Demo 3: Docker voting application on ECS Fargate

Introduction

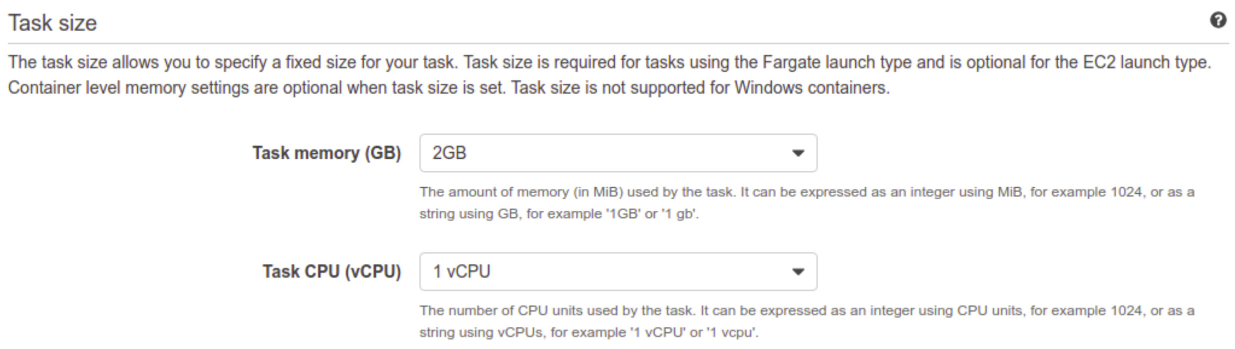
Following are the learning objectives of this demonstration:

- Launch docker voting application on ECS cluster in Fargate mode

Use following docker images:

Vote: dipesh017/demo:vote
Redis: redis:5.0-alpine3.10
Worker: dipesh017/demo:worker
Db: postgres:9.4
POSTGRES_USER : postgres
POSTGRES_PASSWORD : postgres
Result: dipesh017/demo:result

1. Launch a Fargate cluster with the name "demo.TCP"
2. Create 5 task definitions (one for each service)
    a. Mention the name for each task definition
    b. Mention CPU/Memory (1 core CPU, 2 GB RAM)



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.

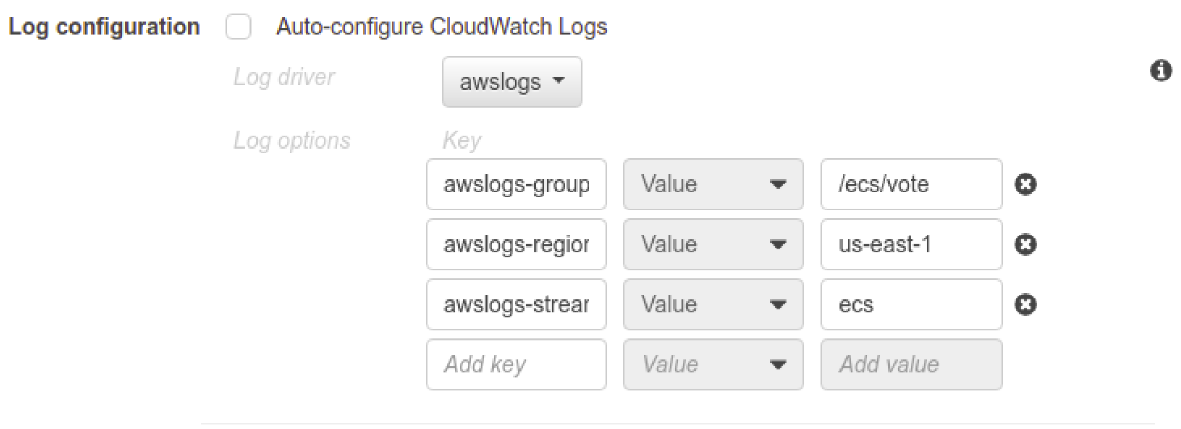
Task memory (GB): 2GB

The amount of memory (in MiB) used by the task. It can be expressed as an integer using MiB, for example 1024, or as a string using GB, for example '1GB' or '1 gb'.

Task CPU (vCPU): 1 vCPU

The number of CPU units used by the task. It can be expressed as an integer using CPU units, for example 1024, or as a string using vCPUs, for example '1 vCPU' or '1 vcpu'.

    c. Mention logging



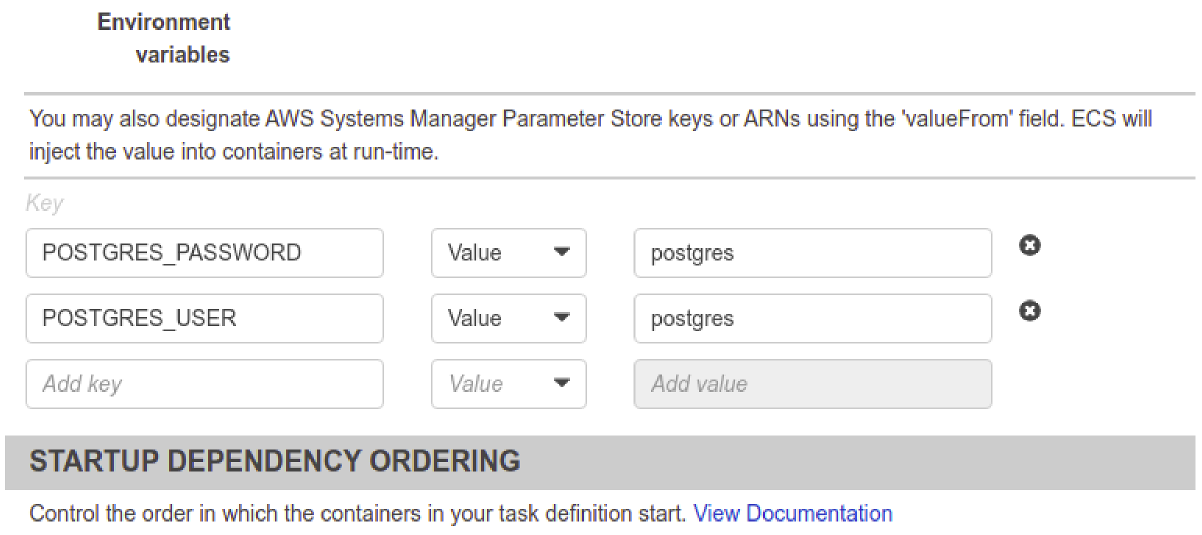
Log configuration ☐ Auto-configure CloudWatch Logs

Log driver: awslogs

Log options

Key		
awslogs-group	Value	/ecs/vote
awslogs-regior	Value	us-east-1
awslogs-strear	Value	ecs
Add key	Value	Add value

d. Mention environment variables in db Task definition



Environment variables

You may also designate AWS Systems Manager Parameter Store keys or ARNs using the 'valueFrom' field. ECS will inject the value into containers at run-time.

Key		
POSTGRES_PASSWORD	Value	postgres
POSTGRES_USER	Value	postgres
Add key	Value	Add value

STARTUP DEPENDENCY ORDERING

Control the order in which the containers in your task definition start. View Documentation

3. Create services in the demo cluster for each component.
   a. Mention replica 1 for each
   b. Create a Security group, allowing all tcp traffic from 0.0.0.0/0
   c. Create service discovery

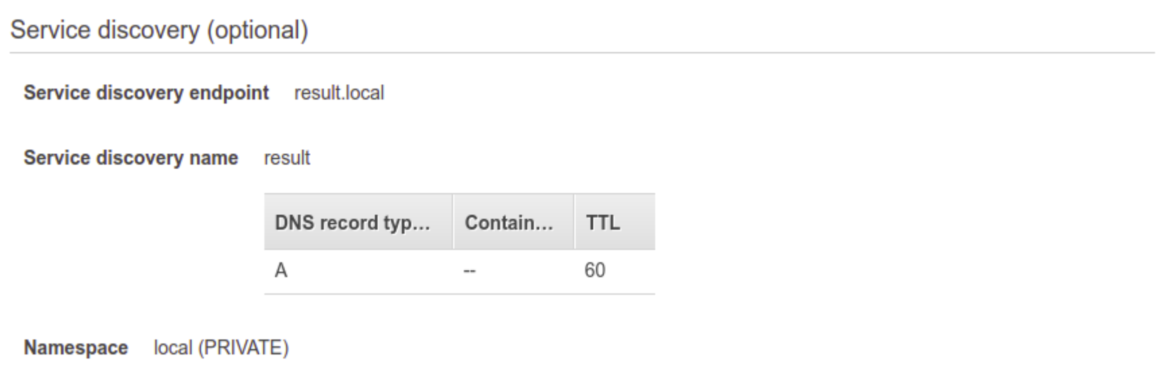
      Vote: vote.local

      Redis: redis.local

      Worker: worker.local

      Db: db.local

      Result: result.local


Service discovery (optional)

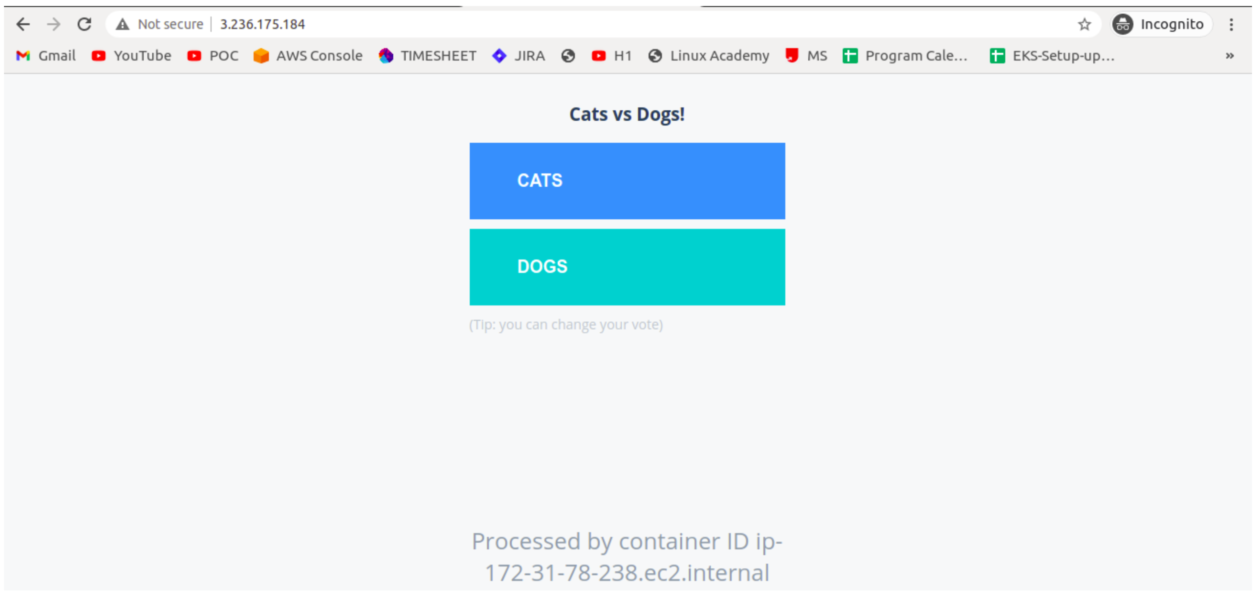
Service discovery endpoint result.local

Service discovery name result

DNS record typ...	Contain...	TTL
A	--	60

Namespace local (PRIVATE)

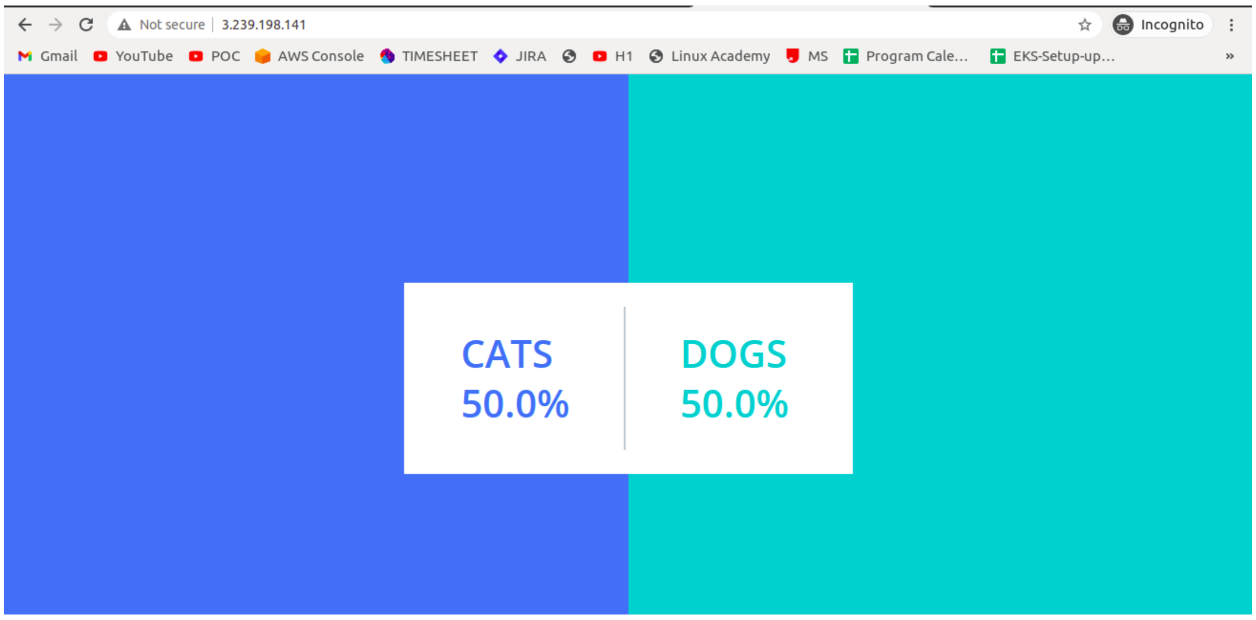
4. Launch the services
    a. Verify logs in to Task definition
    b. Verify Frontends by hitting public IP and workflow



Cats vs Dogs!

(Tip: you can change your vote)

Processed by container ID ip-172-31-78-238.ec2.internal



CATS
50.0%

DOGS
50.0%

Disclaimer

- After completing your work, you must terminate all the AWS resources such as EC2 instances, RDS instances, S3 buckets, or any other AWS resource you will create during the hands-on activity.
- Delete the custom **VPC,** and **NAT** gateways after the work get completed. Stop running ECS tasks and update service definition to set desired tasks as 0 before leaving the ECS console. Delete the **ECS -EC2 cluster** and terminate all its EC2 instances when not in use.
- Do not forget to delete all the **load balancers** and **autoscaling groups** before logging out from your account.

Note: Please create an EC2 instance with less than t2.medium size to avoid budget overshoot and AWS account suspension.