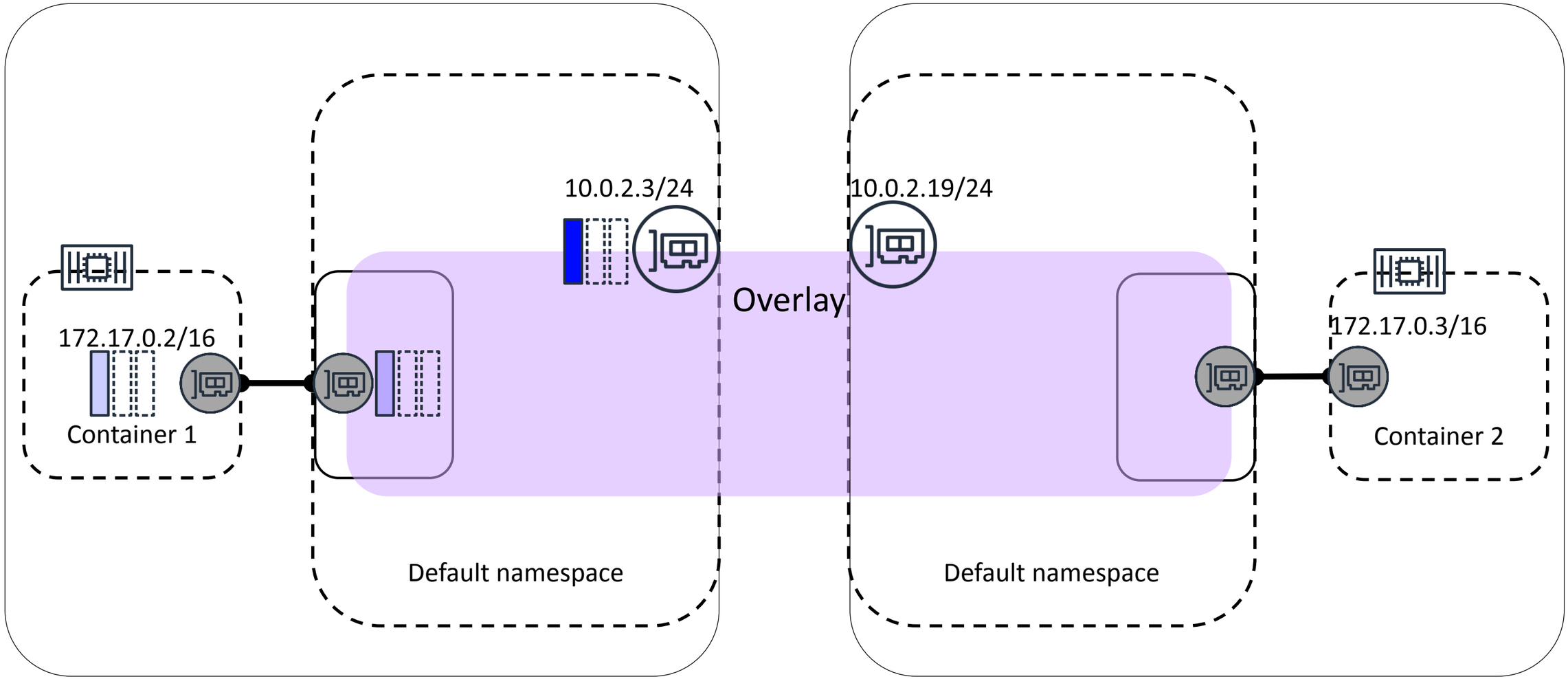
Packet traversal - Overlay [0/n]



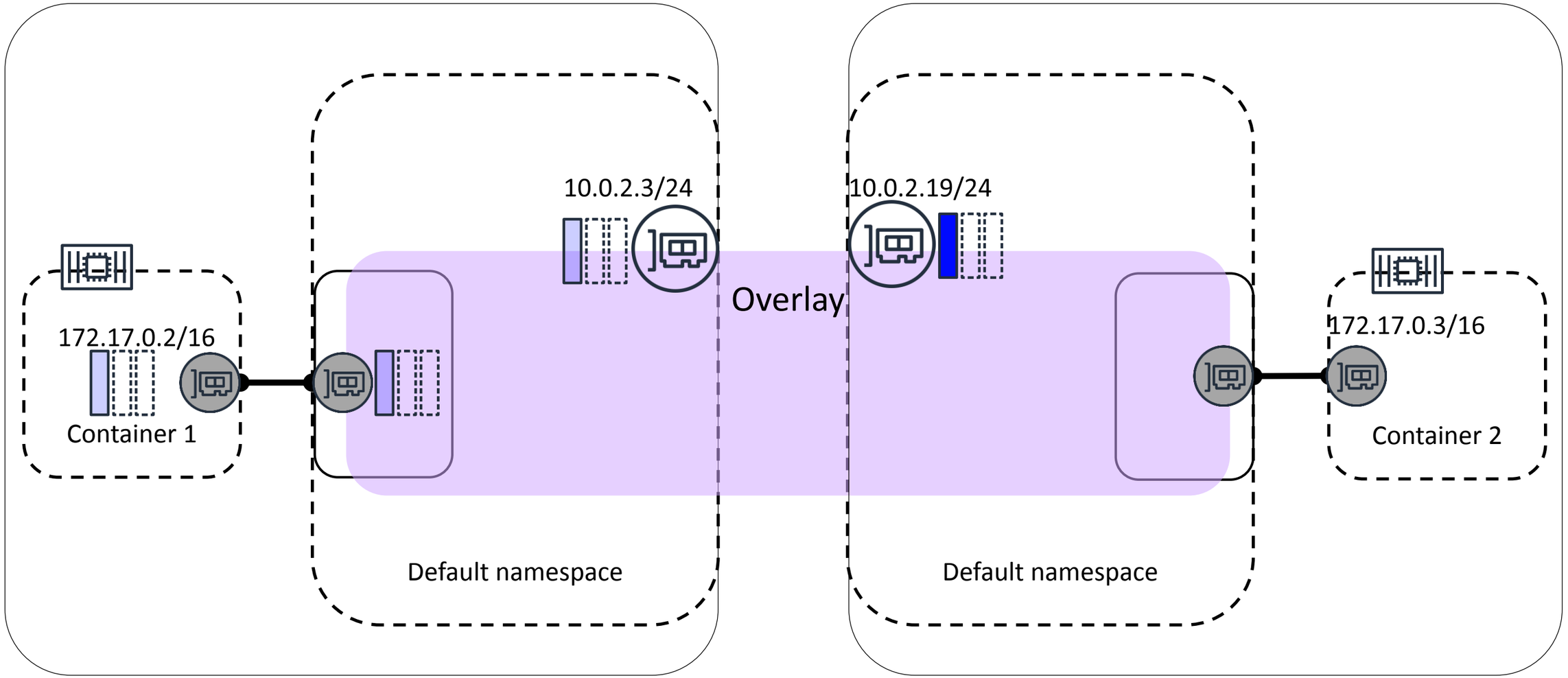
Packet traversal – Overlay [1/n]



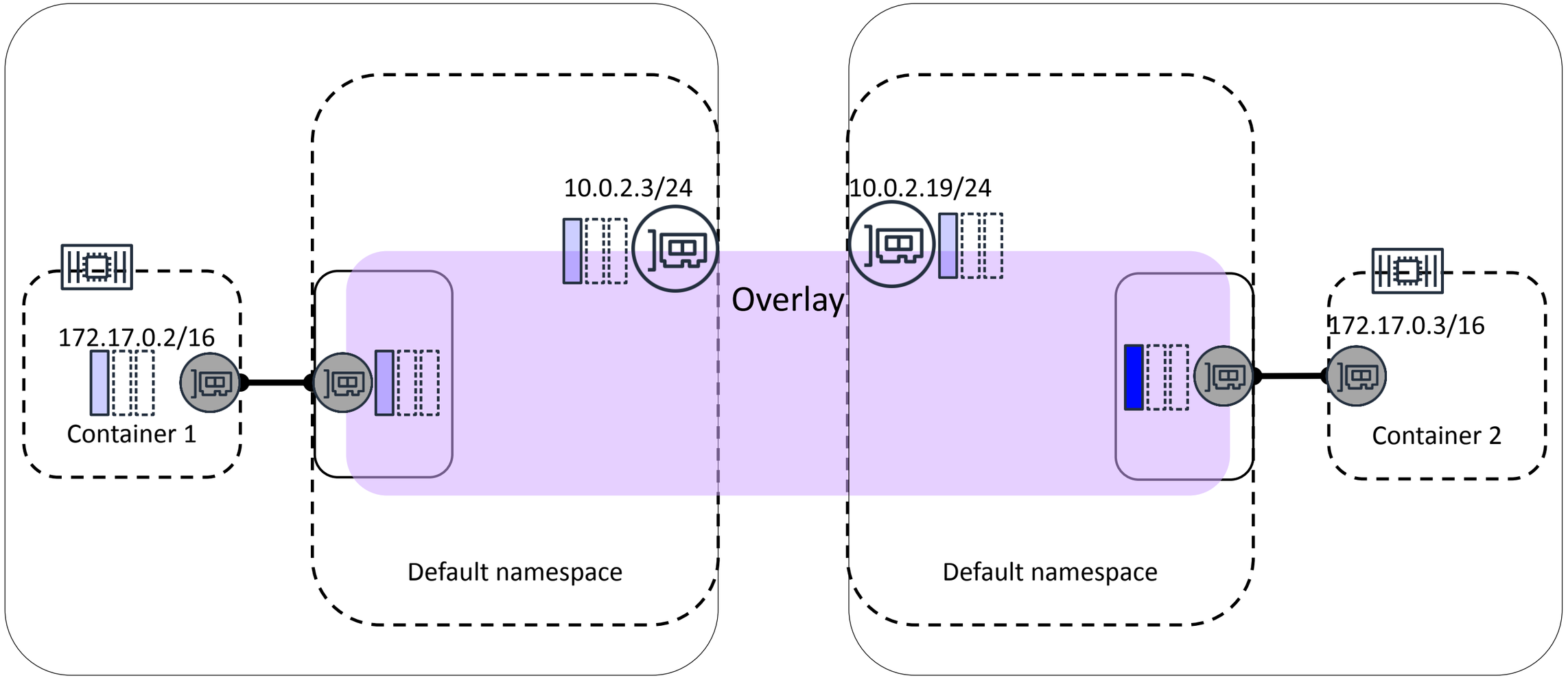
Packet traversal - Overlay [2/n]



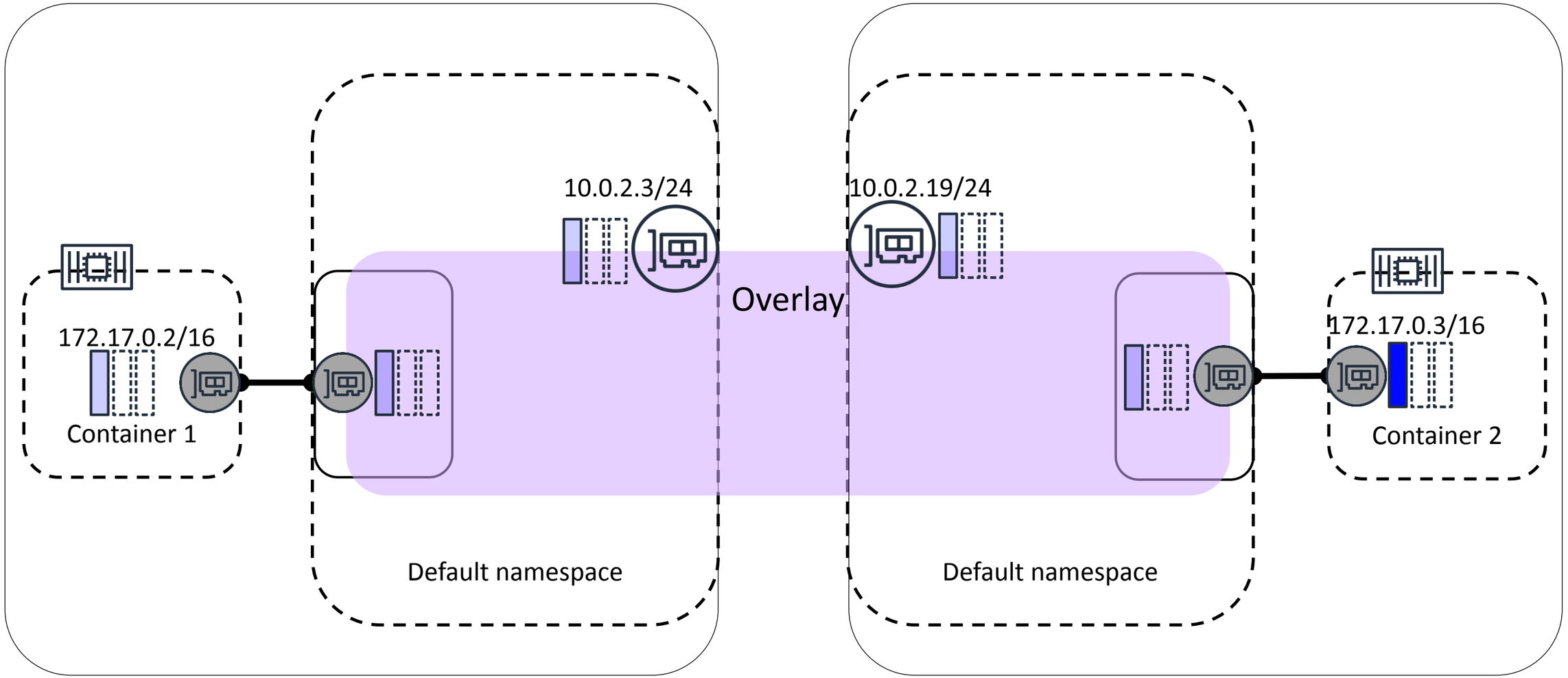
Packet traversal - Overlay [3/n]



Packet traversal - Overlay [4/n]



Packet traversal - Overlay [5/n]

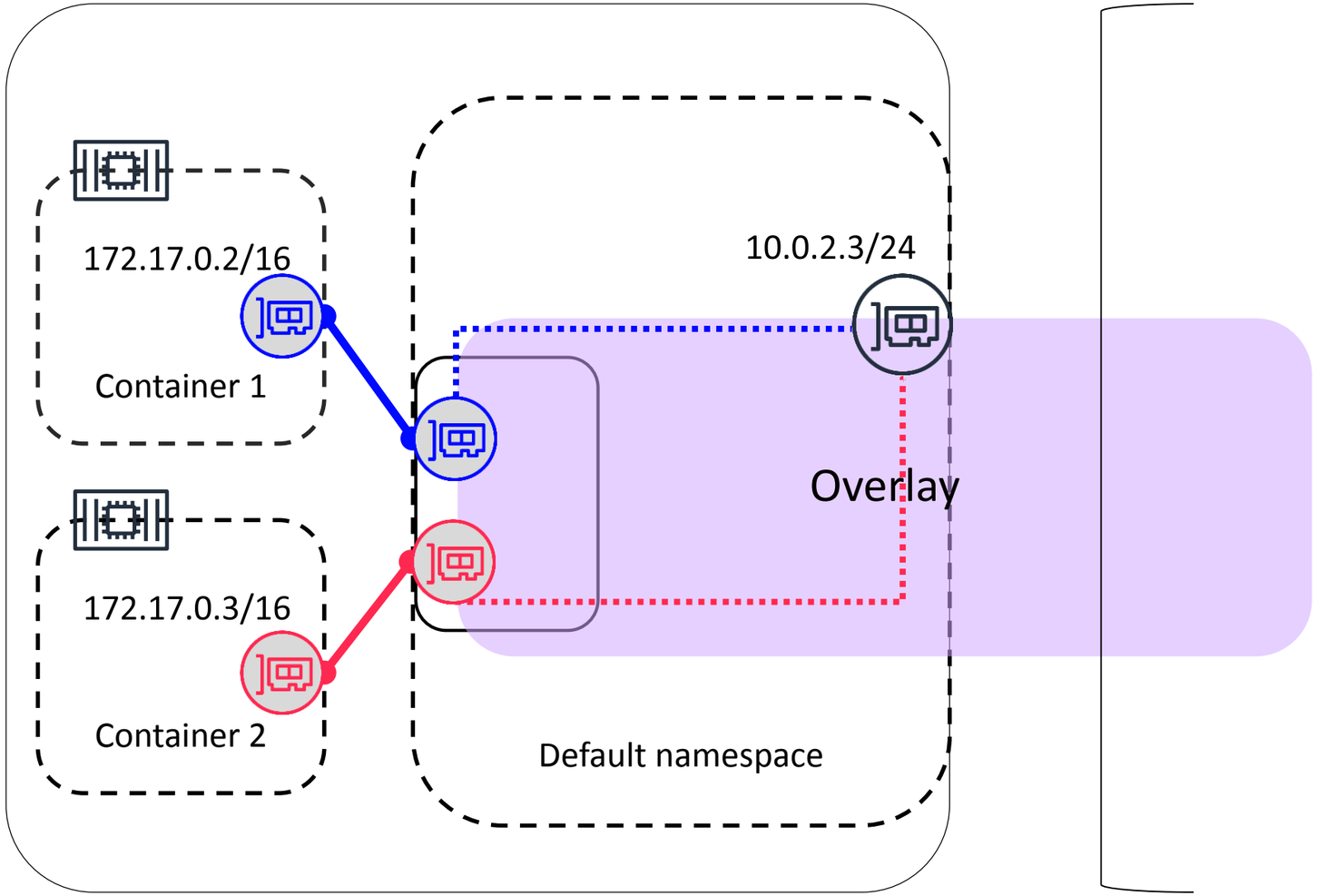


Overlay – Shared resources

Isolation & Access control

Hard to enforce when all containers are sharing a network interface

Malicious/misconfigured actors can easily gain access to the host



1. Discovery
2. Security
3. Performance, monitoring and on-demand delivery

Use-case – Performance, Monitoring & On-demand delivery

Performance: Habib runs HPC workloads that are very sensitive to latencies. He wants a highly optimized networking stack for her application, where different workers can easily locate one another, without any overhead of an overlay network.

Scaling, on-demand delivery: Iris's app went viral. She wants to scale her app by an order of magnitude to deal with increased load. She wants to ensure that networking resources start scaling to support this as well. Also, she doesn't want to pay for resources when traffic returns to steady-state.

Monitoring: James wants to quickly discover the microservice that's resulting increased end-to-end latency for his application. He wants to isolate the copy of the microservice that's causing this spike and weigh away traffic from the same.

Monitoring: Kajal wants to monitor the reachability of her applications as a health-check. This helps with resilient to networking events and increases the availability of her microservice.

VPC task/pod networking

VPC networking—ENIs

VPC is already acting as an overlay network

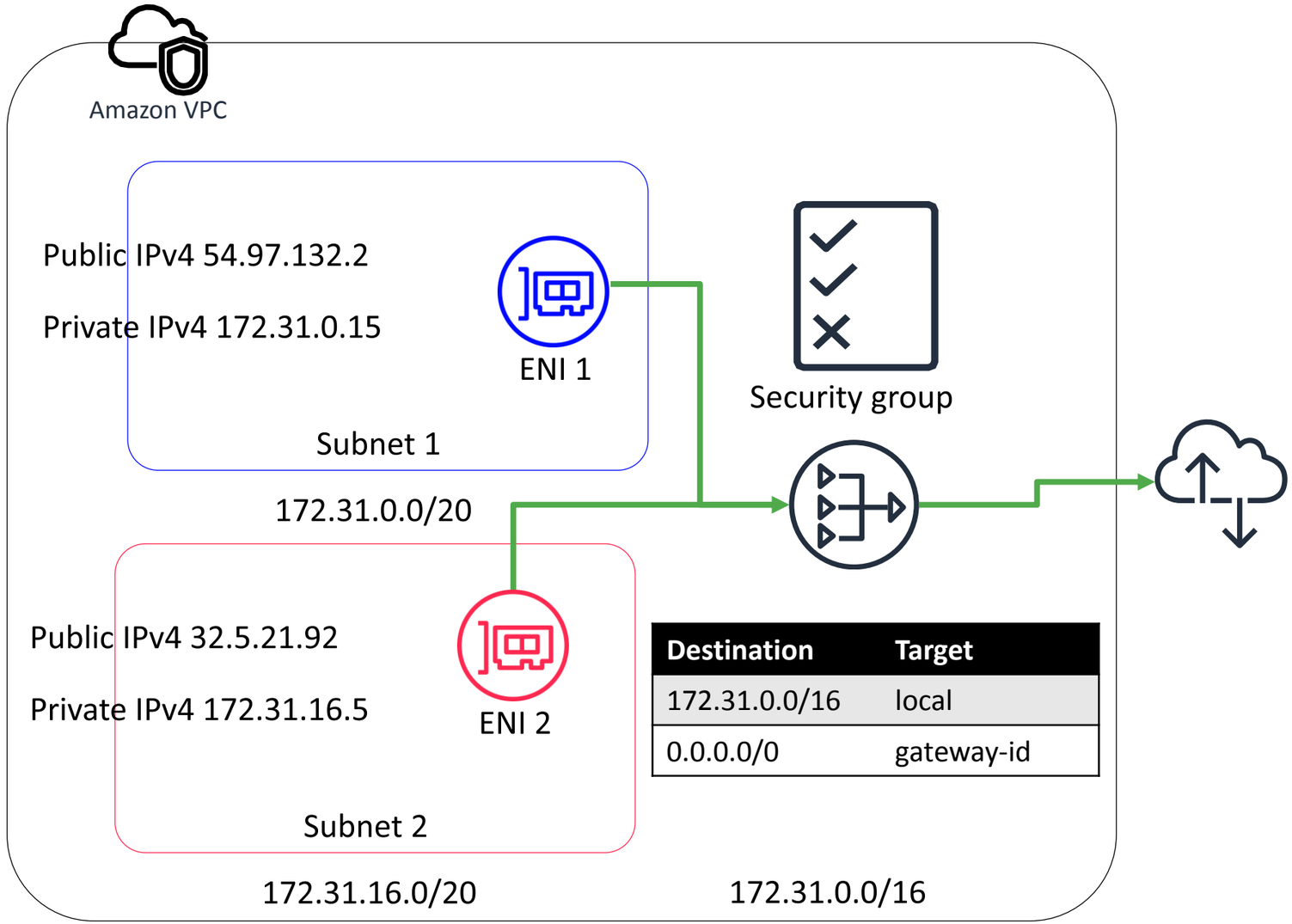
- No need to manage IP addresses
- LBs, Amazon Route 53 (Route 53) are already integrated
- IP addresses are routable within the VPC

Attachment is on-demand

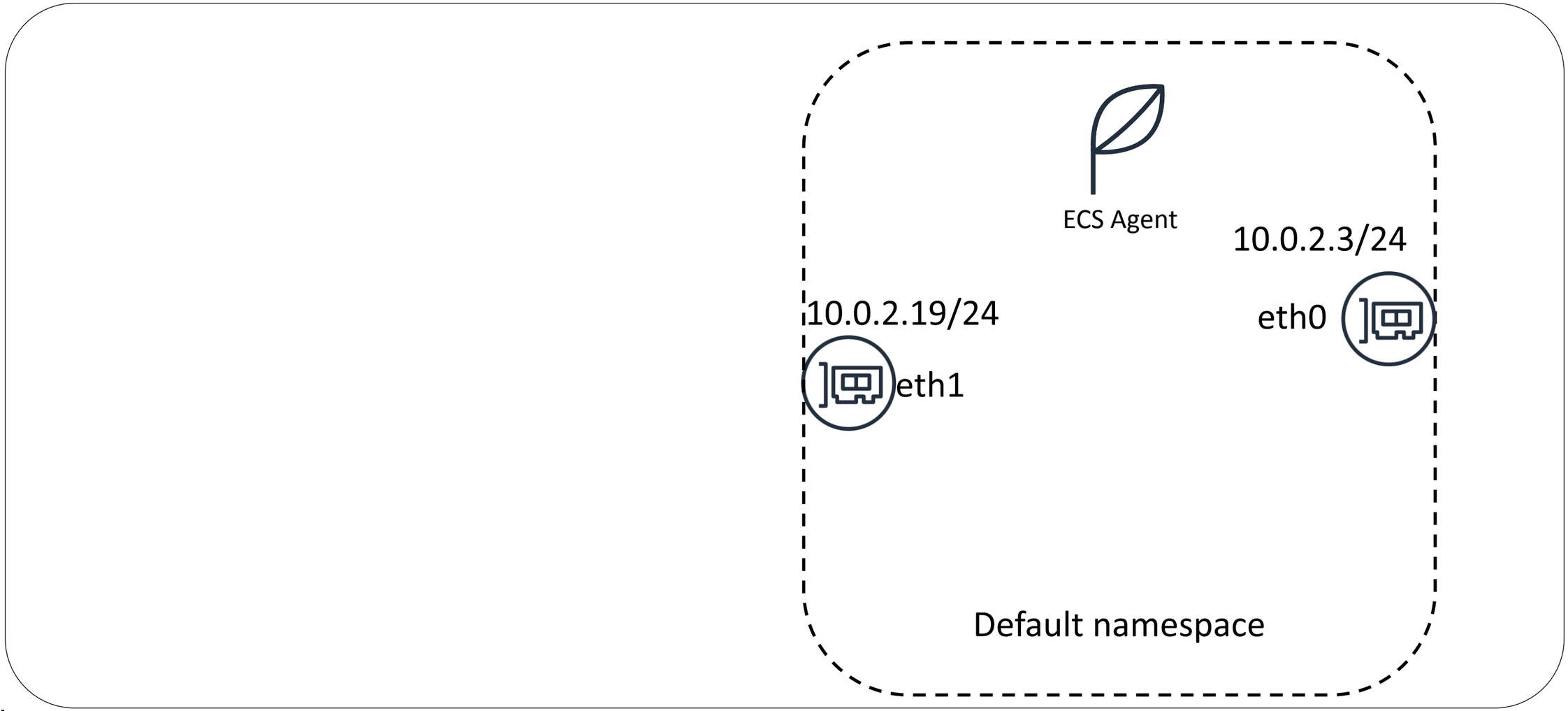
- Can be created on-demand and attached to instances

Security

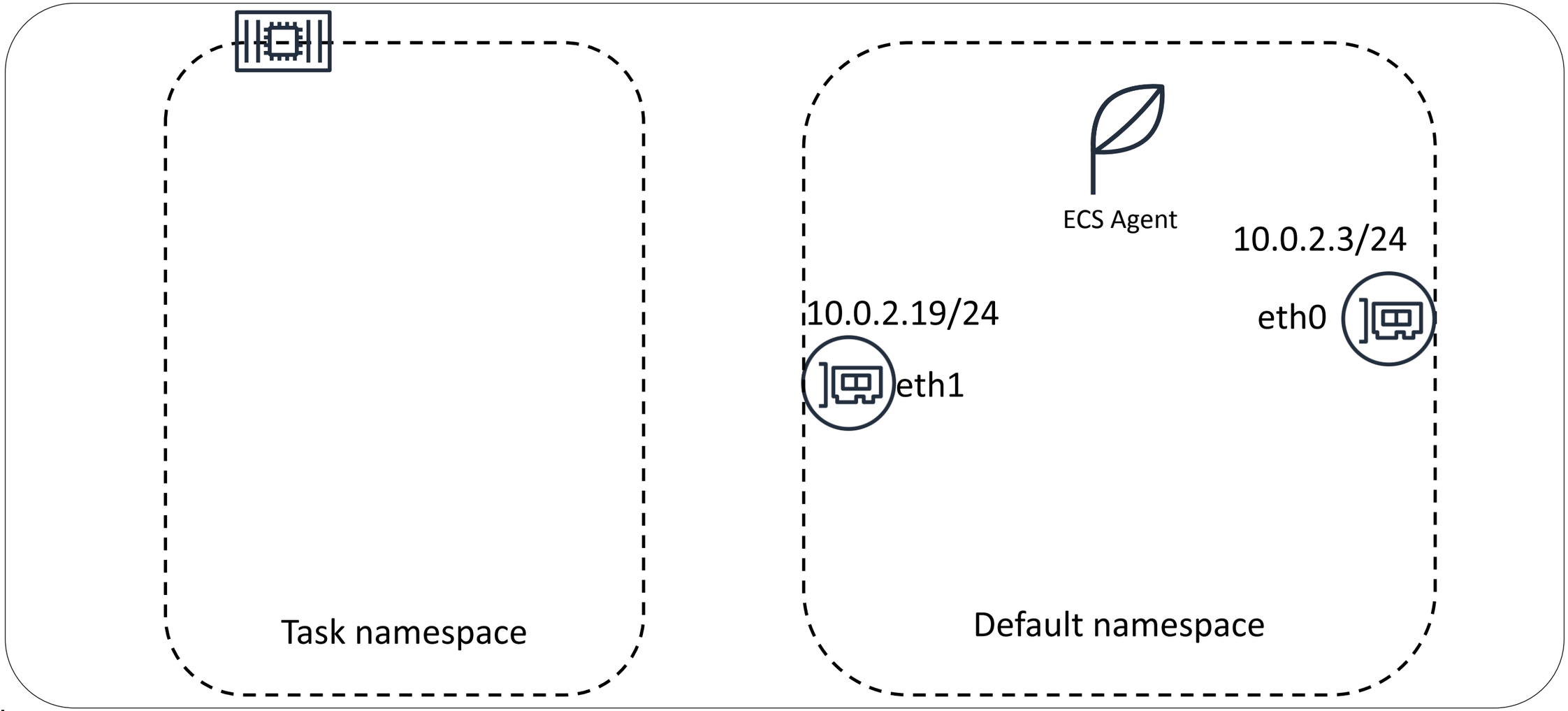
- Routing tables, security groups, and network ACLs are already built into VPC
- Network traffic can be monitored using flow logs



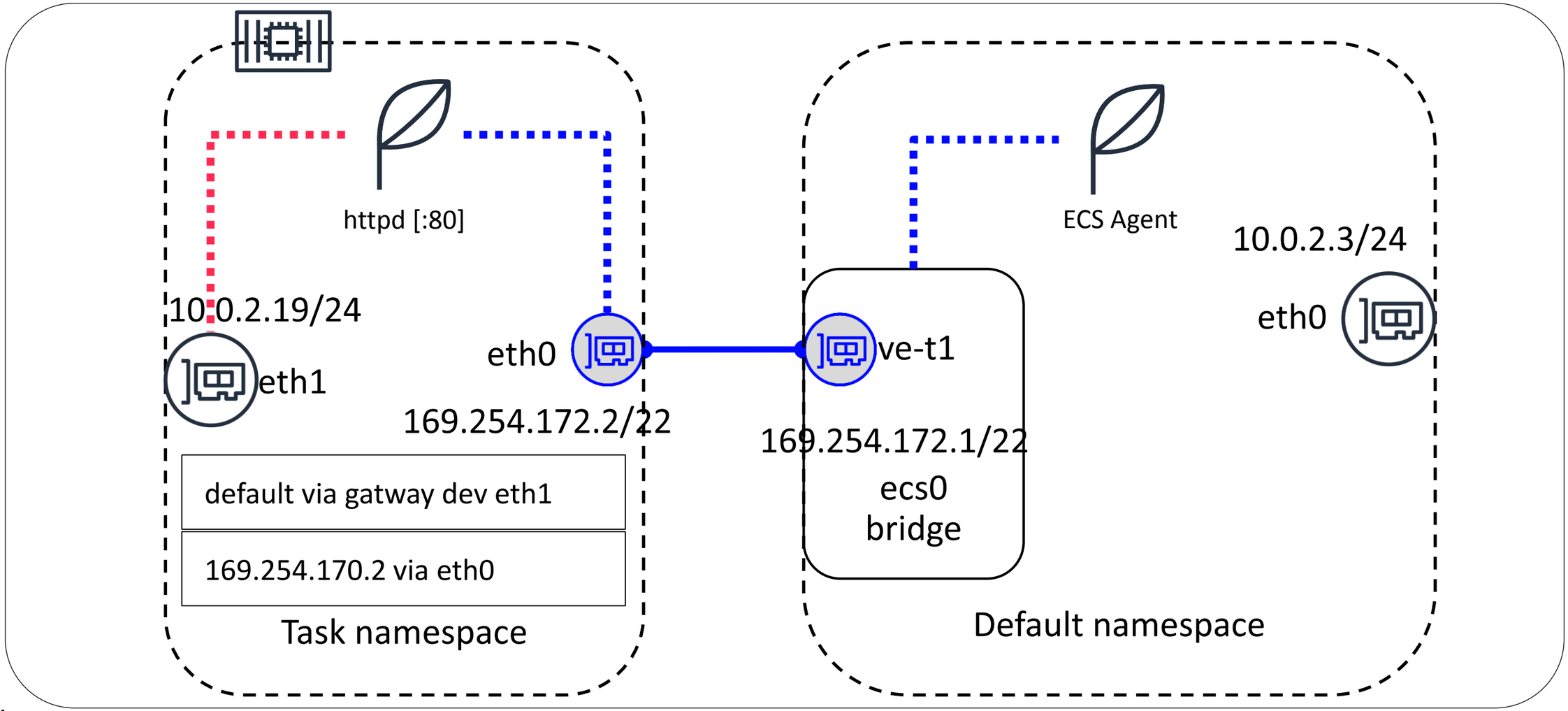
Networking layout – ‘awsvpc’ mode



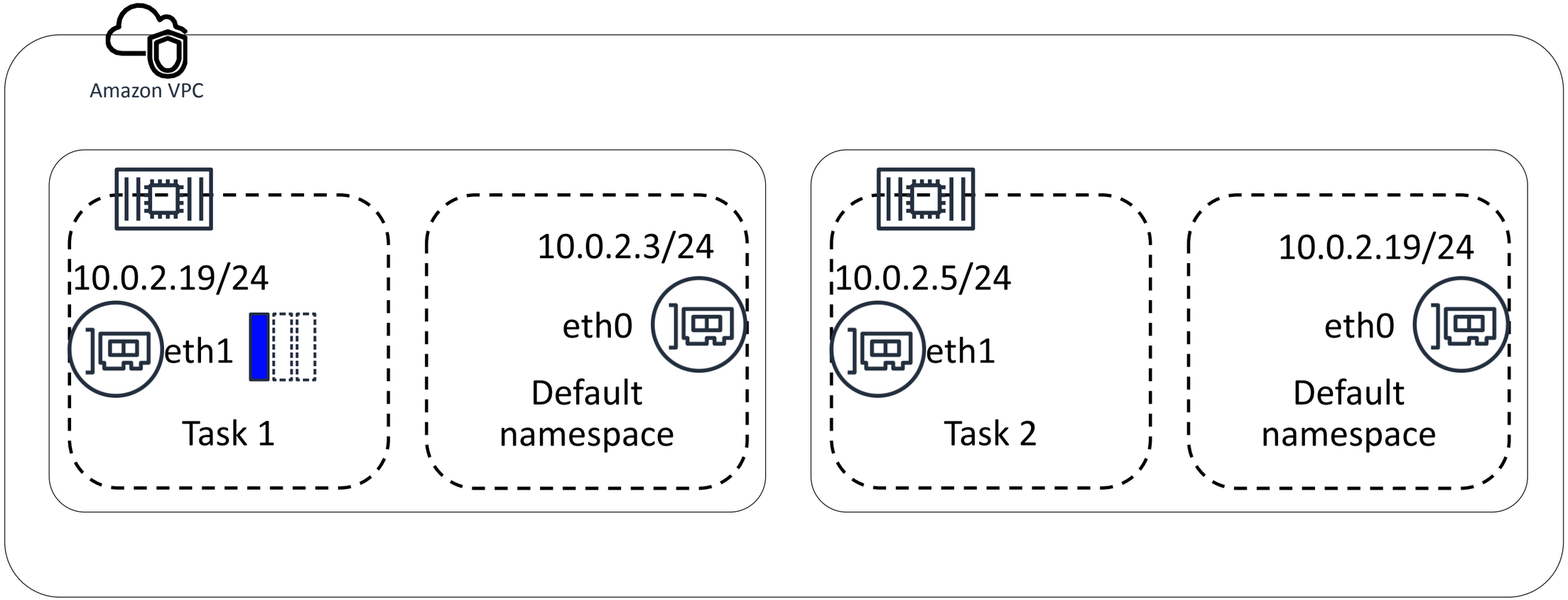
Networking layout – ‘awsvpc’ mode



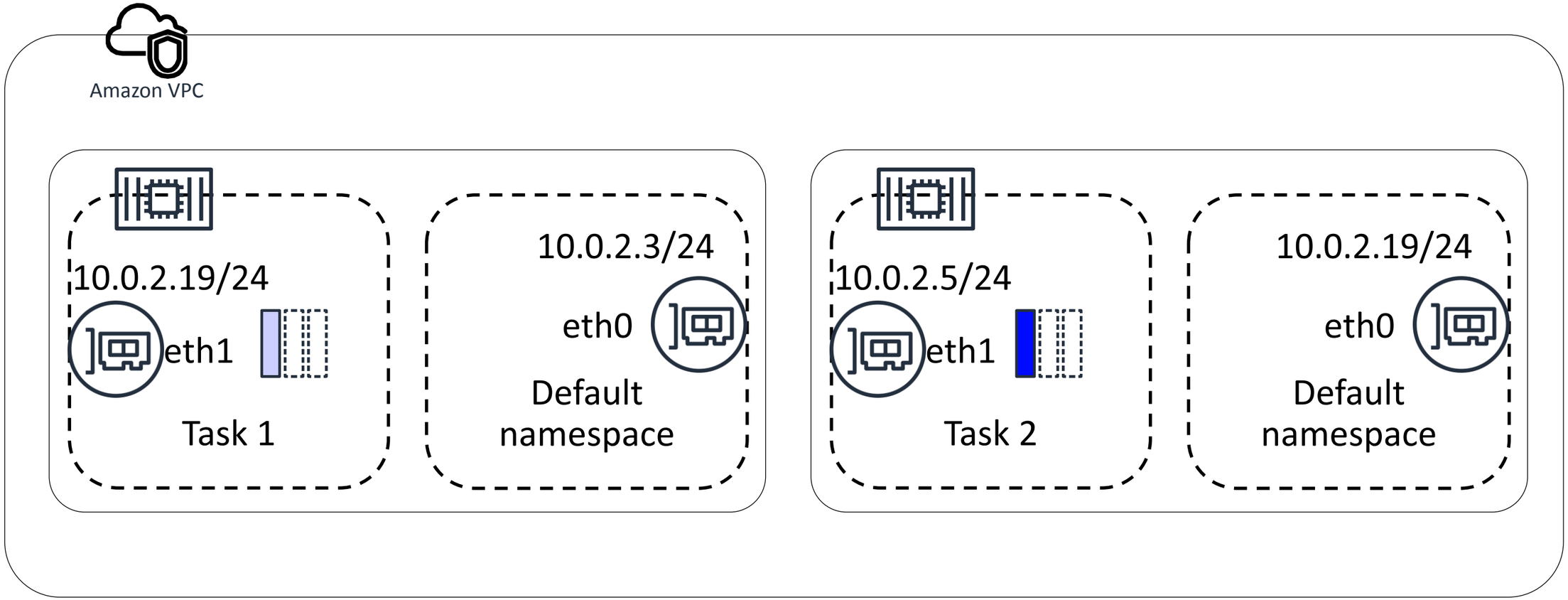
Networking layout – ‘awsvpc’ mode



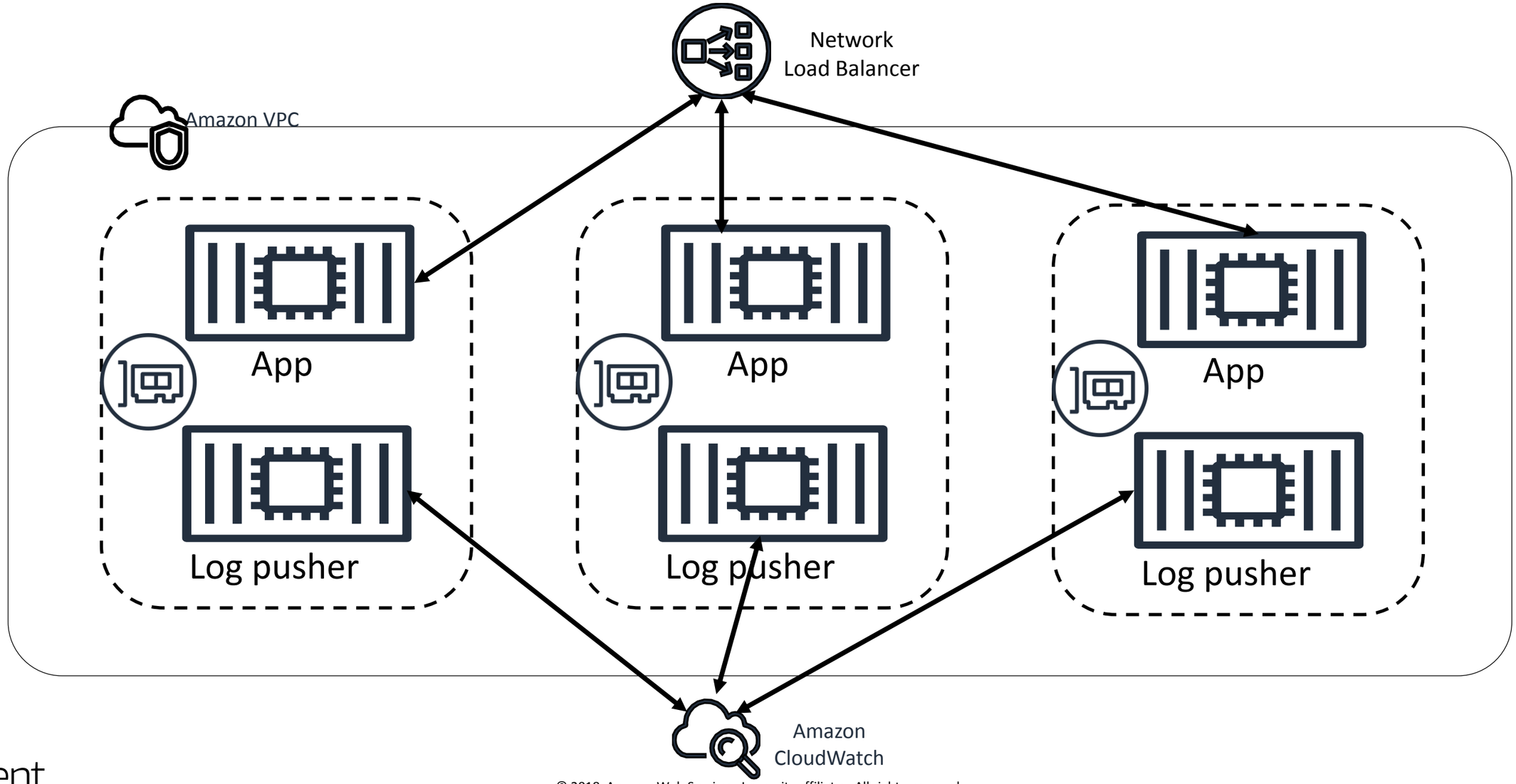
Packet traversal – ‘awsvpc’ mode



Packet traversal – ‘awsvpc’ mode



Fargate ‘awsvpc’ mode

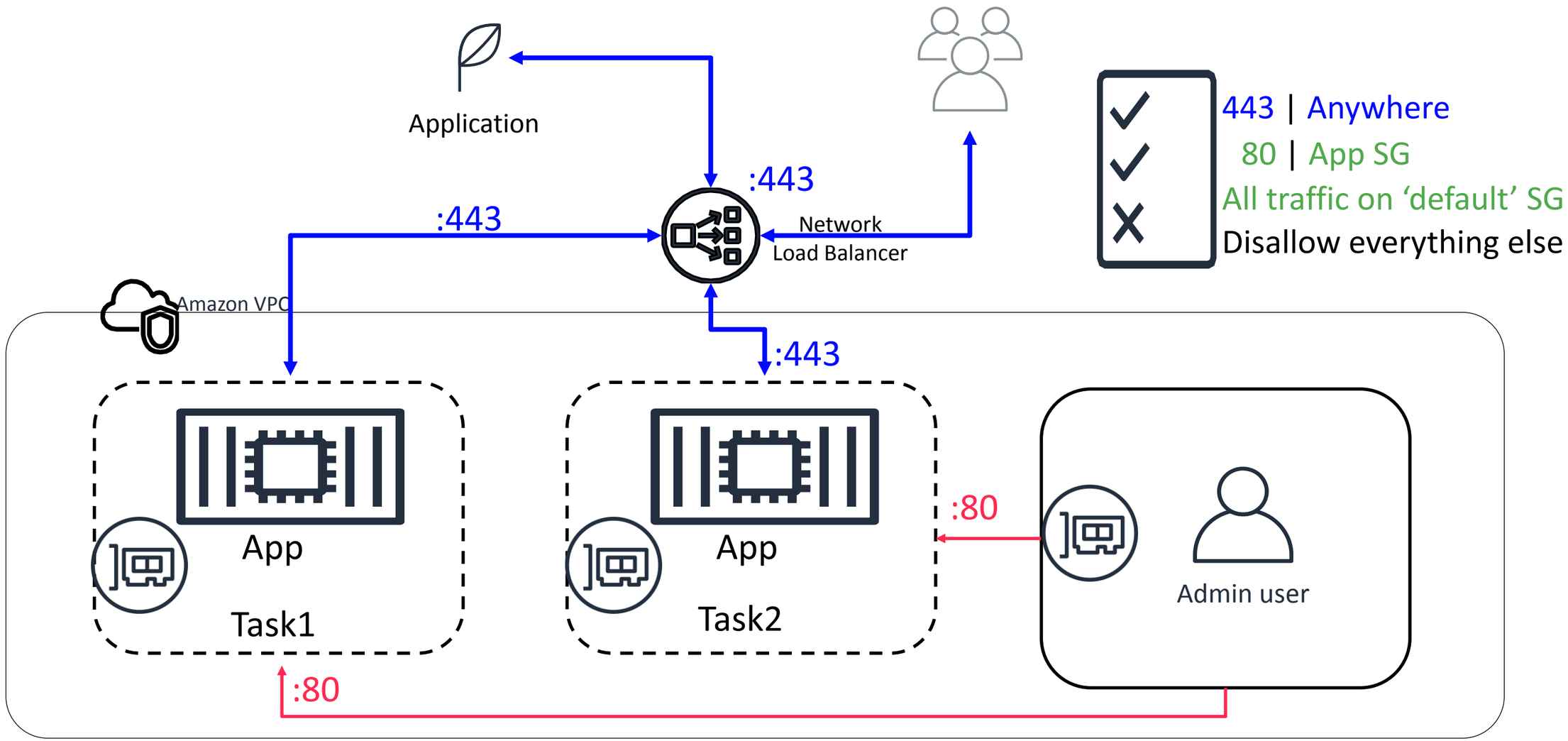


Example: Restricting access using security groups

```
$ aws ec2 describe-security-groups \
       --group-ids sg-0dbde04ab3d67ab73

SECURITYGROUPS  for accessing admin APIs
sg-0dbde04ab3d67ab73    webapp admin sg

IPPERMISSIONS           -1
USERIDGROUPPAIRS        sg-252e9040
123456789012

IPPERMISSIONS   8100    tcp     8100
USERIDGROUPPAIRS        sg-094b4a050931890ce
123456789012

IPPERMISSIONSEGRESS     -1
IPRANGES        0.0.0.0/0
```

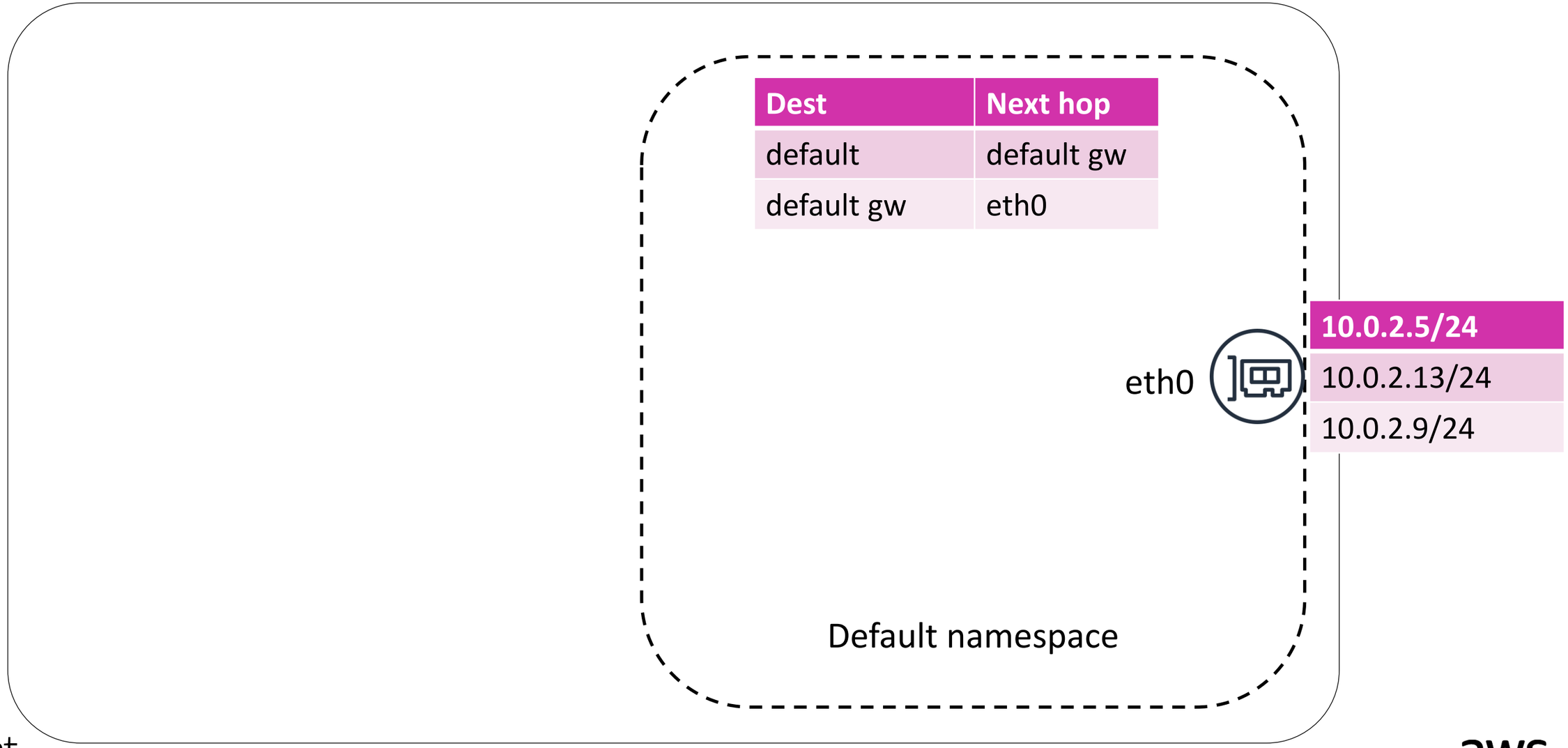
```
$ aws ec2 describe-security-groups \
       --group-ids sg-094b4a050931890ce

SECURITYGROUPS  for accessing webapp
externall  sg-094b4a050931890ce   webapp sg

IPPERMISSIONS   443    tcp     443
IPRANGES        0.0.0.0/0

IPPERMISSIONSEGRESS     -1
IPRANGES        0.0.0.0/0
```

Access control in ‘awsvpc’ mode

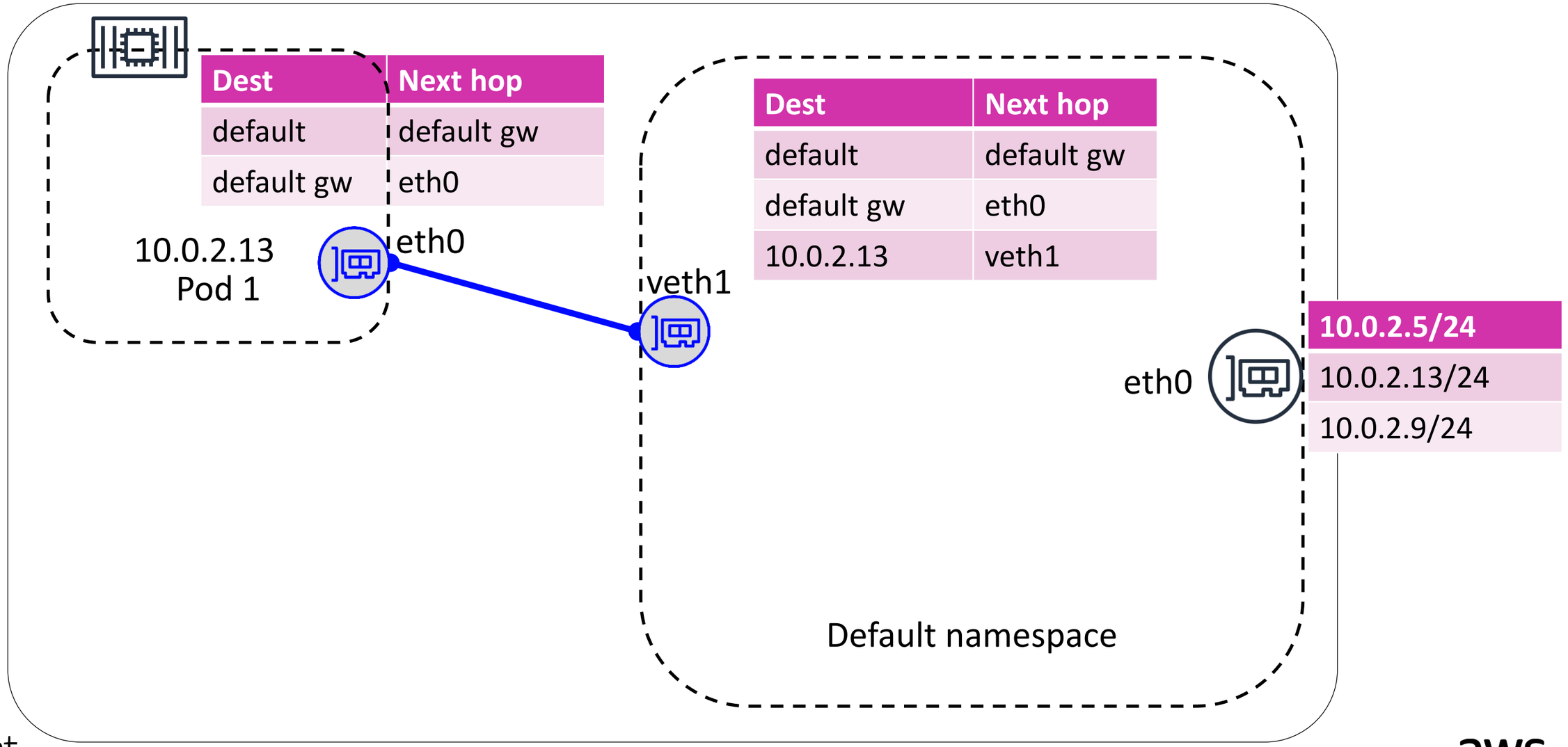


Demo

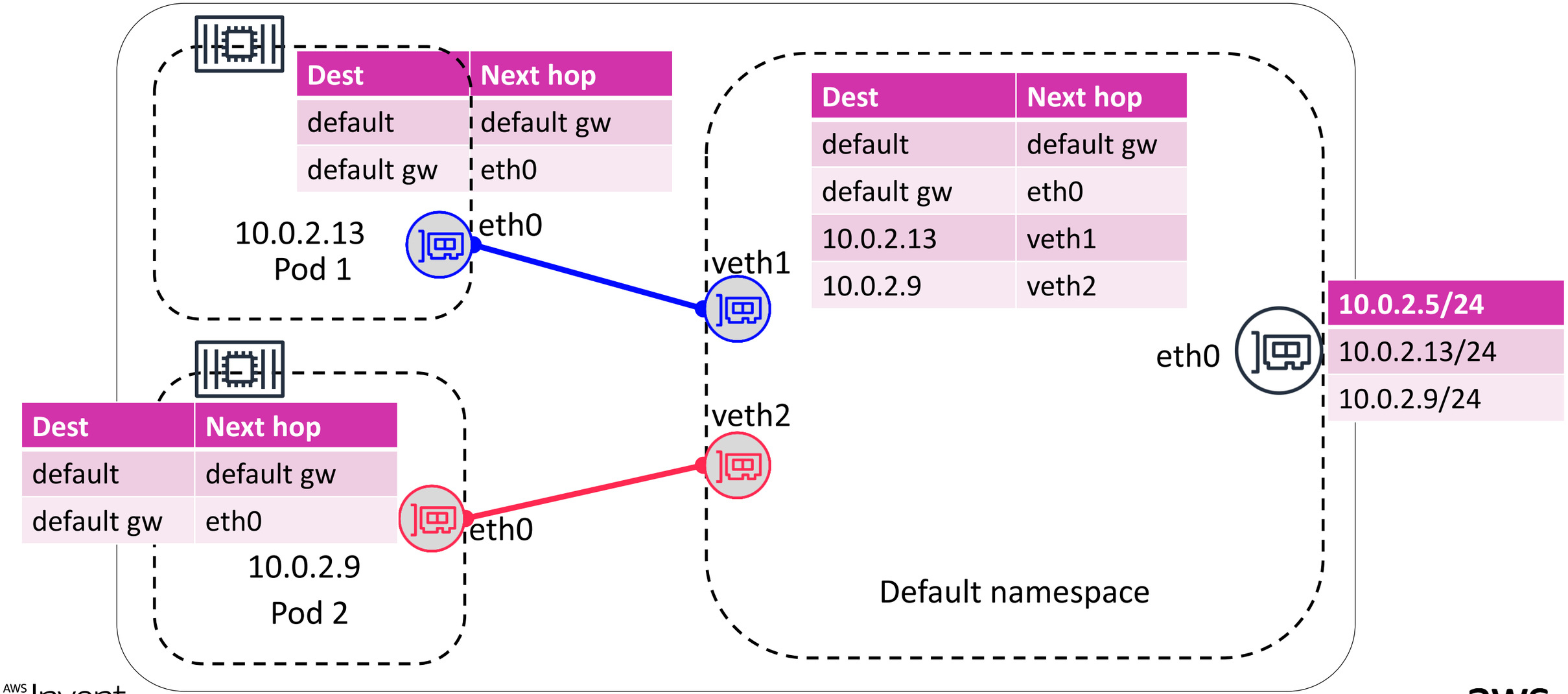
Amazon EKS VPC Networking



Amazon EKS VPC Networking



Amazon EKS VPC Networking



How did we do?

Use-case - Discovery

Within an instance: Alice deploys an application that consists of multiple containers, which are deployed together (colocated on an instance). Container-A wants to communicate with Container-B using Container-B's name.

*>> Container links (**bridge** mode) or localhost:port (**awsvpc** mode in ECS, EKS pod ENI networking)*

Across instances: Bob has two microservices Service-A & Service-B. Service-A wants to call remote APIs supported by Service-B. Service-B should be reachable by containers under Service-A.

>> Any supported networking mode in ECS, EKS, Fargate; Use LB or service-discovery integration

Use-case - Discovery

Consistent addressing: Cara creates service-1 using container-A, which exposes port 80. Cara should be able to run multiple instances of service-1 on same container instance without worrying about port conflicts.

*>> **awsvpc** mode in ECS, EKS pod ENI networking*

VPC integration: Deepak runs his own DNS server in his VPC. He wants his applications to get domain names and IPv4 addresses from this domain.

*>> **awsvpc** mode in ECS integrated with all of the VPC options, including custom DNS/DHCP option sets*

Use-case - Security

Access control: Elizabeth wants to ensure that none of the applications hosted on the infrastructure she maintains expose port 22 to the internet.

*>> **awsvpc** mode, use security groups*

Isolation: Feng hosts applications from multiple teams in his organization. He has to ensure that network traffic from applications belonging to multiple teams are isolated from each other (including the network interfaces that are used).

*>> **awsvpc** mode in ECS*

Access control, VPC integration: Grace's security team has compiled a list of network policies using security groups and VPC ACLs. She wants to ensure that these are enforced on all of the applications she deploys.

*>> **awsvpc** mode, use security groups; EKS pod ENI networking with network policies*

Use-case – Performance, Monitoring & On-demand delivery

Performance: Habib runs HPC workloads that are very sensitive to latencies. He wants a highly optimized networking stack for her application, where different workers can easily locate one another, without any overhead of an overlay network.

*>> **awsvpc** mode in ECS; EKS pod ENI networking*

Scaling, on-demand delivery: Iris's website went viral overnight. She wants to scale her application by an order of magnitude. She wants to ensure that networking resources start scaling to support this as well. Also, she doesn't want to pay for resources when traffic returns to steady-state.

>> Any supported networking mode in ECS, EKS, Fargate; orchestrator ensures that resources are torn down on scale-in

Use-case – Performance, Monitoring & On-demand delivery

Monitoring: James wants to quickly discover the microservice that's resulting increased end-to-end latency for his application.

>> Partner solutions; Appmesh

He wants to isolate the copy of the microservice that's causing this spike and weigh away traffic from the same.

>> Use LB, service-discovery integration with any supported networking mode; Appmesh

Monitoring: Kajal wants to monitor the reachability of her applications as a health-check. This helps with resilient to networking events and increases the availability of her microservice.

>> Use LB or service-discovery integration with any supported networking mode

Service mesh

Related links

https://github.com/nathanpeck/awesome-ecs

https://github.com/awslabs/amazon-ecs-nodejs-microservices

https://medium.com/containers-on-aws/using-aws-application-load-balancer-and-network-load-balancer-with-ec2-container-service-d0cb0b1d5ae5

https://medium.com/containers-on-aws/how-to-setup-service-discovery-in-elastic-container-service-3d18479959e6

https://docs.aws.amazon.com/AmazonECS/latest/developerguide/service-load-balancing.html

https://docs.aws.amazon.com/AmazonECS/latest/developerguide/create-service-discovery.html

https://aws.amazon.com/blogs/compute/under-the-hood-task-networking-for-amazon-ecs/

Thank you!

Anirudh Aithal
aithal@amazon.com, @aaithal

AWS re:Invent

aws