ECS, Fargate, ECR, & EKS

Amazon ECS

Elastic container service. It is Amazon's own container platform. Let you launch container instance and run them.

If you want to launch docker container on AWS you will be launching **ECS Tasks**. Task definition tells what docker image to use to run the container, how much CPU and RAM are given to each container, and other configuration. The ECS task will be launched within ECS clusters.

ECS clusters can be EC2 launch type or fargate launch type.

EC2 launch type

Your cluster will be comprised of EC2 instance underneath which you have to provision in advance. After your container needs to be running on some kind of host right? In this case you will be running the container in EC2 instances.

In order to make the cluster using EC2 instances, each of the EC2 instances have to run ECS Agent to register itself into the ECS service and cluster that's specified.

Only after you run ECS Agent and regsiter itself will then able to run ECS task. AWS will take care of starting / stopping the containers.

Fargate launch type

This time you don't need to provision any EC2 instance underneath, it is all serverless (although there will be servers underneath but you don't have to worry about it!).

It is still considered a ECS cluster even though it is serverless :)

For fargate launch type you just need to define task definitions, then AWS will run the task for you without having you worrying about allocating the servers. Launch more tasks and it will scale automatically because is serverless!

Launching ECS tasks

When you launch your ECS tasks you get the option to pick which launch type. Either via the EC2 launch type from the cluster you defined. Or via fargate which you don't need to manage the infrastructure yourself. You get to pick it.

Service and tasks in a cluster

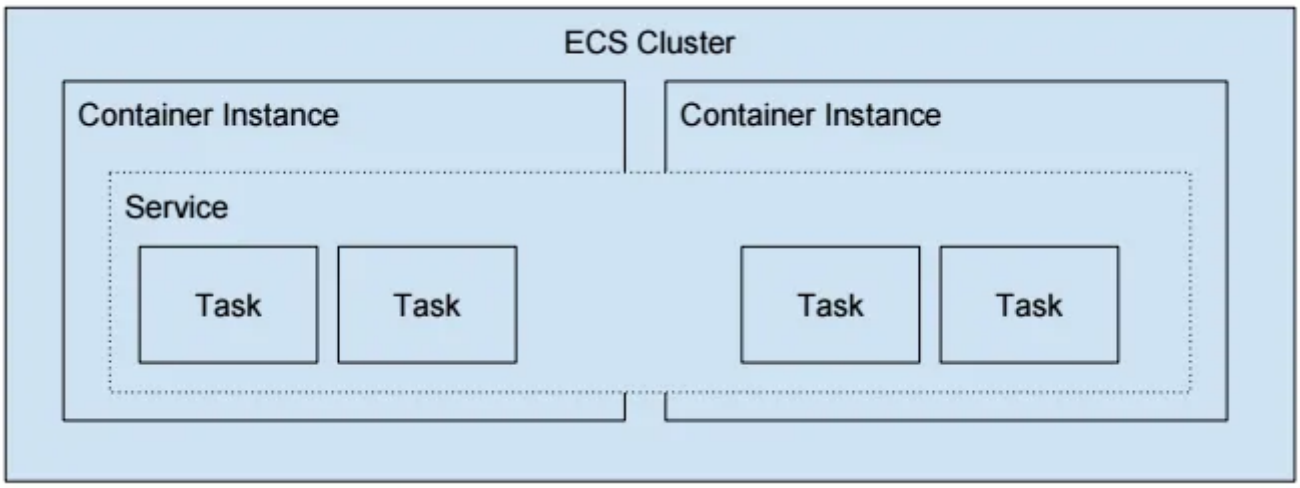
Task definition again tells AWS how to spin up the container. It contains the image links, configurations, how much CPU and RAM to give the container.

Task, is a running container with the defined Task definitions

Service, so you will be launching task under a service which is finally under the cluster. Service allows you specify the number of running task that you should maintain at all times, if any of the tasks gets unhealthy, it will stop it and then replace with a healthy one.

Service uses the cluster's resources , so it could run one task in one instance and another in another EC2 instances.

You could run task directly without being under a service, this is good for short-term task or one time task.



IAM for ECS

EC2 instance profile role: This is only for EC2 launch type only because only EC2 launch type have ECS Agent. The profile will be used by **ECS agent** to make API to ECS service, ECR (for pulling docker images), and CloudWatch for sending logs.

ECS task role: Valid for both fargate and EC2 launch type. This allow each task to have a specific IAM role. This is so that each of your container can have access to different AWS service access. Task A need access to S3, Task B need access to a DynamoDB.

Load balancer integrations

You can expose containers running in clusters in front of a load balancer. ALB is supported.

NLB is only recommended if you need high throughput / high performance or pair it with AWS private link.

Don't use the classic load balancer

Data volumes (EFS)

To have persistent data, because container's data are not saved, you need to use EFS file system. You would mount them in your container in order to save data from the container.

Otherwise, files you write in the container will just be deleted after the container finishes.

EFS is multi-AZ also serverless because you don't have to worry about provisioning the storage it has. Scales automatically.

If you pair it with Fargate + EFS, this is serverless design. Fargate again you do not need to manage the infrastructure that runs the container yourself, it is all serverless. Scales automatically for you.

S3 cannot be used as a mount file system!

ECS Auto scaling

There is two level of scaling, scaling on service level and scaling on EC2 level.

To automatically scale the number of task that's running in a service you can use AWS Application Auto Scaling, the metric that you can use are:

- Scale it on average CPU utilization on Service
- Memory utilization
- ALB request count per target

Target tracking (scale it based on target metric value), step scaling (scale it based on specified CloudWatch alarm), or scheduled scaling (Scale it based on a specific time).

Remember that scaling the tasks doesn't mean the EC2 instances are scaled. Application Auto Scaling just increases the number of running container if the CPU utilization for that service exceeded threshold, or whatever you set your scaling policy to be. It is scaling for each service! You increase the number of task (containers).

To scale the EC2 instances you can use ASG based on CPU utilization like how we used before. Or the smarter way is ECS Cluster Capacity Provider and as soon as you lack capacity to launch tasks it will scale. It is paired with ASG. So if you use ASG with ECS cluster use it with capacity provider!

ECS Solution architecture

1. ECS task invoked by event bridge

S3 bucket sent event to event bridge, you can set rule to run a ECS task, to process whatever the event from the S3. Then sent result to DynamoDB.

This is serverless architecture.

2. Event bridge schedule

Can schedule a rule to trigger every hour to run a ECS task every hour. The container can do something every hour.

3. SQS queue

Message sent to SQS queue, then service in ECS cluster poll for messages and then let the task process it. You can scale ECS for task level based on the queue length as well. I.e increase the number of task based on the length of queue.

ECR

Elastic container registry. This is where you can store and pull images on AWS. There are both private and public registry for you to store images.

Fully integrated with ECS. When your EC2 instances want to pull image from ECR it needs sufficient permission from IAM role.

There is image vulnerability scanning built-in to ECR.

EKS

Elastic Kubernetes services. Another way of managing containers.

Kubernetes is a system for automatically deployment, scaling, and management of containers. It is an alternative to ECS because it is open-source.

You can launch Kubernetes backed by EC2 instances or Fargate which is serverless.

Use EKS if your company is already using Kubernetes on-premises or in another cloud and is migrating to AWS. Kubernetes itself work with any cloud provider.

Tasks in Kubernetes are called EKS pods.

EKS node are like EC2 instances that the pods are running in.

Node types

Managed node groups: AWS create and manages nodes (EC2 instances) for you. Can do on-demand or spot instances

Self-managed nodes: You have to create the nodes yourself and register them to the cluster. On-demand or spot instances

Fargate: You can also use fargate with EKS, you don't have to manage any nodes.

Data volumes

Data volumes can be attached to cluster for persistent data. It uses **Container Storage interface** .

EBS, EFS (only with fargate), FSx for Lustre, FSx for NetApp ONTAP. Can be use as volumes.

App Runner

Managed service to make it easy to deploy web apps and APIs at scale

You don't need to know any infrastructures.

You start with source code or container image, then you do configuration on CPU, RAM, for your web application. Then AWS will build and deploy your web app. Magic.

Automatic scaling, highly available, load balancer, encryption, you can connect to database, cache, and message broker.

Revision #6
Created 2023-02-23 00:54:03 UTC by Tamarine
Updated 2023-07-27 01:56:45 UTC by Tamarine