

ECS Service discovery workshop

In this workshop you're going to manually create and deploy 2 services into **ECS Fargate**, and configure **ECS Service Discovery** using Cloudmap so they can find and talk to each other.

The **backend** polls the service discovery API for information about the available namespaces, services and instances, and renders the result in an HTML page.

The **worker** application is a simple webserver, which responds to a *GET* '/' with a message.

Prerequisites

Clone the project git repository in your **Cloud9** environment.

```
git clone https://github.com/santatamas/ecs-workshop.git
cd ~/environment/ecs-workshop
```

Creating the ECR repositories

First, create 2 docker image repositories for our backend and worker images.

```
aws ecr create-repository --repository-name flask-backend
aws ecr create-repository --repository-name flask-worker
```

Setting up the required environment variables

You need to export a few environment variables for later use.

```
export BACKEND_REPO=$(aws ecr describe-repositories | \
jq -r .repositories[0].repositoryUri)
export WORKER_REPO=$(aws ecr describe-repositories | \
jq -r .repositories[1].repositoryUri)
```

Building the container images

Next, you'll build the docker images for the worker, and backend applications.

```
cd backend; docker build . -t flask-backend:latest; cd ..  
cd workers; docker build . -t flask-worker:latest; cd ..
```

Tagging & pushing the images to ECR

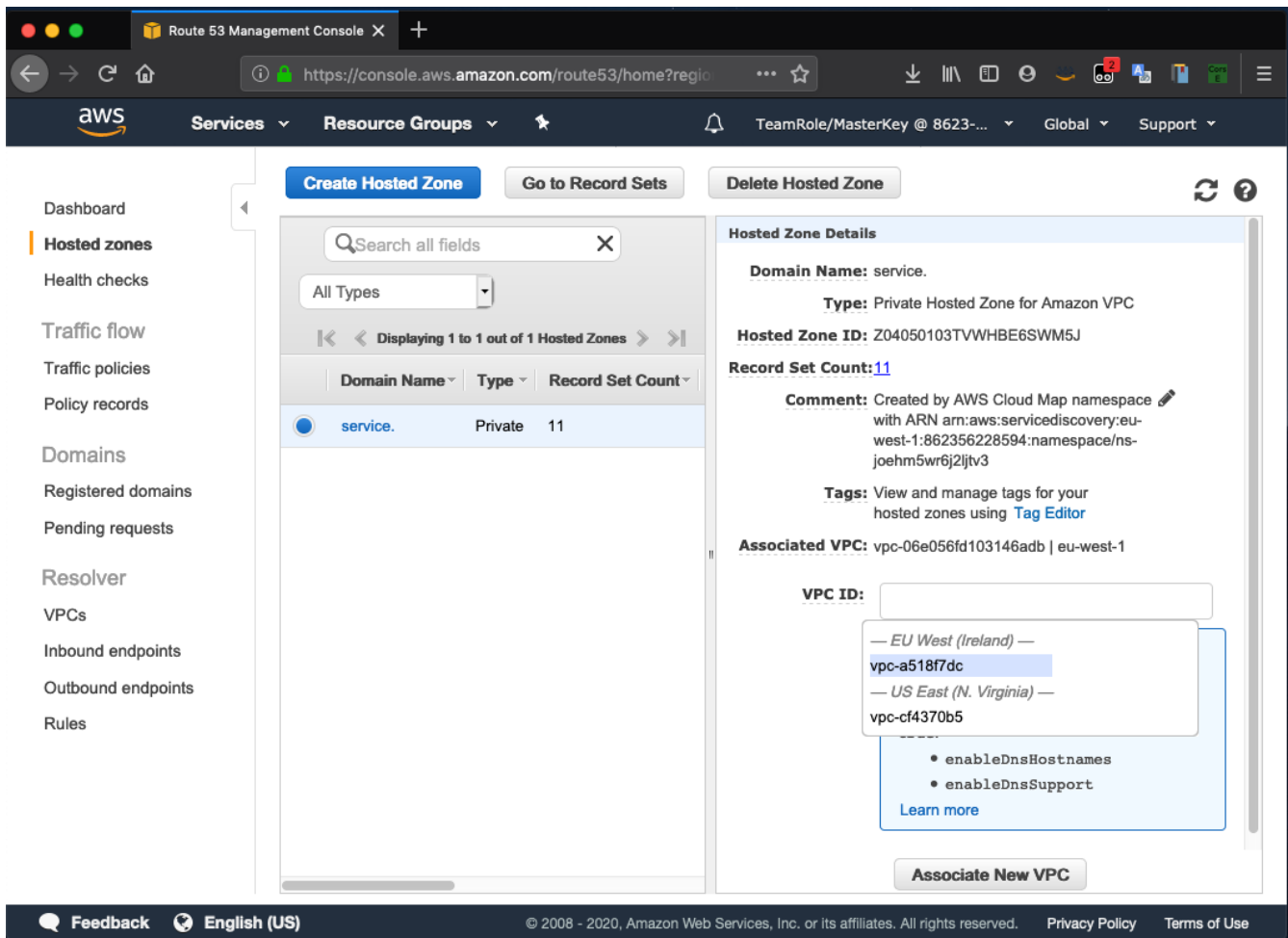
As a final step, let's tag the images, and push them to the ECR repositories you just created.

```
$(aws ecr get-login --no-include-email --region eu-west-1)  
docker tag flask-backend:latest ${BACKEND_REPO}:latest  
docker push ${BACKEND_REPO}:latest  
docker tag flask-worker:latest ${WORKER_REPO}:latest  
docker push ${WORKER_REPO}:latest
```

[OPTIONAL] Running the backend application from Cloud9

It's possible to run the **backend** application straight from your Cloud9 environment. By default, the hosted zone in Route53 is not associated with your Cloud9 VPC, hence the private DNS name resolution for your services will fail.

To fix this, open your AWS Management Console, and navigate to Route53. Select your hosted zone (should be called **service.**), and from the VPC ID dropdown on the right hand side, select the VPC of your Cloud9 environment.



Finally, click on *Associate New VPC*.

Export your AWS credentials to environment variables

```
export AWS_ACCESS_KEY_ID=$(aws configure get default.aws_access_key_id)
export AWS_SECRET_ACCESS_KEY=$(aws configure get default.aws_secret_access_key)
export AWS_SESSION_TOKEN=$(aws configure get default.aws_session_token)
```

Run the following command in your Cloud9 Terminal to run the **backend** docker image locally:

```
docker run -p 80:80 \
-e AWS_ACCESS_KEY_ID=${AWS_ACCESS_KEY_ID} \
-e AWS_SECRET_ACCESS_KEY=${AWS_SECRET_ACCESS_KEY} \
-e AWS_DEFAULT_REGION=eu-west-1 \
-e AWS_SESSION_TOKEN=${AWS_SESSION_TOKEN} \
-d flask-backend:latest
```

Now you should be able to access the website within your Cloud9 environment, without having to deploy it to ECS.

```
curl localhost
```

Allowing the ECS Task execution role to access the Service Discovery API

Our backend application will access the Service Discovery API to retrieve a list of namespaces, services and instances. The Task Execution role we created during the previous workshop does not provide access to this API, so we'll have to extend it.

- Open IAM, and select the Task Execution role starting with **container-demo-ECSTaskExecutionRole**

The screenshot shows the AWS IAM console interface. On the left is a navigation sidebar with sections: 'Identity and Access Management (IAM)' containing 'Dashboard', 'Access management' (Groups, Users, Roles, Policies, Identity providers, Account settings), 'Access reports' (Access analyzer, Archive rules, Analyzer details, Credential report, Organization activity, Service control policies (SCPs)), and a search bar 'Search IAM'. Below the sidebar, it displays 'AWS account ID: 862356228594'. The main content area is titled 'Roles > container-demo-ECSTaskExecutionRole-1MTLBV7W7QUJT' and 'Summary'. It includes a 'Delete role' button and a list of role details: Role ARN, Role description (with an 'Edit' link), Instance Profile ARNs, Path, Creation time, Last activity, and Maximum CLI/API session duration. Below this is a tabbed interface with 'Permissions' selected. It shows 'Permissions policies (1 policy applied)' with an 'Attach policies' button and an 'Add inline policy' link. A table lists the policy 'AmazonECSTaskExecutionRole' as an 'Inline policy'. Below the table are buttons for 'Policy summary', '{} JSON', 'Edit policy', and 'Simulate policy'. A 'Filter' input is present above a table with columns 'Service', 'Access level', and 'Resource'. The table shows two entries: 'CloudWatch Logs' with 'Limited: Write' access to 'All resources', and 'Elastic Container Registry' with 'Limited: Read' access to 'All resources'. At the bottom, it indicates 'Permissions boundary (not set)'.

Identity and Access Management (IAM)

Dashboard

Access management

- Groups
- Users
- Roles**
- Policies
- Identity providers
- Account settings

Access reports

- Access analyzer
- Archive rules
- Analyzer details
- Credential report
- Organization activity
- Service control policies (SCPs)

Search IAM

AWS account ID: 862356228594

Roles > container-demo-ECSTaskExecutionRole-1MTLBV7W7QUJT

Summary

Delete role

Role ARN: arn:aws:iam::862356228594:role/container-demo-ECSTaskExecutionRole-1MTLBV7W7QUJT

Role description: [Edit](#)

Instance Profile ARNs: [Add](#)

Path: /

Creation time: 2020-02-21 10:02 UTC+0100

Last activity: 2020-02-21 13:36 UTC+0100 (Today)

Maximum CLI/API session duration: 1 hour [Edit](#)

Permissions

Trust relationships

Tags

Access Advisor

Revoke sessions

Permissions policies (1 policy applied)

Attach policies

Add inline policy

Policy name	Policy type
AmazonECSTaskExecutionRole	Inline policy

Policy summary

{ } JSON

Edit policy

Simulate policy

Filter

Service	Access level	Resource
CloudWatch Logs	Limited: Write	All resources
Elastic Container Registry	Limited: Read	All resources

Allow (2 of 222 services) [Show remaining 220](#)

Permissions boundary (not set)

- Select the existing policy, and update it with the following JSON

```
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "VisualEditor0",
    "Effect": "Allow",
    "Action": [
      "servicediscovery:*",
      "logs:CreateLogStream",
      "ecr:GetDownloadUrlForLayer",
      "ecr:BatchGetImage",
      "ecr:GetAuthorizationToken",
      "logs:PutLogEvents",
      "ecr:BatchCheckLayerAvailability"
    ],
    "Resource": "*"
  }
]
}

```

Creating a new ECS Task Definition for the backend

- Open your AWS Management Console, and navigate to Amazon ECS.
- Create a new Task Definition

The screenshot shows the AWS Management Console interface for Amazon ECS. The left sidebar contains navigation links for Amazon ECS, Amazon EKS, Amazon ECR, AWS Marketplace, and Subscriptions. The main content area is titled 'Task Definitions' and includes a description: 'Task definitions specify the container information for your application, such as how many containers are part of your task, what resources they will use, how they are linked together, and which host ports they will use. [Learn more](#)'. Below the description, there are buttons for 'Create new Task Definition' (highlighted with a red circle), 'Create new revision', and 'Actions'. A status filter shows 'ACTIVE' selected. A table lists existing task definitions:

Task Definition	Latest revision status
ecs-cwagent-daemon-service	ACTIVE
ecsdemo-crystal	ACTIVE
ecsdemo-frontend	ACTIVE
ecsdemo-nodejs	ACTIVE

The footer of the console shows 'Feedback', 'English (US)', and copyright information: '© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use'.

- Select **FARGATE**, and click *Next step*

- Next, name your definition *backend*, and re-use the task and execution roles from your previous workshop. For task memory and CPU, use 1GB and 0.5 vCPU.

Configure task and container definitions

A task definition specifies which containers are included in your task and how they interact with each other. You can also specify data volumes for your containers to use. [Learn more](#)

Task Definition Name* ⓘ

Requires Compatibilities* FARGATE

Task Role ⓘ

Optional IAM role that tasks can use to make API requests to authorized AWS services. Create an Amazon Elastic Container Service Task Role in the [IAM Console](#) ⓘ

Network Mode ⓘ

If you choose <default>, ECS will start your container using Docker's default networking mode, which is Bridge on Linux and NAT on Windows. <default> is the only supported mode on Windows.

Task execution IAM role

This role is required by tasks to pull container images and publish container logs to Amazon CloudWatch on your behalf. If you do not have the `ecsTaskExecutionRole` already, we can create one for you.

Task execution role ⓘ

Task size ⓘ

The task size allows you to specify a fixed size for your task. Task size is required for tasks using the Fargate launch type and is optional for the EC2 launch type. Container level memory settings are optional when task size is set. Task size is not supported for Windows containers.

Task memory (GB)

The valid memory range for 0.5 vCPU is: 1GB - 4GB.

Task CPU (vCPU)

The valid CPU range for 1GB memory is: 0.25 vCPU - 0.5 vCPU.

Task memory maximum allocation for container memory reservation



Task CPU maximum allocation for containers



- Add a new container definition, and use the URI of the **backend** docker image from your ECR repository.

Add container

Standard

Container name*

backend

Image*

862356228594.dkr.ecr.eu-west-1.amazonaws.com/flask-backend

Private repository authentication*

☐

Memory Limits (MiB)

Soft limit

128

+ Add Hard limit

Define hard and/or soft memory limits in MiB for your container. Hard and soft limits correspond to the ``memory`` and ``memoryReservation`` parameters, respectively, in task definitions.
ECS recommends 300-500 MiB as a starting point for web applications.

Port mappings

Container port

Protocol

tcp

+ Add port mapping

Host port mappings are not valid when the network mode for a task definition is host or awsvpc. To specify different host and container port mappings, choose the Bridge network mode.

Advanced container configuration

- Finally, click "Create" to create the Task definition.

Creating a new ECS Service for the backend

- Open Amazon ECS, and display the Clusters

Amazon ECS

- Clusters**
- Task Definitions
- Account Settings

Amazon EKS

- Clusters

Amazon ECR

- Repositories

AWS Marketplace

- Discover software
- Subscriptions

Clusters > container-demo-ECSCluster-18TZ373QSB652

Cluster : container-demo-ECSCluster-18TZ373QSB652

Get a detailed view of the resources on your cluster. [Update Cluster](#) [Delete Cluster](#)

Status **ACTIVE**

Registered container instances 0

Pending tasks count 0 Fargate, 0 EC2

Running tasks count 9 Fargate, 0 EC2

Active service count 3 Fargate, 1 EC2

Draining service count 0 Fargate, 0 EC2

Services | Tasks | ECS Instances | Metrics | Scheduled Tasks | Tags | Capacity Providers

[Create](#) [Update](#) [Delete](#) [Actions](#) Last updated on February 21, 2020 1:45:25 PM (3m ago)

Filter in this page Launch type ALL Service type ALL

	Service Name	Status	Servi...	Task ...	Desir...	Run...	Laun...	P
<input type="checkbox"/>	ecsdemo-nodejs	ACTIVE	REPL...	ecsd...	3	3	FAR...	L
<input type="checkbox"/>	cwagent-daemon-service	ACTIVE	DAE...	ecs-c...	0	0	EC2	--
<input type="checkbox"/>	ecsdemo-crystal	ACTIVE	REPL...	ecsd...	3	3	FAR...	L
<input type="checkbox"/>	ecsdemo-frontend	ACTIVE	REPL...	ecsd...	3	3	FAR...	L

- Click *Create* to create a new Service
- Use the values shown below, then click *Next step*

Configure service

A service lets you specify how many copies of your task definition to run and maintain in a cluster. You can optionally use an Elastic Load Balancing load balancer to distribute incoming traffic to containers in your service. Amazon ECS maintains that number of tasks and coordinates task scheduling with the load balancer. You can also optionally use Service Auto Scaling to adjust the number of tasks in your service.

Launch type ☒ FARGATE ☐ EC2 [?](#)

Task Definition Family

backend [Enter a value](#)

Revision

1 (latest)

Platform version LATEST [?](#)

Container version

Cluster

Service name

Service type* REPLICA

Number of tasks

Minimum healthy percent

Maximum percent

Deployments

Choose a deployment option for the service.

- Deployment type*
- ☒ Rolling update
- ☐ Blue/green deployment (powered by AWS CodeDeploy)
- This sets AWS CodeDeploy as the deployment controller for the service. A CodeDeploy application and deployment group are created automatically with [default settings](#) for the service. To change to the rolling update deployment type after the service has been created, you must re-create the service and select the "rolling update" deployment type.

i Tagging requires that you opt in to the new ARN and resource ID format. The IAM user/role has not opted in to the new ARN format. Opt-in to the new format to use this feature. [Manage your opt-in settings.](#)

*Required

Cancel

Next step

- Use the existing VPC and subnet for the service. Make sure to select a public subnet, you'll have to be able to access the container from your browser.

Configure network

VPC and security groups

VPC and security groups are configurable when your task definition uses the awsvpc network mode.

Cluster VPC* vpc-06e056fd103146adb ... ⓘ

Subnets* ⓘ

subnet-0e02a7d77c1ead58e ✕
(10.0.0.0/24) - eu-west-1a
assign ipv6 on creation: Disabled

Security groups* backen-5404 **Edit** ⓘ

Auto-assign public IP ENABLED ⓘ

Health check grace period

If your service's tasks take a while to start and respond to ELB health checks, you can specify a health check grace period of up to 2,147,483,647 seconds during which the ECS service scheduler will ignore ELB health check status. This grace period can prevent the ECS service scheduler from marking tasks as unhealthy and stopping them before they have time to come up. This is only valid if your service is configured to use a load balancer.

Health check grace period requires a load balancer. ⓘ

- Use the existing Service Discovery namespace (created during the previous workshop)

Service discovery (optional)

Service discovery uses Amazon Route 53 to create a namespace for your service, which allows it to be discoverable via DNS.

Enable service discovery integration ☒

Namespace* service | PRIVATE ⓘ

Configure service discovery service

☒ Create new service discovery service

☐ Select existing service discovery service

Service discovery name*

backend



Alphanumeric and underscores strings, with periods in the middle are valid. For more information, see [Route 53 Auto Naming documentation](#).

Enable ECS task health propagation

Enabling this field allows ECS to communicate the task state to Route 53 and reduce the amount of time it takes for unhealthy tasks to be removed from DNS.

Docker health checks

Docker health checks are defined in your task definition. They allow multiple health checks for essential, aggregated containers to determine the health of your service.

Enable public DNS health check

DNS health checks in a private namespace are not currently supported.

DNS records for service discovery

DNS record type*

A

**TTL***

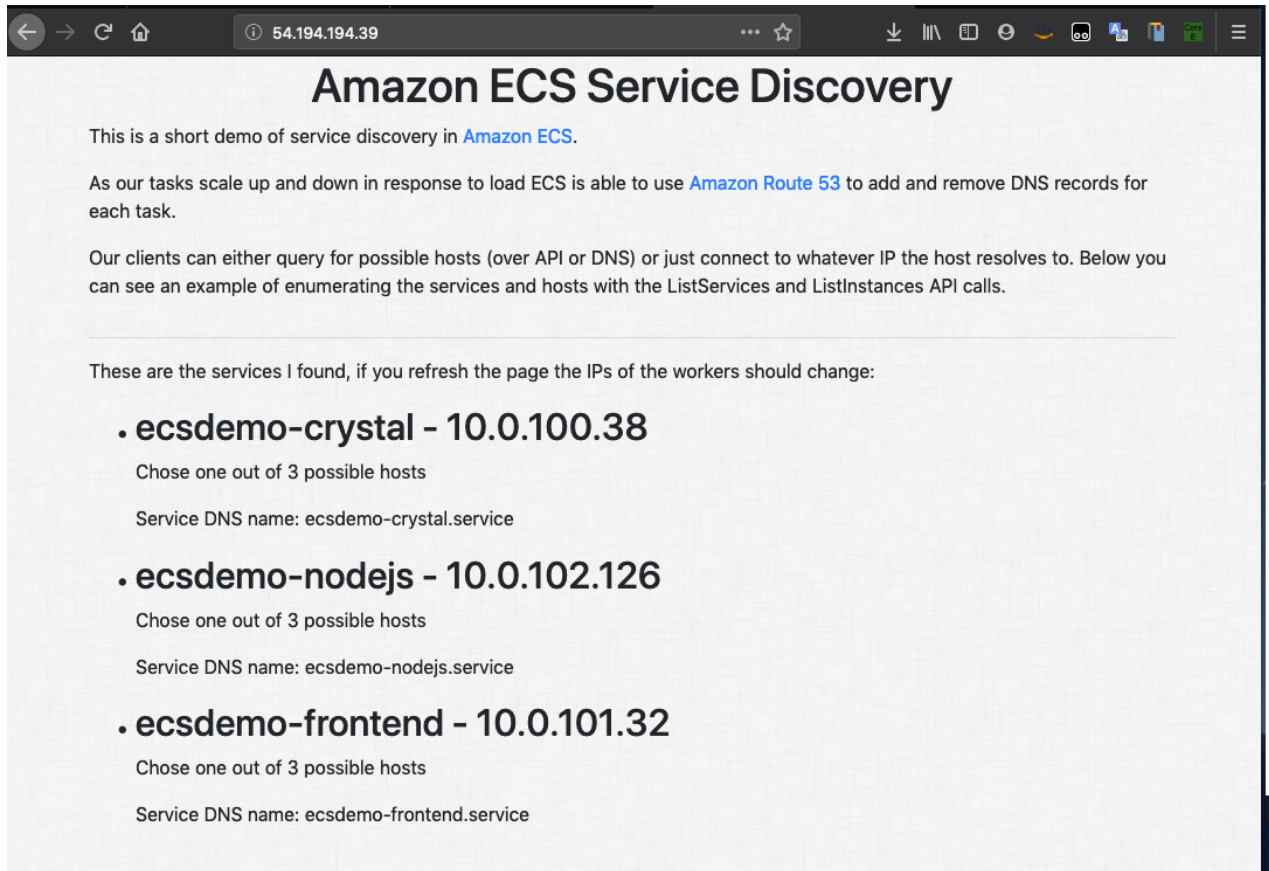
60

second(s)

[+ Add DNS record](#)

- Finish the setup, and create the service (Next->Next->Finish)

- Open the public IP of the container, and you should see the running application



Creating a new ECS Task Definition for the worker nodes

- Create a new Task Definition using the previous steps. Use the name "worker", and use the URI of the **worker** docker image from your ECR repository.

Creating a new ECS Service for the workers

- Create a new service
- Use the settings shown below. Please note that we specified 3 running instances for this service, to test load balancing.

Configure service

A service lets you specify how many copies of your task definition to run and maintain in a cluster. You can optionally use an Elastic Load Balancing load balancer to distribute incoming traffic to containers in your service. Amazon ECS maintains that number of tasks and coordinates task scheduling with the load balancer. You can also optionally use Service Auto Scaling to adjust the number of tasks in your service.

Launch type	<input checked="" type="radio"/> FARGATE <input type="radio"/> EC2 ⓘ
Task Definition	<div>Family <div>worker ▼</div><div>Revision 1 (latest) ▼</div></div> <div>Enter a value</div>
Platform version	<div>LATEST ▼ ⓘ</div>
Cluster	<div>container-demo-ECSClus... ▼ ⓘ</div>
Service name	<div>worker ⓘ</div>
Service type*	<div>REPLICA ⓘ</div>
Number of tasks	<div>3 ⓘ</div>
Minimum healthy percent	<div>100 ⓘ</div>
Maximum percent	<div>200 ⓘ</div>



- Use the VPC settings shown below. This time, try to specify a different subnet for the worker instances.

Configure network

VPC and security groups

VPC and security groups are configurable when your task definition uses the awsvpc network mode.

Cluster VPC* vpc-06e056fd103146adb ... 

Subnets* subnet-02811583cc53c7a5b  
(10.0.102.0/24) - eu-west-1c
assign ipv6 on creation: Disabled

Security groups* worker-2269  

Auto-assign public IP ENABLED 

Health check grace period

If your service's tasks take a while to start and respond to ELB health checks, you can specify a health check grace period of up to 2,147,483,647 seconds during which the ECS service scheduler will ignore ELB health check status. This grace period can prevent the ECS service scheduler from marking tasks as unhealthy and stopping them before they have time to come up. This is only valid if your service is configured to use a load balancer.

Health check grace period requires a load balancer. 

- Use the existing Service Discovery Namespace (created during the previous workshop)

Service discovery (optional)

Service discovery uses Amazon Route 53 to create a namespace for your service, which allows it to be discoverable via DNS.

Enable service discovery ☒
integration

Namespace* service | PRIVATE 

Configure service discovery ☒ Create new service discovery
service ☐ Select existing service

discovery service

Service discovery name*

worker



Alphanumeric and underscores strings, with periods in the middle are valid. For more information, see [Route 53 Auto Naming documentation](#).

Enable ECS task health
propagation



Enabling this field allows ECS to communicate the task state to Route 53 and reduce the amount of time it takes for unhealthy tasks to be removed from DNS.

Docker health checks

Docker health checks are defined in your task definition. They allow multiple health checks for essential, aggregated containers to determine the health of your service.

Enable public DNS health
check

DNS health checks in a private namespace are not currently supported.

DNS records for service discovery

DNS record type*

A



TTL*

60

second(s)



[+ Add DNS record](#)

- Refresh the **backend** application in your browser to see the newly discovered worker service. The application also sends an HTTP request to the service, and displays the

response.

Amazon ECS Service Discovery

This is a short demo of service discovery in [Amazon ECS](#).

As our tasks scale up and down in response to load ECS is able to use [Amazon Route 53](#) to add and remove DNS records for each task.

Our clients can either query for possible hosts (over API or DNS) or just connect to whatever IP the host resolves to. Below you can see an example of enumerating the services and hosts with the ListServices and ListInstances API calls.

These are the services I found, if you refresh the page the IPs of the workers should change:

- **ecsdemo-crystal - 10.0.102.212**
Chose one out of 3 possible hosts
Service DNS name: ecsdemo-crystal.service
- **ecsdemo-nodejs - 10.0.101.74**
Chose one out of 3 possible hosts
Service DNS name: ecsdemo-nodejs.service
- **worker - 10.0.102.112**
Service says: Hello from the worker service!
Chose one out of 3 possible hosts
Service DNS name: worker.service
- **ecsdemo-frontend - 10.0.102.47**
Chose one out of 3 possible hosts
Service DNS name: ecsdemo-frontend.service

Summary

In this exercise you:

- Set up ECR repositories, built and pushed docker images
- Created ECS Fargate Task Definitions for the **backend** and **worker** applications
- Deployed the Task Definitions to services
- Used service discovery to communicate with the deployed services