Introducing ECS Service Discovery

Communication service for building containerized microservices on AWS

Nathan Peck

April 23, 2018

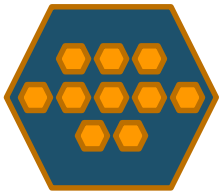
aws

Application communication is evolving

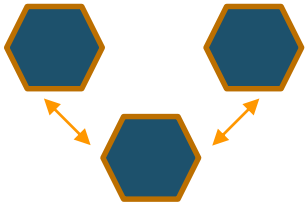
Across the room

Across cities / continents 1:1

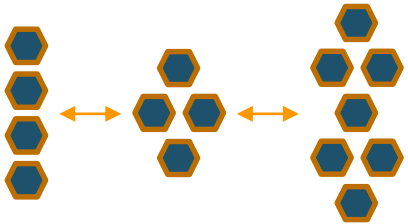
Dynamic name,
number, and location



Functional calls



Known endpoints, APIs

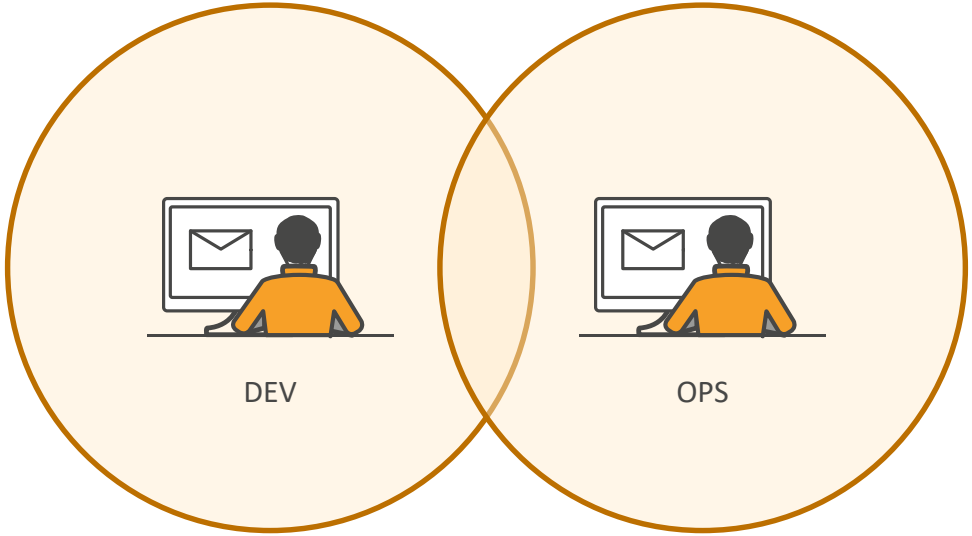


Find endpoints,
then connect

aws

Developers need to connect microservices

Build apps
invoking other services
by name



Ensure that service
name resolves to
correct IP/port

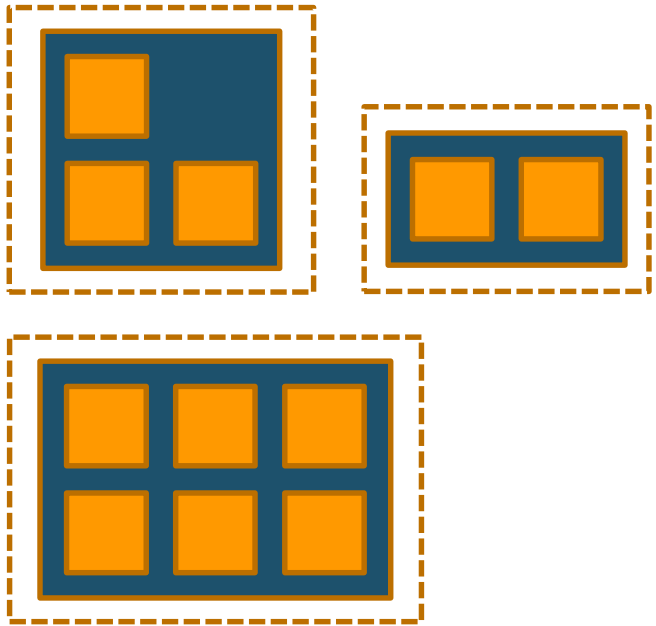
aws

What is Service Discovery?

“Where is Service X?”

Friendly name -> IP + port

E.g., app: {10.0.4.5:8080, 10.0.4.6:8080 }



aws

Why is it non trivial?

Dynamic by design:

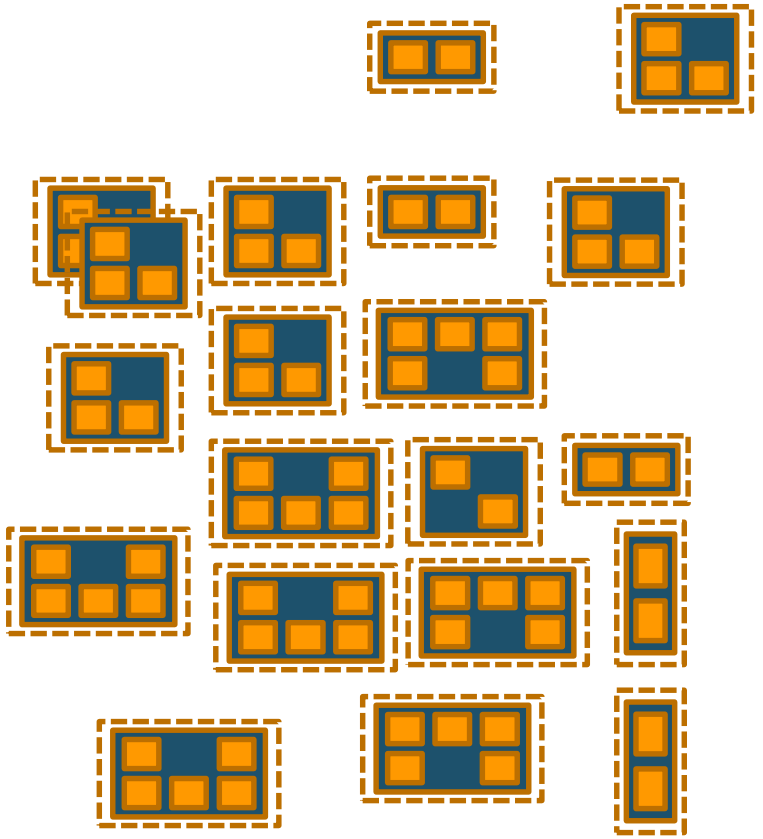
Number of containers & instances

Auto assigned IP addresses & ports

Placement, scheduling, scaling

Deployments and upgrades

Health and connectivity



aws

Decision criteria

Service Registry

Where is info about services stored?

Registration mechanism

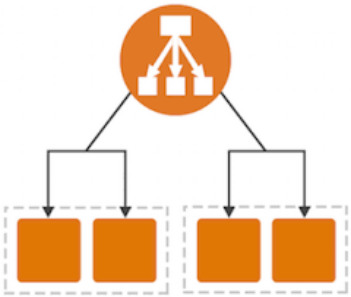
How is a service added to the registry?

Discovery mechanism

How does an application send traffic to a service from the registry?

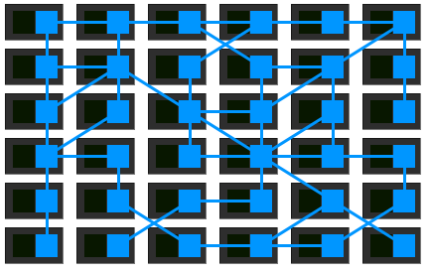
aws

Current patterns require install, setup and management



Load Balancers

Key-value store



Service Mesh

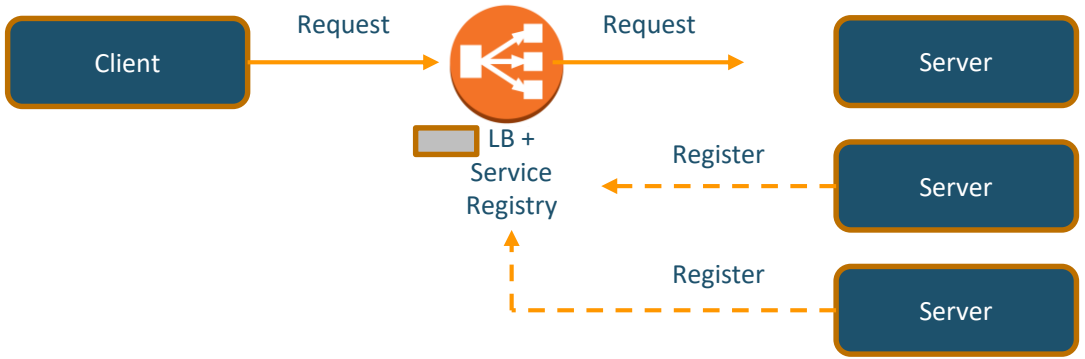
aws

Load Balancers – Server Side Discovery

E.g.,
Load Balancer

Benefits
Client is simpler

Drawbacks
Install, manage
Availability, capacity
More hops



DNS based – Client Side Discovery

E.g.,
Route 53 Based

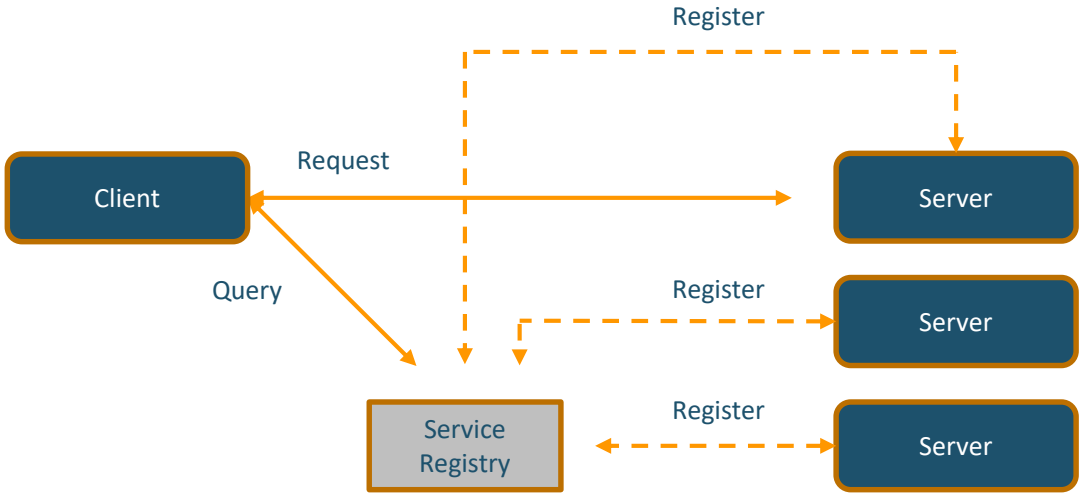
Benefits

Fewer hops

Drawbacks

Client must be registry aware

Client implements discovery logic



aws

Requires registrations by agents

E.g.,
Consul, Etcd and Zookeeper

Benefits

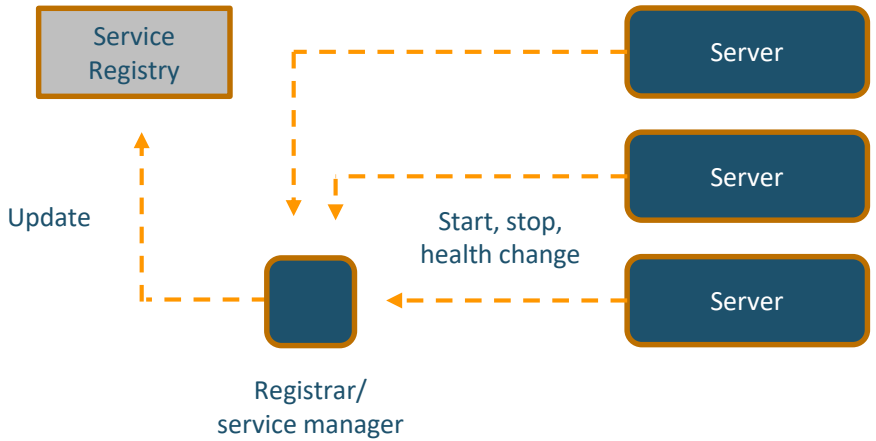
Registrar registers, unregisters and performs health checks

Service less complex

Drawbacks

State may not reflect whether service can handle requests, only active or unavailable

Install, configure and manage other component, unless it is a part of infrastructure



aws

Requires registrations by agents

E.g.,
Consul, Etcd and Zookeeper

Benefits

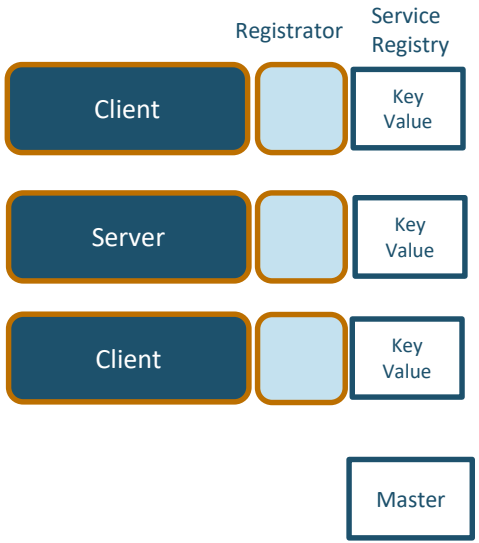
Registrar registers, unregisters and performs health checks

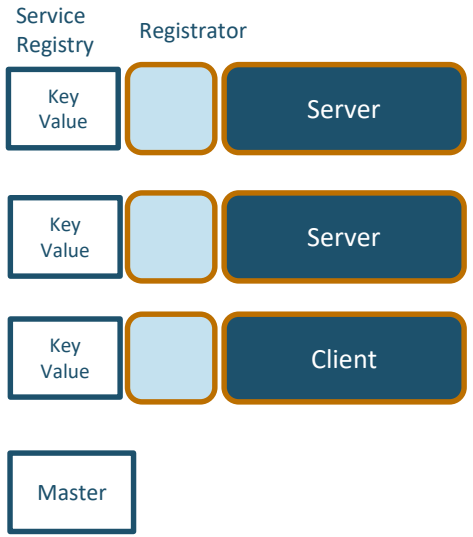
Service less complex

Drawbacks

State may not reflect whether service can handle requests, only active or unavailable

Install, configure and manage other component, unless it is a part of infrastructure

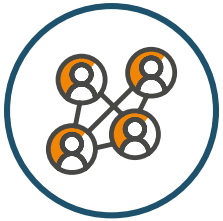




This should be easier!

Predictable Names for services

Auto updated with latest, healthy IP, port



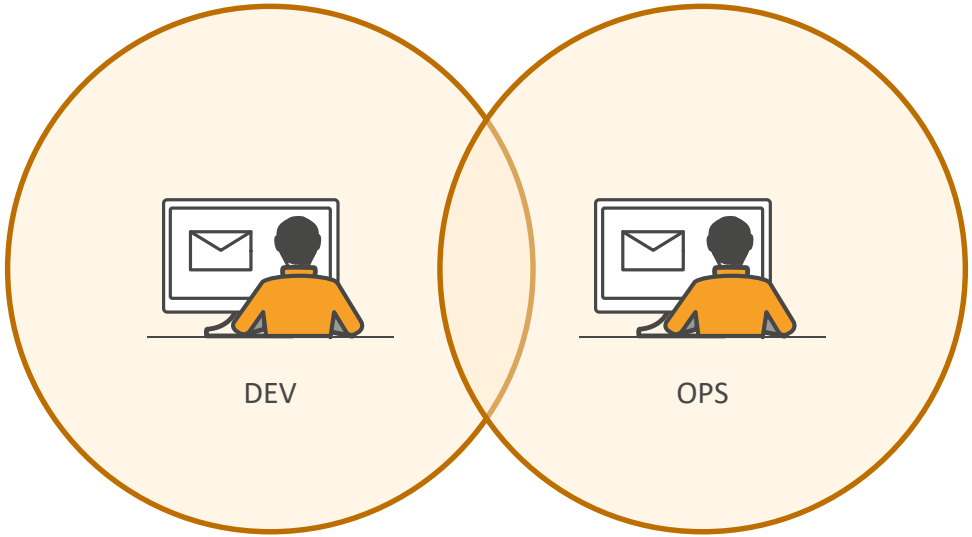
Managed: No overhead of installation or monitoring

High availability, high scale

Extensible: Flexible boundaries for auto discovery

aws

Introducing managed service discovery for ECS

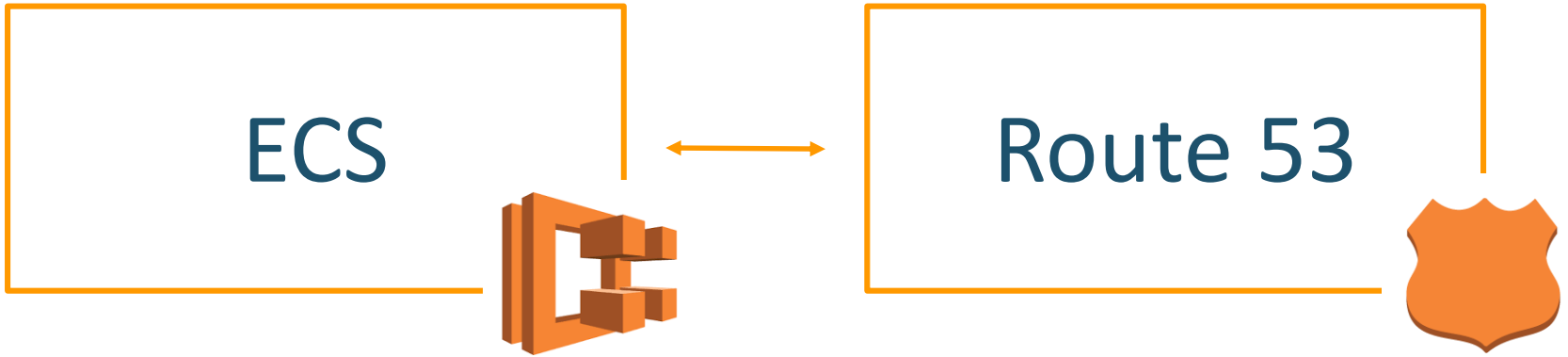


You build apps where services are invoked by name & name resolves to IP/port dynamically

You turn on service discovery during deployment — service creation

aws

ECS service discovery is powered by Route 53



ECS updates service registry based on naming convention, task registrations, de-registrations and health

Route 53 provides Service Registry

aws

Demo

https://servicediscovery.ranman.com/

Enables these use cases

1

Blue green deployments

- *myapp.staging.local*
- *myapp.prod.local*
- Private IP
- abstract cluster details

2

Internal micro services

- *web.myapp.local*
- *Expose Private IP*

3

External micro services

- *web.myapp.mycompany.com*
- *Expose public IP or ELB EIP*
- *network + container health check*

aws

Enables these use cases

4

Across ECS & Kubernetes

- Service1.myapp.ecs
- Service2.myapp.eks

5

Across ECS &
AWS & On-Prem

- *Service1.myapp.ecs*
- *Service2.myapp.ec2*
- *Service3.myapp.onprem*

6

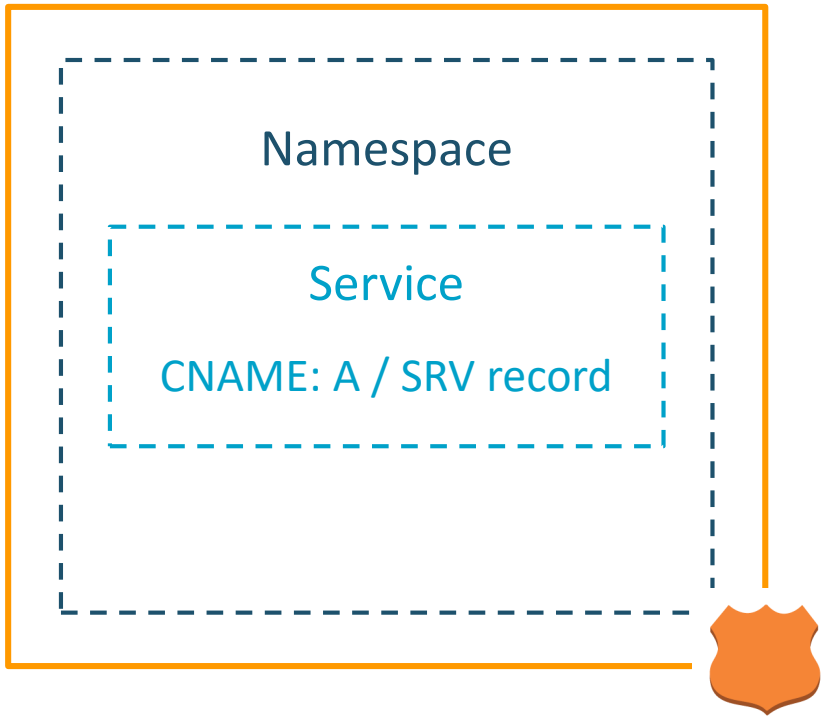
Expose to service mesh

- *Service1.myapp.local*
- *Service2.myapp.local*

aws

What’s New?

Route 53 provides Service Registry

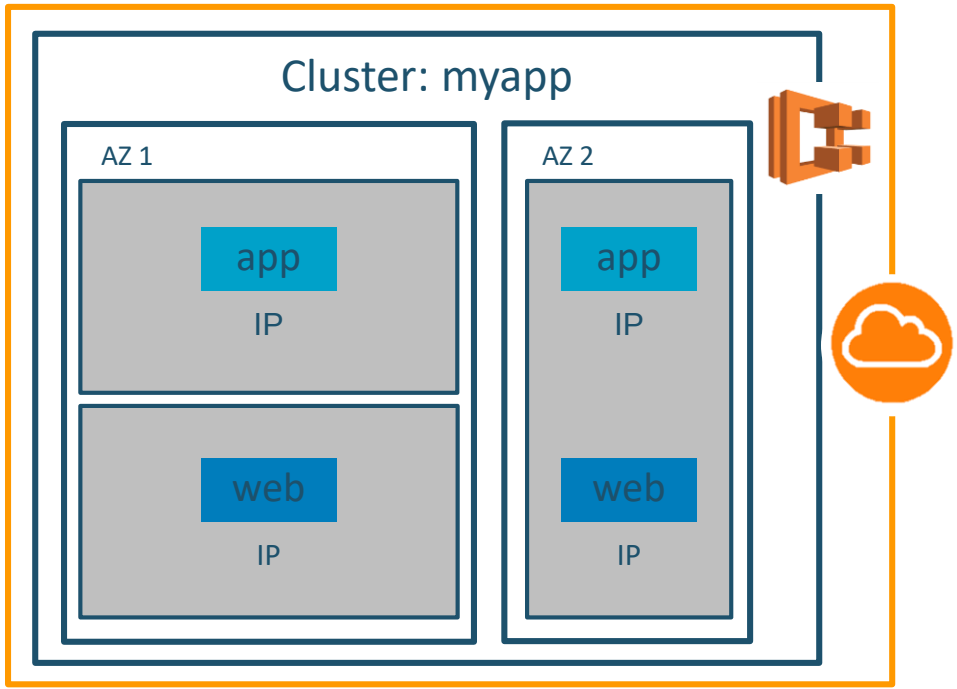


Route 53 provides APIs to create

- Namespace
- CNAME per service autoname
- A records per task IP
- SRV records per task IP + port

aws

ECS schedules & places service endpoints



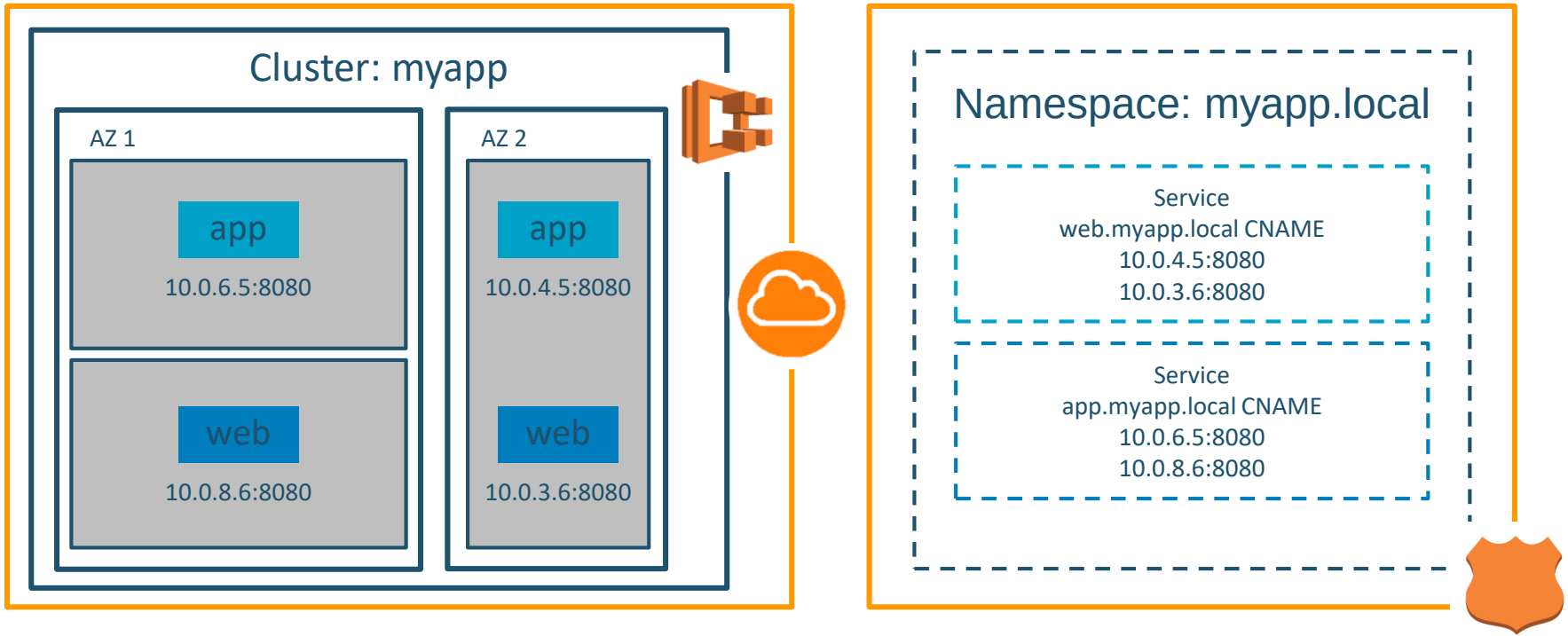
ECS Scheduler updates on:

- Service scaling
  - Task registrations
  - Task de-registrations
- Task health
- Scheduling / Placement changes
- ECS instance changes

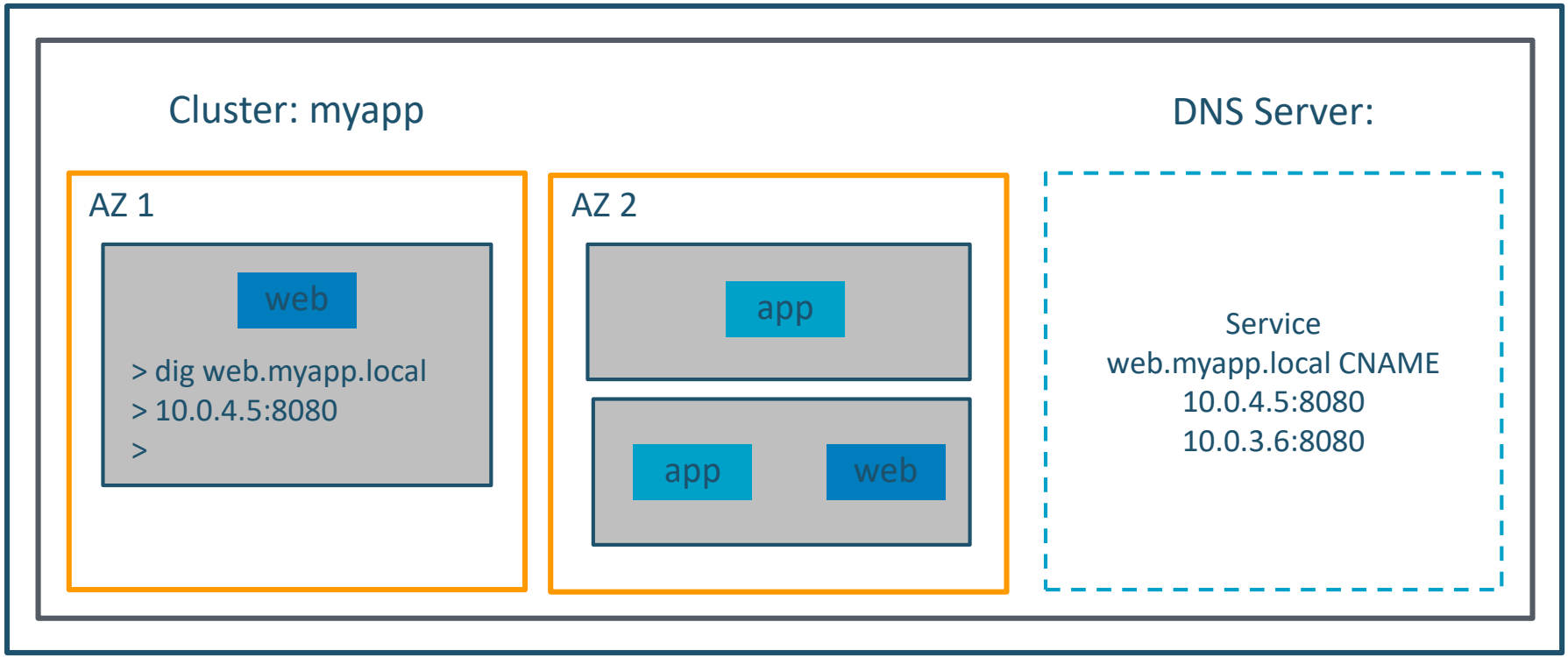
ECS maintains latest state of the dynamic environment in Service Registry

aws

ECS updates service endpoints in Route 53

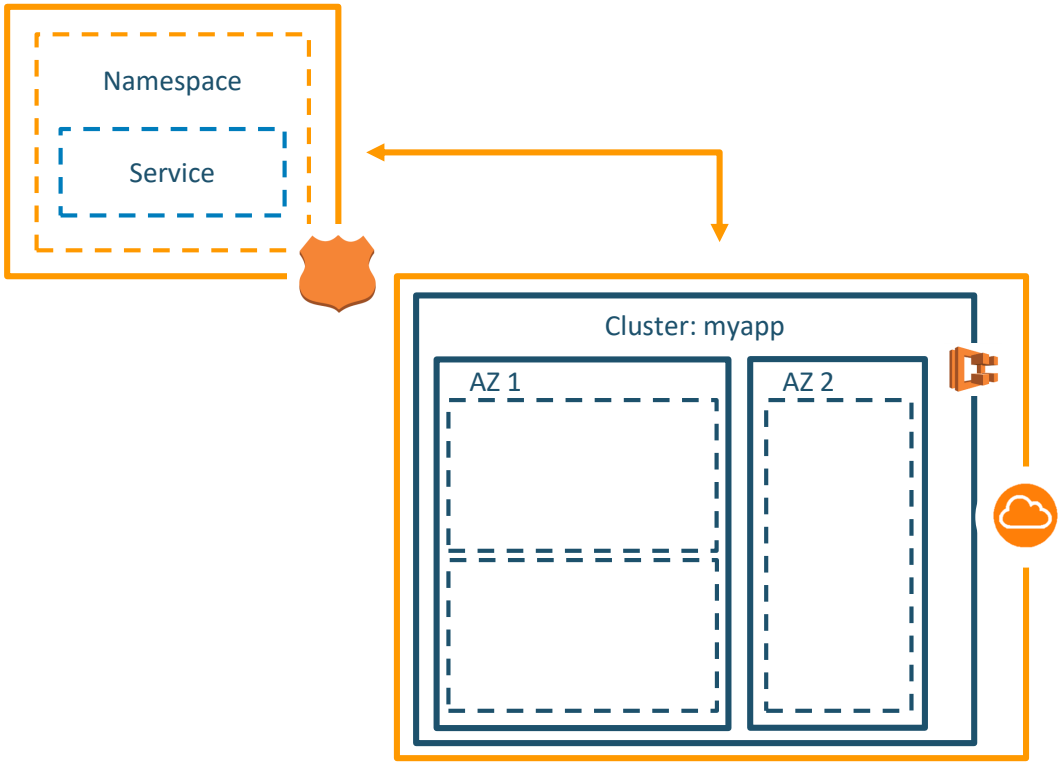


Services connect to latest endpoints via DNS



aws

Benefits of this approach



Managed

- Setup once, use forever

Highly available

- Same availability and scalability as Route 53

Extensible

- Public APIs that can be used across AWS
- Works across clusters, accounts, AZs
- Works across AWS services

aws

Amazon Route 53 Auto Naming

What is Route 53 Auto Naming?

API that powers ECS Service Discovery

Service name registration and management tool

DNS-based service discovery mechanism

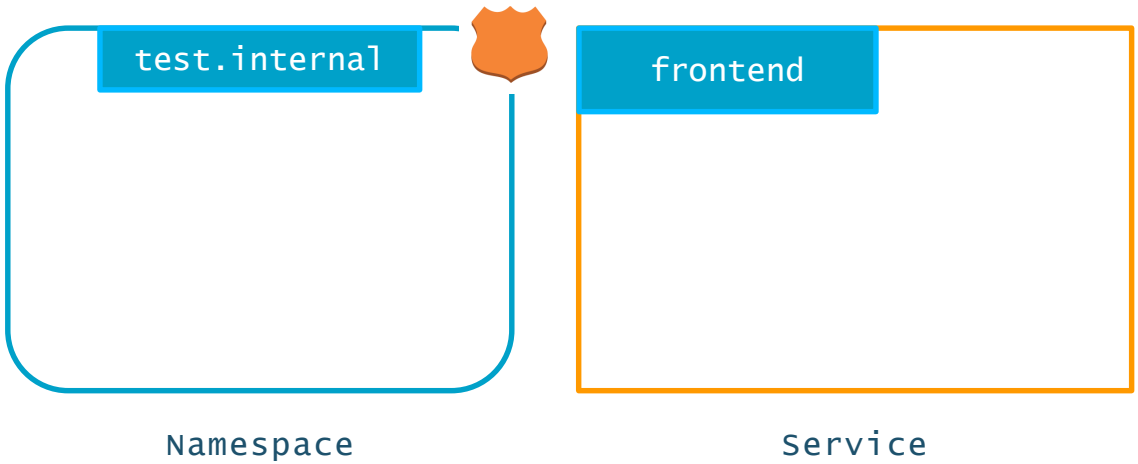
aws

Introducing new abstractions

Namespace

Service

Service Instance

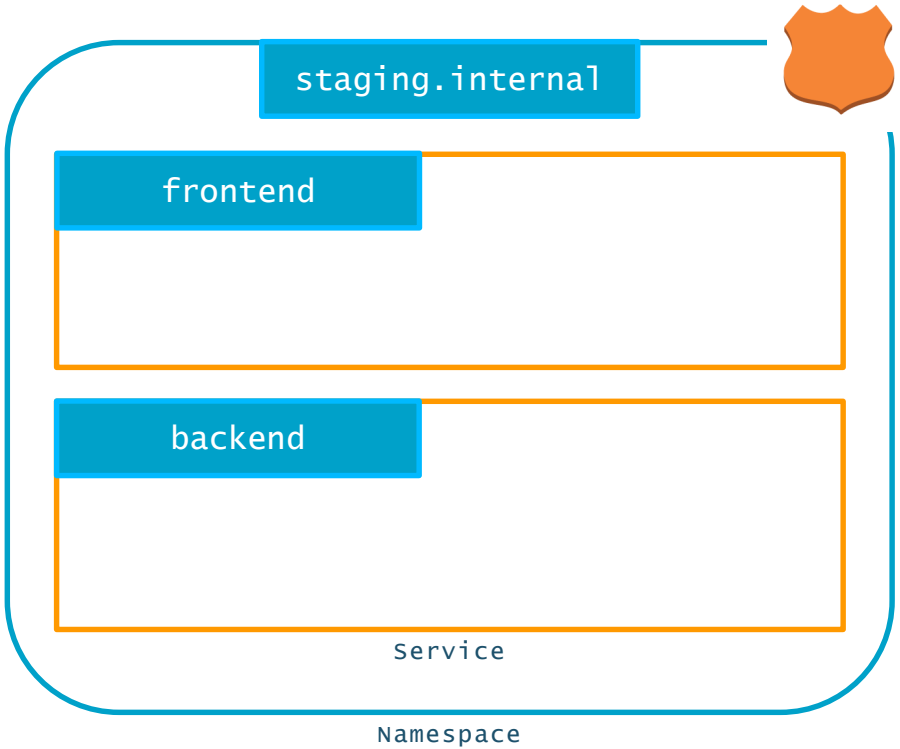


Namespace

Logical group of services

Private or public visibility

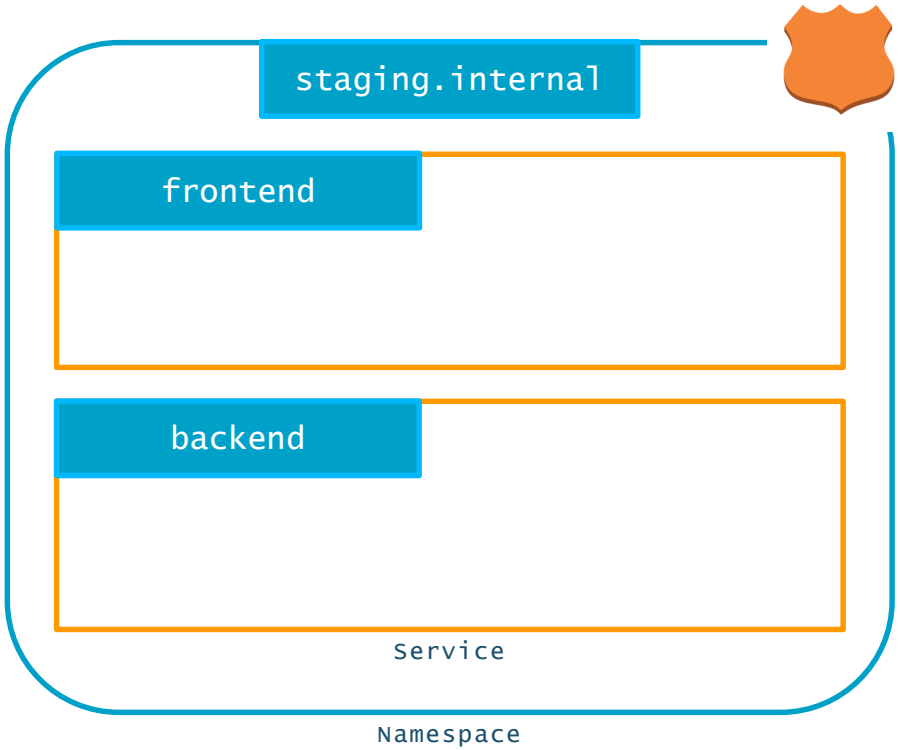
Service-linked hosted zone in Amazon Route 53



Namespace

Namespace management APIs:

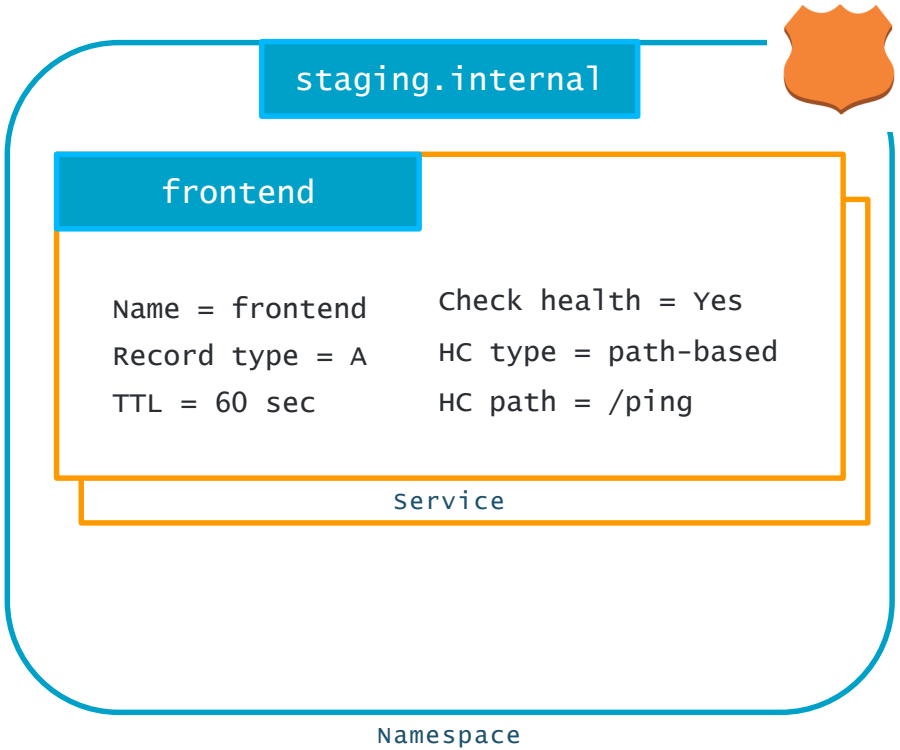
- createPublicDnsNamespace
- createPrivateDnsNamespace
- deleteNamespace
- listNamespaces
- getNamespace



Service

ECS: application component run on one or many tasks

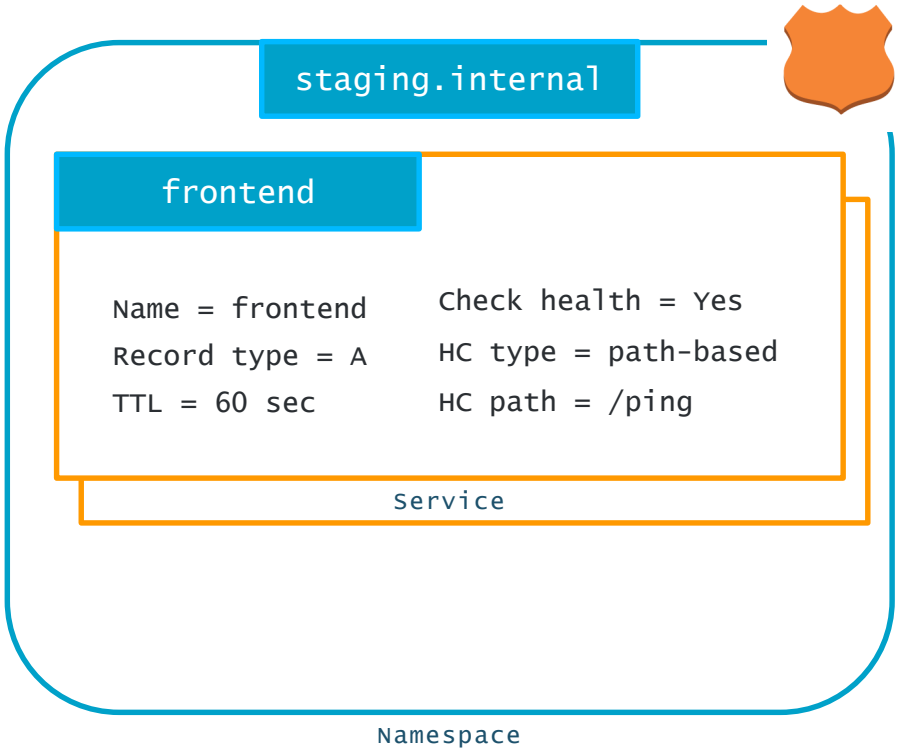
Auto Naming: service naming and health checking template



Service

Service management APIs:

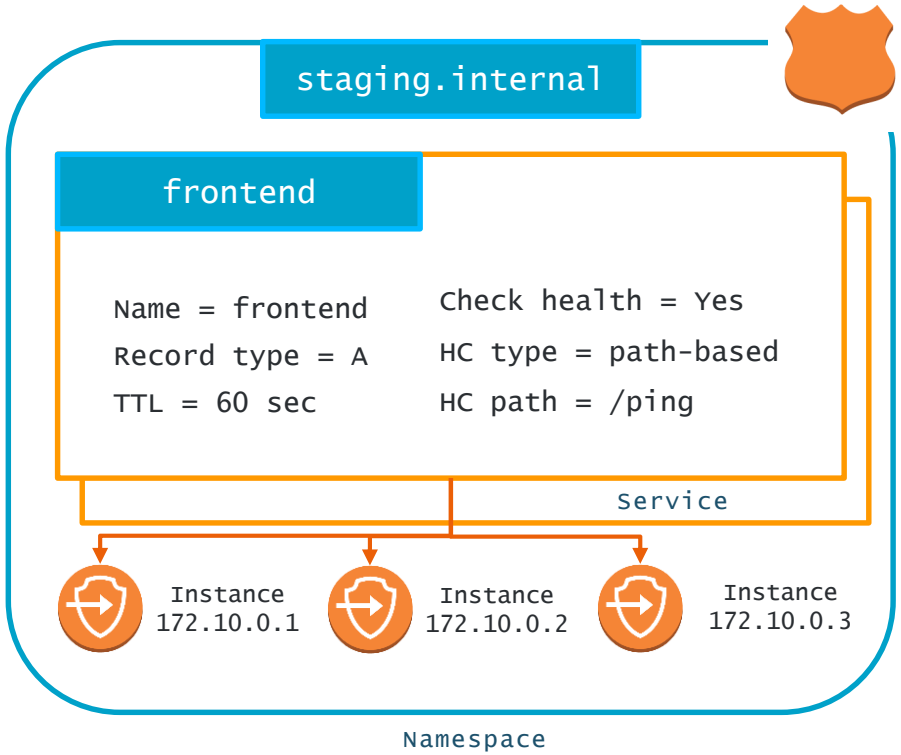
- createService
- deleteService
- updateService
- listServices
- getService



Service Instance

Represents an actual endpoint

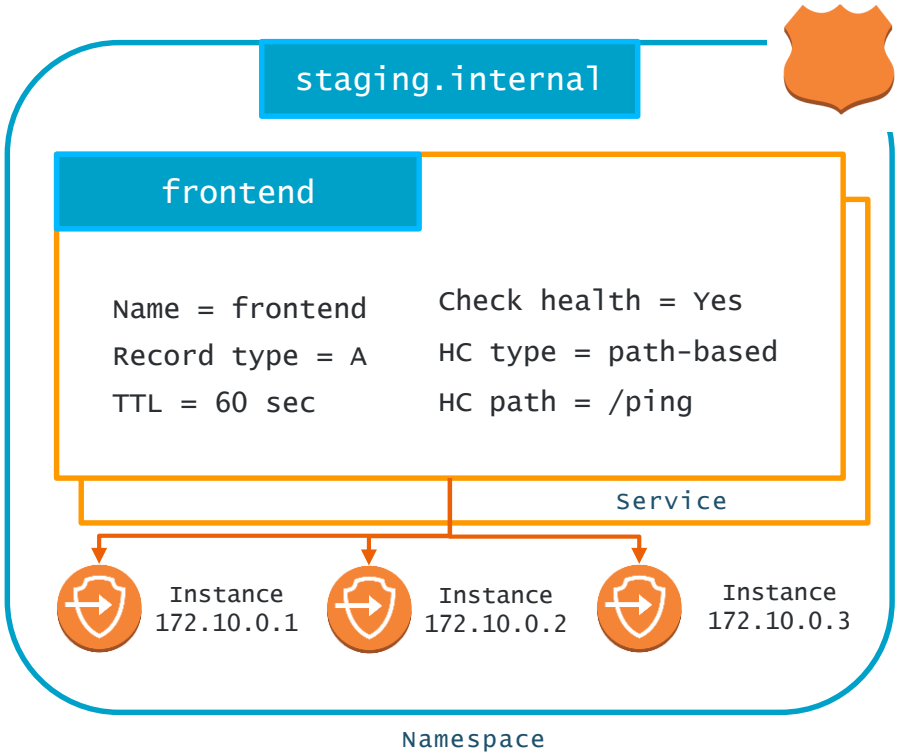
Addressable by *ip* or *ip:port*



Service Instance

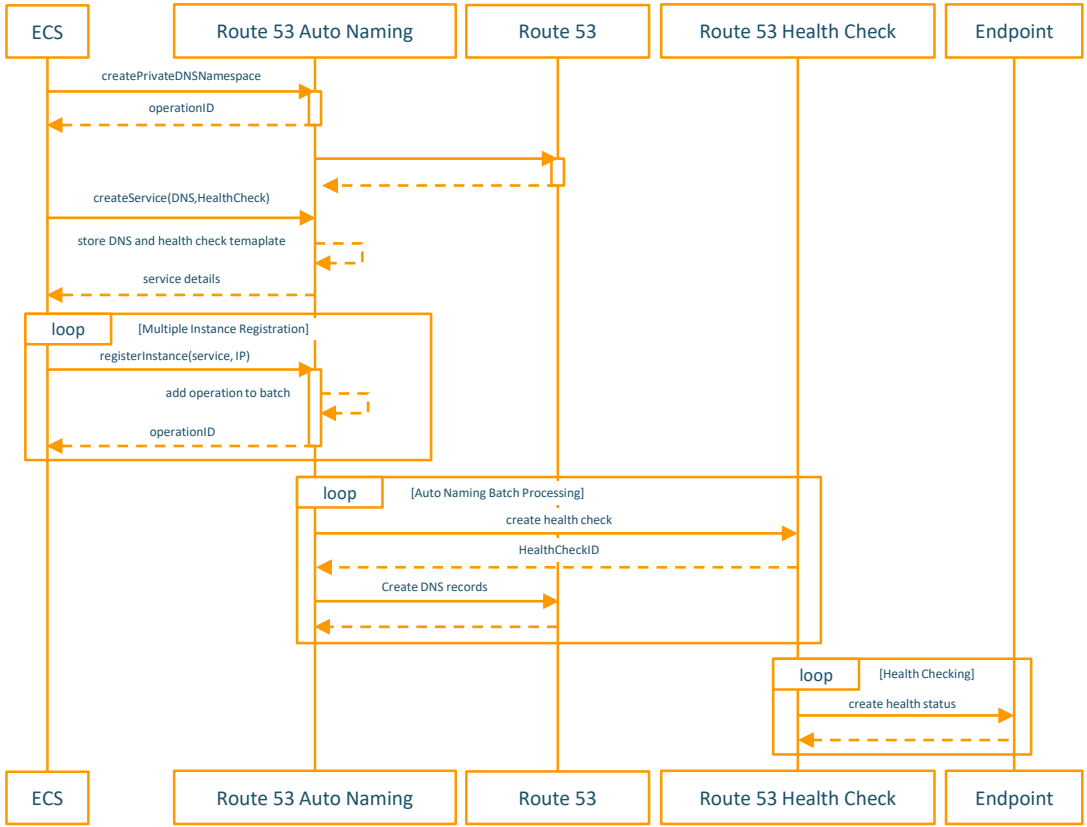
Service Instance management APIs:

- registerInstance
- deregisterInstance
- listInstances
- getInstance
- getInstanceHealthStatus



aws

Route 53 Auto Naming Workflow



Discovery over DNS

Auto Naming uses Route 53 Multivalue Answer Routing

Each DNS query returns up to 8 healthy* endpoints

Client-side load balancing

* If health-checking is enabled for the corresponding Route 53 Auto Naming Service

aws

Amazon Route 53 Auto Naming benefits

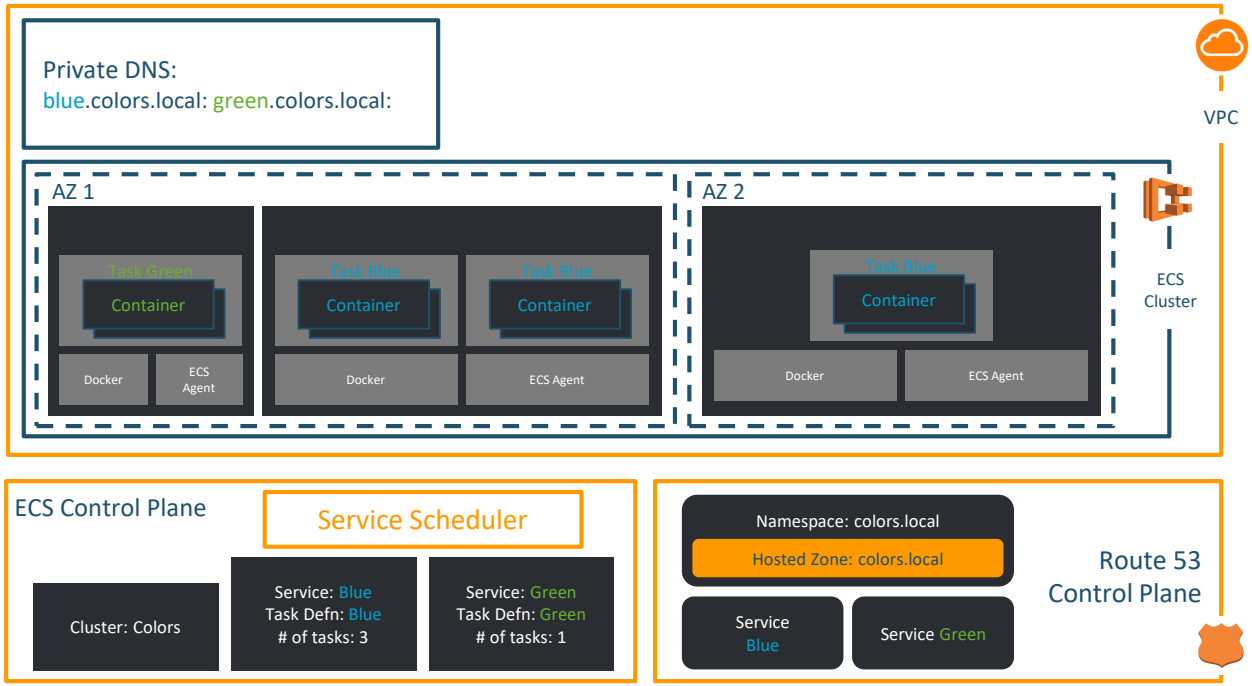
Managed API designed for micro-service architectures

Regional control plane for better availability

Discovery via Route 53 DNS with 100% availability SLA

Under the Hood

Services can discover other services using DNS



Service Green -> Service Blue
(option 1)

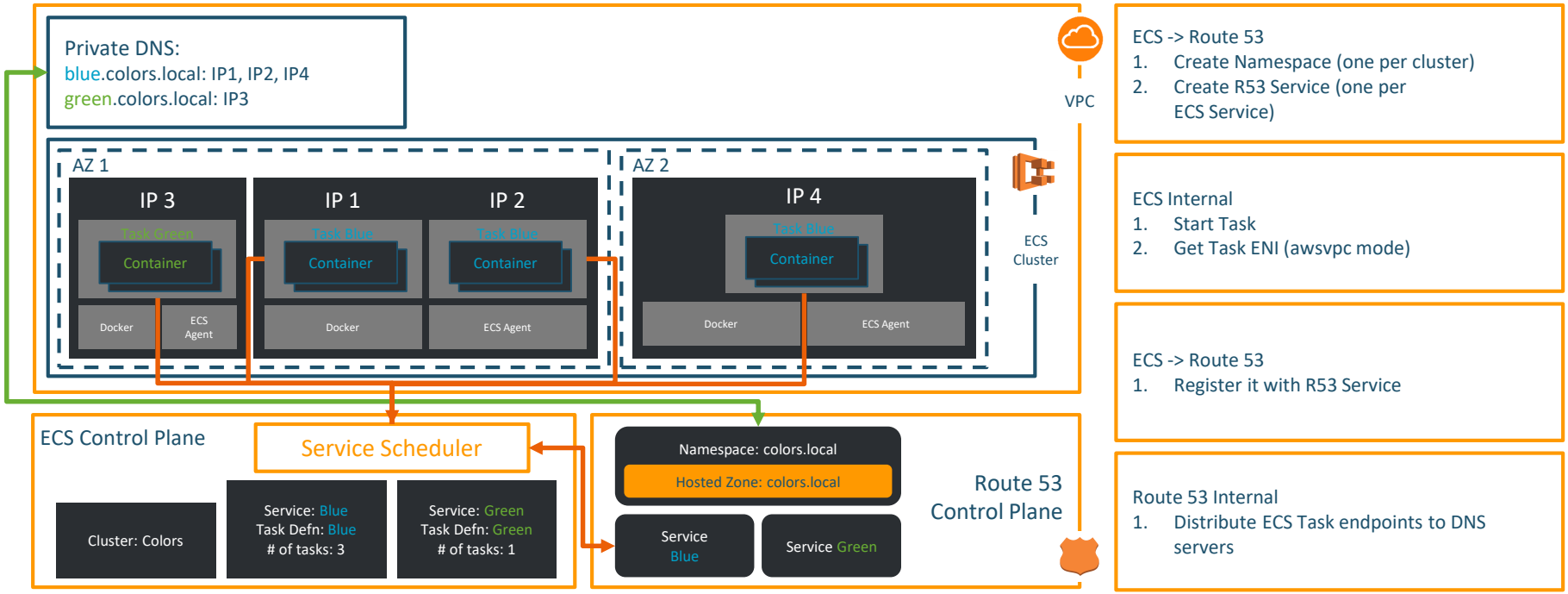
- Green does a DNS query
- Query hits internal vpc DNS server
- Green get a list of IP / IP+Port
- Green communicates with that IP/ IP + Port

(option 2)

- Green calls Service endpoint via an API
- Gets list of all IP/ IP+port for a service

aws

Services can discover other services using DNS



ECS -> Route 53
1. Create Namespace (one per cluster)
2. Create R53 Service (one per ECS Service)

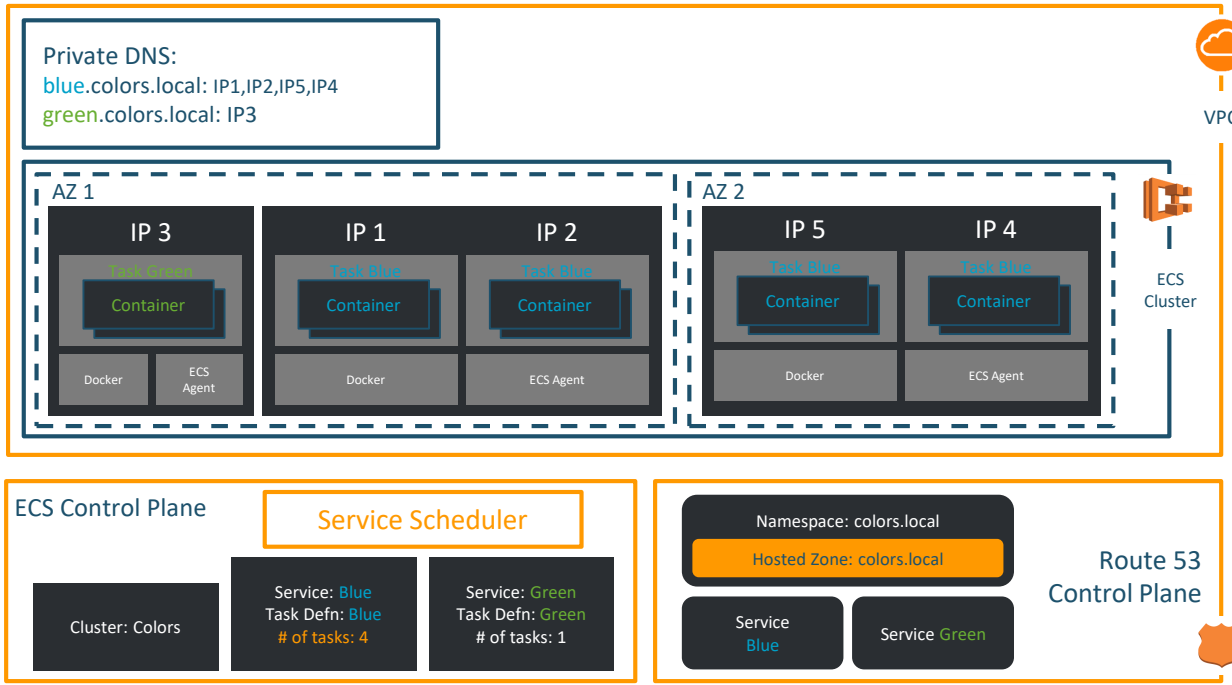
ECS Internal
1. Start Task
2. Get Task ENI (awsvpc mode)

ECS -> Route 53
1. Register it with R53 Service

Route 53 Internal
1. Distribute ECS Task endpoints to DNS servers

aws

Services can discover other services using DNS



ECS –> R53 ongoing

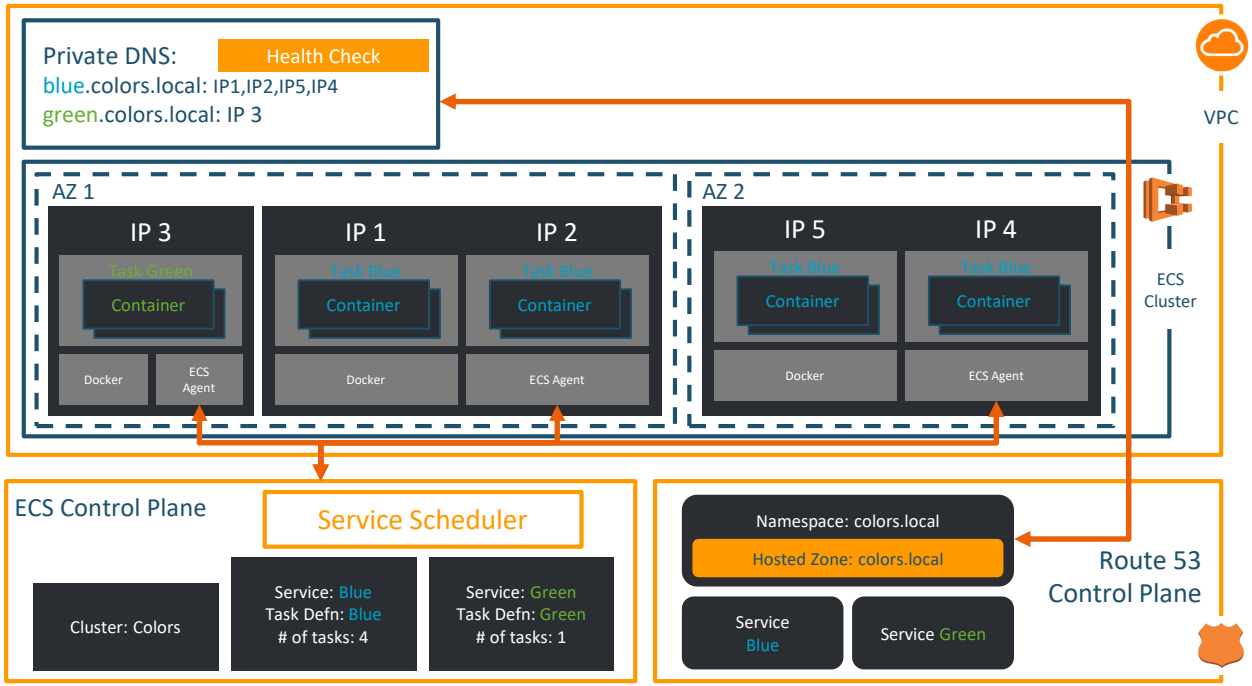
1. Register/ Deregister Tasks into Route 53 Service based on task state and health
2. Updates Tasks in Route 53 Service on cluster auto scaling or service scaling

Route 53 Ongoing

1. For Service query, return IP/Port
2. For Task Health Check details, update DNS servers
3. For ECS Task state changes, update DNS servers

aws

Services can discover other services using DNS



Service Green -> Service Blue (option 1)

- Green does a DNS query
- Query hits internal vpc DNS server
- Green get a list of IP / IP+Port
- Green communicates with that IP/ IP + Port

(option 2)

- Green calls Service endpoint via an API
- Gets list of all IP/ IP+port for a service

aws

Requirements & Limits

- Tasks must use “awsvpc” network mode
- Must specify container health check endpoint for essential containers in task
- Console workflow assumes 1:1:1 mapping between vpc:cluster:app

- Dependent on Route 53 propagation delay for registering new IPs (scaling) or stopping traffic to unhealthy tasks
- Client side logic needed for query, resolution and retries, 8 DNS records fetched in each query
- Private VPC: container health check only, no way to add on network health

aws

How do I use it?

- Native integration with Amazon ECS

  Works with Fargate and EC2 launch types

- Also available for Kubernetes!

  https://github.com/kubernetes-incubator/external-dns

- Available in all AWS regions where Amazon ECS and Route 53 Auto Naming are available

  https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services

aws

Thank You!

Learn more at https://aws.amazon.com/ecs/resources


peckn@amazon.com


nathankpeck


nathanpeck