Amazon Elastic Container Service (ECS)

1. What is Amazon Elastic Container Service?

- ECS is a highly scalable, fast, container management service that makes it easy to run, stop, and manage Docker containers on a cluster.
- ECS runs your containers on a cluster of Amazon EC2 (Elastic Compute Cloud) virtual machine instances pre-installed with Docker.
- It handles installing containers, scaling, monitoring, and managing these instances through both an API and the AWS Management Console.
- It allows you to simplify your view of EC2 instances to a pool of resources, such as CPU and memory.
- The specific instance a container runs on, and maintenance of all instances, is handled by the platform. You don't have to think about it.
- You can host your cluster on a serverless infrastructure that is managed by Amazon ECS by launching your services or tasks using the Fargate launch type.
- Amazon ECS can be used to create a consistent deployment and build experience, manage, and scale batch and Extract-Transform-Load (**ETL**) workloads, and build sophisticated application architectures on a microservices model.
- Amazon AWS is a regional service.

1.1 Features of Amazon ECS

- Amazon AWS is a regional service that simplifies running application containers in a highly available manner across multiple Availability Zones within a Region.
- You can create Amazon ECS clusters within a new or existing VPC.
- After a cluster is up and running, you can define task definitions and services that specify which Docker container images to run across your clusters.
- Container images are stored in and pulled from container registries, which may exist within or outside of your AWS infrastructure.

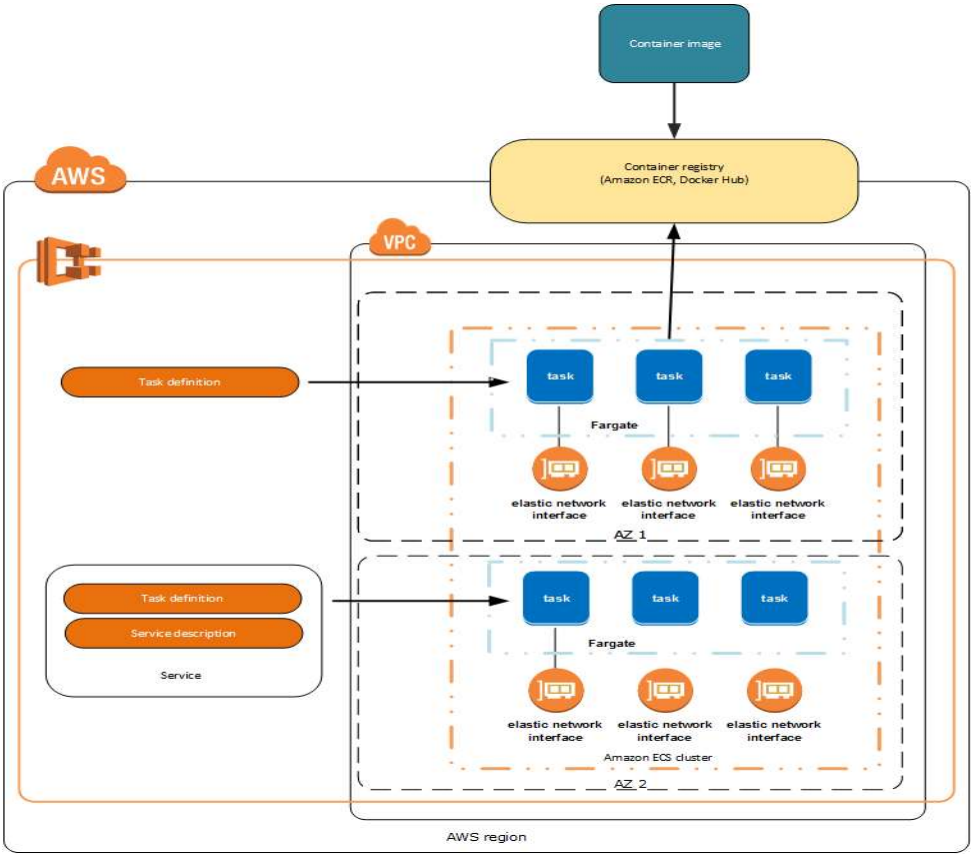
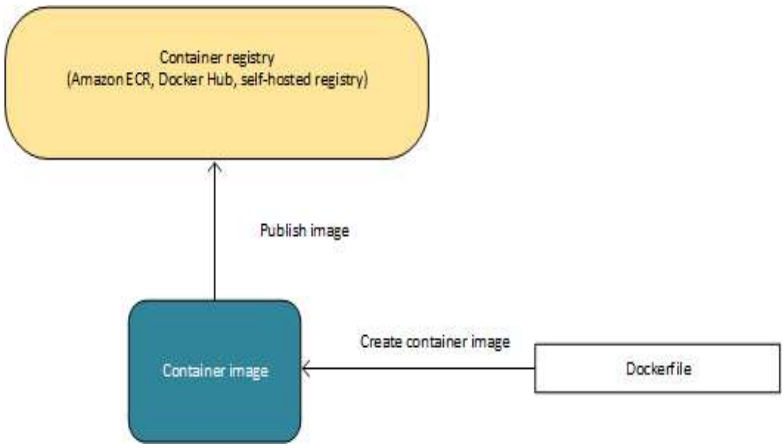


Fig: **Architecture of an Amazon ECS environment**

1.1.1. Containers and Images

- To deploy applications on Amazon ECS, your application components must be architected to run in ***containers***.
- Containers are created from a read-only template called an ***image***.
- Images are typically built from a **Dockerfile**, a plain text file that specifies all of the components that are included in the container. These images are then stored in a ***registry*** from which they can be downloaded and run on your cluster.



1.1.2. Task Definitions

- To prepare your application to run on Amazon ECS, you create a ***task definition***. The task definition is a text file, in JSON format, that describes one or more containers, up to a maximum of ten, that form your application.

- Task definitions specify various parameters for your application.

  - Task family – the name of the task, and each family can have multiple revisions.

  - IAM task role – specifies the permissions that containers in the task should have.

  - Network mode – determines how the networking is containers.

  - Container definitions - specify which image to use, how much CPU and memory the container are allocated, and many more options.

  - Volumes – allow you to share data between containers and even persist the data on the container instance when the containers are no longer running.

  - Task placement constraints – lets you customize how your tasks are placed within the infrastructure.

  - Launch types – determines which infrastructure your tasks use.

  - Example:

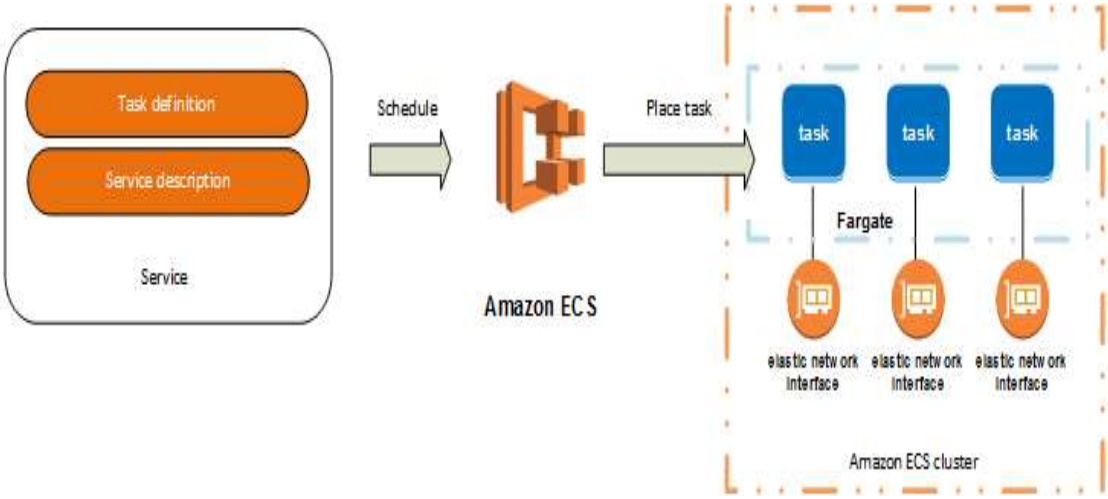
```
{
    "family": "webserver",
    "containerDefinitions": [
        {
            "name": "web",
            "image": "nginx",
            "memory": "100",
            "cpu": "99"
        },
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "networkMode": "awsvpc",
    "memory": "512",
    "cpu": "256",
}
```

1.1.3. Tasks and Scheduling

- A ***task*** is the instantiation of a task definition within a cluster. After you have created a task definition for your application within Amazon ECS, you can specify the number of tasks that will run on your cluster.

- Each task that uses the Fargate launch type has its own isolation boundary and does not share the underlying kernel, CPU resources, memory resources, or elastic network interface with another task.

- The Amazon ECS **task scheduler** is responsible for placing tasks within your cluster.
- There are several different scheduling options available.
    - **REPLICA** — places and maintains the desired number of tasks across your cluster. By default, the service scheduler spreads tasks across Availability Zones. You can use task placement strategies and constraints to customize task placement decisions.
    - **DAEMON** — deploys exactly one task on each active container instance that meets all of the task placement constraints that you specify in your cluster. When using this strategy, there is no need to specify a desired number of tasks, a task placement strategy, or use Service Auto Scaling policies.
- You can upload a new version of your application task definition, and the ECS scheduler automatically starts new containers using the updated image and stop containers running the previous version.

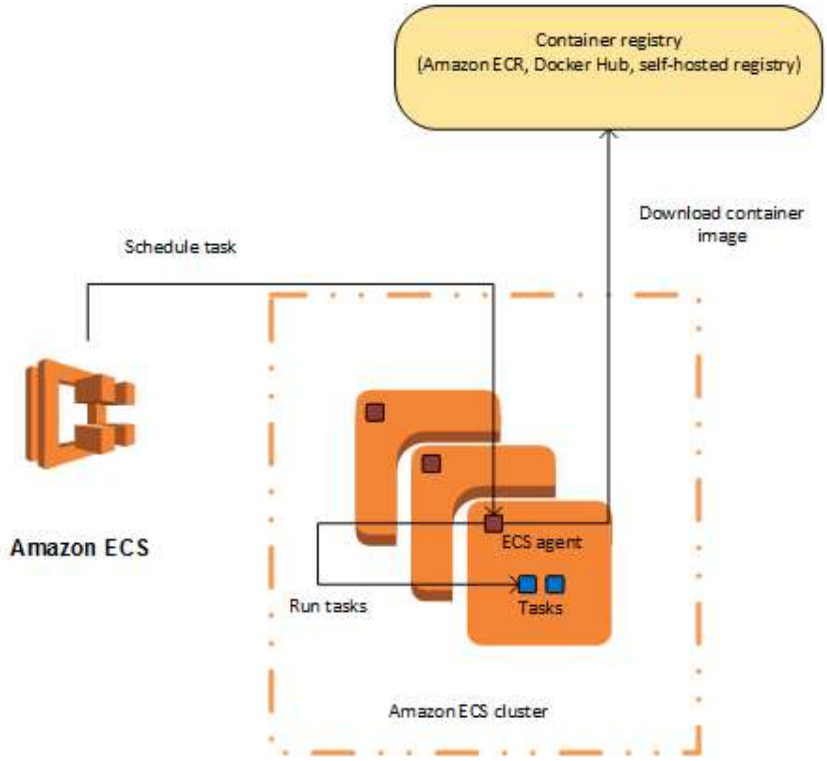


1.1.4. Clusters

- When you run tasks using Amazon ECS, you place them on a ***cluster***, which is a logical grouping of resources.
- Clusters are Region-specific
- Clusters can contain task using both Fargate and EC2 Launch Type.
- When using the Fargate launch type with tasks within your cluster, Amazon ECS manages your cluster resources.
- When using the EC2 launch type, then your clusters are a group of container instances you manage. An Amazon ECS container instance is an Amazon EC2 instance that is running the Amazon ECS container agent. Amazon ECS downloads your container images from a registry that you specify, and runs those images within your cluster.
- Before you can delete a cluster, you must delete the services and deregister the container instances inside that cluster.

1.1.5. Container Agent

- The ***container agent*** runs on each infrastructure resource within an Amazon ECS cluster.
- It sends information about the resource's current running tasks and resource utilization to Amazon ECS, and starts and stops tasks whenever it receives a request from Amazon ECS.
- Container agent is only supported on Amazon EC2 instances.



2. Getting Started with Amazon ECS

2.1 Amazon ECS using Fargate

Step 1: Create a Task Definition

- A task definition is like a blueprint for your application.
- Each time you launch a task in Amazon ECS, you specify a task definition. The service then knows which Docker image to use for containers

1. Open the Amazon ECS console first-run wizard at https://console.aws.amazon.com/ecs/home#/firstRun.
2. From the navigation bar, select the **US East (N. Virginia)** Region.

Note: You can complete this first-run wizard using these steps for any Region that supports Amazon ECS using Fargate.

3. Configure your container definition parameters.

   For **Container definition**, the first-run wizard comes preloaded with the `sample-app`, `nginx`, and `tomcat-webservercontainer` definitions in the console. You can optionally rename the container or review and edit the resources used by the container (such as CPU units and memory limits) by choosing **Edit** and editing the values shown

Note: If you are using an Amazon ECR image in your container definition, be sure to use the full registry/repository:tag naming for your Amazon ECR images.
For example: *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`my-web-app`*`:`*`latest`*.

4. For **Task definition**, the first-run wizard defines a task definition to use with the preloaded container definitions. You can optionally rename the task definition and edit the resources used by the task (such as the **Task memory** and **Task CPU**values) by choosing **Edit** and editing the values shown

   Task definitions created in the first-run wizard are limited to a single container for simplicity. You can create multi-container task definitions later in the Amazon ECS console.

5. Choose **Next**.

Step 2: Configure the Service

- Configure the Amazon ECS service that is created from your task definition.
- A service launches and maintains a specified number of copies of the task definition in your cluster.
- The **Amazon ECS sample** application is a web-based Hello World–style application that is meant to run indefinitely. By running it as a service, it restarts if the task becomes unhealthy or unexpectedly stops.

1. In the **Service name** field, select a name for your service.
2. In the **Number of desired tasks** field, enter the number of tasks to launch with your specified task definition.
3. In the **Security group** field, specify a range of IPv4 addresses to allow inbound traffic from, in CIDR block notation. For example, `203.0.113.0/24`.

4. (Optional) You can choose to use an Application Load Balancer with your service. When a task is launched from a service that is configured to use a load balancer, the task is registered with the load balancer. Traffic from the load balancer is distributed across the instances in the load balancer.

Important: Application Load Balancers do incur cost while they exist in your AWS resources. Complete the following steps to use a load balancer with your service.

    a. In the **Container to load balance** section, choose the **Load balancer listener port**. The default value here is set up for the sample application, but you can configure different listener options for the load balancer.
5. Review your service settings and click **Save**, **Next**.

Step 3: Configure the Cluster

- Here you name your cluster, and then Amazon ECS takes care of the networking and IAM configuration for you.

1. In the **Cluster name** field, choose a name for your cluster.
2. Click **Next** to proceed.

Step 4: Review

1. Review your task definition, task configuration, and cluster configuration and click **Create** to finish. You are directed to a**Launch Status** page that shows the status of your launch. It describes each step of the process (this can take a few minutes to complete while your Auto Scaling group is created and populated).
2. After the launch is complete, choose **View service**.

Step 5: (Optional) View your Service

- If your service is a web-based application, such as the Amazon ECS sample application, you can view its containers with a web browser.

1. On the **Service:** ***`service-name`*** page, choose the **Tasks** tab.
2. Choose a task from the list of tasks in your service.
3. In the **Network** section, choose the **ENI Id** for your task. This takes you to the Amazon EC2 console where you can view the details of the network interface associated with your task, including the **IPv4 Public IP** address.
4. Enter the **IPv4 Public IP** address in your web browser and you should see a webpage that displays the **Amazon ECS sample**application.

Step 6: Clean Up

- Some Amazon ECS resources, such as tasks, services, clusters, and container instances, are cleaned up using the Amazon ECS console.
- Other resources, such as Amazon EC2 instances, Elastic Load Balancing load balancers, and Auto Scaling groups, must be cleaned up manually in the Amazon EC2 console or by deleting the AWS CloudFormation stack that created them.

1. Open the Amazon ECS console at https://console.aws.amazon.com/ecs/.
2. In the navigation pane, choose **Clusters**.
3. On the **Clusters** page, select the cluster to delete.
4. Choose **Delete Cluster**. At the confirmation prompt, enter **delete me** and then choose **Delete**. Deleting the cluster cleans up the associated resources that were created with the cluster, including Auto Scaling groups, VPCs, or load balancers.

2.2 Getting Started Amazon ECS with EC2 instance type

Step 1: Create a Task Definition

1. Open the Amazon ECS console first-run wizard at https://console.aws.amazon.com/ecs/home#/firstRun.
2. From the navigation bar, select the **South America (Sao Paulo)** Region.
3. Configure your task definition parameters.

The first-run wizard comes preloaded with a task definition named `console-sample-app-static`, and you can see the `simple-app` container defined in the console. You can optionally rename the task definition or review and edit the resources used by the container (such as CPU units and memory limits). Choose the container name and editing the values shown (CPU units are under the **Advanced options** menu). Task definitions created in the first-run wizard are limited to a single container for simplicity. You can create multi-container task definitions later in the Amazon ECS console.

4. Choose **Next step**.

Step 2: Configure the Service

1. For **Service name**, select a name for your service.
2. For **Desired number of tasks**, enter the number of tasks to launch with your specified task definition.
3. (Optional) You can choose to use an Application Load Balancer with your service. When a task is launched from a service that is configured to use a load balancer, the task is registered with the load balancer. Traffic from the load balancer is distributed across the instances in the load balancer.

Important: Application Load Balancers do incur cost while they exist in your AWS resources.

    a. Choose the **Application Load Balancer listener port**. The default value here is set up for the sample application, but you can configure different listener options for the load balancer.
    b. In the **Application Load Balancer target group name** field, specify a name for the target group.
4. Review your service settings and choose **Next step**.

Step 3: Configure the Cluster

1. For **Cluster name**, choose a name for your cluster.
2. For **EC2 instance type**, choose the instance type to use for your container instances. Instance types with more CPU and memory resources can handle more tasks.
3. For **Number of instances**, type the number of Amazon EC2 instances to launch into your cluster for task placement. The more instances you have in your cluster, the more tasks you can place on them. Amazon EC2 instances incur costs while they exist in your AWS resources.
4. Select a key pair name to use with your container instances. This is required for you to log into your instances with SSH. If you do not specify a key pair here, you cannot connect to your container instances with SSH. If you do not have a key pair, you can create one in the Amazon EC2 console..
5. (Optional) In the **Security Group** section, you can choose a CIDR block that restricts access to your instances. The default value (**Anywhere**) allows access from the entire internet.
6. In the **Container instance IAM role** section, choose an existing Amazon ECS container instance (ecsInstanceRole) role that you have already created, or choose **Create new role** to create the required IAM role for your container instances.
7. Choose **Review & launch**.

Step 4: Review

1. Review your task definition, task configuration, and cluster configurations and click **Launch instance & run service** to finish. (this can take a few minutes to complete while your Auto Scaling group is created and populated).
2. After the launch is complete, choose **View service**.

Step 5: (Optional) View your Service

- If your service is a web-based application, such as the simple-app application, you can view its containers with a web browser.

1. On the **Service:** ***service-name*** page, choose **Tasks**.
2. Choose a task from the list of tasks in your service.
3. In the **Containers** section, expand the container details. In the **Network bindings** section, for **External Link** you will see the **IPv4 Public IP** address to use to access the web application.
4. Enter the **IPv4 Public IP** address in your web browser and you should see a webpage that displays the **Amazon ECS sample** application.

Step 6: Clean Up

1. Open the Amazon ECS console at https://console.aws.amazon.com/ecs/.
2. In the navigation pane, choose **Clusters**.
3. On the **Clusters** page, select the cluster to delete.

Note: If your cluster has registered container instances, you must deregister or terminate them.

4. Choose **Delete Cluster**. At the confirmation prompt, enter **delete me** and then choose **Delete**. Deleting the cluster cleans up the associated resources that were created with the cluster, including Auto Scaling groups, VPCs, or load balancers.

3. Amazon ECS on AWS Fargate

- AWS Fargate is a technology that you can use with Amazon ECS to **run containers without having to manage servers** or clusters of Amazon EC2 instances.
- With AWS Fargate, you no longer have to provision, configure, or scale clusters of virtual machines to run containers.
- This removes the need to choose server types, decide when to scale your clusters, or optimize cluster packing.
- When you run your tasks and services with the Fargate launch type, you package your application in containers, specify the CPU and memory requirements, define networking and IAM policies, and launch the application.
- Each Fargate task has its **own isolation boundary** and does not share the underlying kernel, CPU resources, memory resources, or elastic network interface with another task.
- Example task definition that sets up a web server using the Fargate launch type:

```
{
   "containerDefinitions": [
      {
         "command": [
            "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS
Sample App</title> <style>body {margin-top: 40px; background-color:
#333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1>
<h2>Congratulations!</h2> <p>Your application is now running on a
container in Amazon ECS.</p> </div></body></html>' >
/usr/local/apache2/htdocs/index.html && httpd-foreground\""
         ],
         "entryPoint": [
            "sh",
            "-c"
         ],
         "essential": true,
         "image": "httpd:2.4",
         "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
               "awslogs-group" : "/ecs/fargate-task-definition",
               "awslogs-region": "us-east-1",
               "awslogs-stream-prefix": "ecs"
            }
         },
         "name": "sample-fargate-app",
         "portMappings": [
            {
               "containerPort": 80,
               "hostPort": 80,
               "protocol": "tcp"
            }
         ]
      }
```

```
   ],
   "cpu": "256",
   "executionRoleArn":
"arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
   "family": "fargate-task-definition",
   "memory": "512",
   "networkMode": "awsvpc",
   "requiresCompatibilities": [
       "FARGATE"
    ]
}
```

- Example of a task definition where two containers are sharing a single volume:

```
{
   "containerDefinitions": [
      {
         "image": "my-repo/database",
         "mountPoints": [
            {
               "containerPath": "/var/scratch",
               "sourceVolume": "database_scratch"
            }
         ],
         "name": "database1",
      },
      {
         "image": "my-repo/database",
         "mountPoints": [
            {
               "containerPath": "/var/scratch",
               "sourceVolume": "database_scratch"
            }
         ],
         "name": "database2",
      }
   ],
   "volumes": [
      {
         "name": "database_scratch"
      }
   ]
}
```

4. Amazon ECS Clusters

4.1 Creating a Cluster

1. Open the Amazon ECS console at https://console.aws.amazon.com/ecs/.
2. From the navigation bar, select the Region to use.
3. In the navigation pane, choose **Clusters**.
4. On the **Clusters** page, choose **Create Cluster**.
5. For **Select cluster compatibility**, choose one of the following options and then choose **Next Step**:

- **Networking only**– With this option, you can launch a cluster of tasks using the Fargate launch type. The Fargate launch type allows you to run your containerized applications without the need to provision and manage the backend infrastructure. When you register your task definition, Fargate launches the container for you.
- **EC2 Linux + Networking**– With this option you can launch a cluster of tasks using the EC2 launch type and Linux containers. The EC2 launch type allows you to run your containerized applications on a cluster of Amazon EC2 instances that you manage.
- **EC2 Windows + Networking** – With this option you can launch a cluster of tasks using the EC2 launch type using Windows containers. The EC2 launch type allows you to run your containerized applications on a cluster of Amazon EC2 instances that you manage.

4.1.1 Using the Networking only cluster template

1. On the **Configure cluster** page, enter a **Cluster name**. Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.
2. In the **Networking** section, configure the VPC for your cluster. You can keep the default settings, or you can modify these settings with the following steps.
    a. (Optional) If you choose to create a new VPC, for **CIDR Block**, select a CIDR block for your VPC.
    b. For **Subnets**, select the subnets to use for your VPC. You can keep the default settings, or you can modify them to meet your needs.
3. In the **Tags** section, specify the key and value for each tag to associate with the cluster.
4. In the **CloudWatch Container Insights** section, choose whether to enable Container Insights for the cluster
5. Choose **Create**.

4.1.2 Using the EC2 Linux + Networking or EC2 Windows + Networking cluster template

1. For **Cluster name**, enter a name for your cluster. Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.
2. (Optional) To create a cluster with no resources, choose **Create an empty cluster**, **Create**.
3. For **Provisioning model**, choose one of the following instance types:
    - **On-Demand Instance**– With On-Demand Instances, you pay for compute capacity by the hour with no long-term commitments or upfront payments.

- **Spot**– Spot Instances allow you to bid on spare Amazon EC2 computing capacity for up to 90% off the On-Demand price.

Note: Spot Instances are subject to possible interruptions. We recommend that you avoid Spot Instances for applications that can't be interrupted.

4. For Spot Instances, do the following; otherwise, skip to the next step.
    a. For **Spot Instance allocation strategy**, choose the strategy that meets your needs.
    b. For **Maximum bid price (per instance/hour)**, specify a bid price. If your bid price is lower than the Spot price for the instance types that you selected, your Spot Instances are not launched.
5. For **EC2 instance type**, choose the Amazon EC2 instance type for your container instances. The instance type that you select determines the EC2 AMI Ids and resources available for your tasks. For GPU workloads, choose an instance type from the P2 or P3 instance family.
6. For **Number of instances**, choose the number of EC2 instances to launch into your cluster. These instances are launched using the latest Amazon ECS-optimized Amazon Linux AMI required by the instance type you chose.
7. For **EC2 AMI Id**, choose the Amazon ECS-optimized AMI for your container instances. The available AMIs will be determined by the Region and EC2 instance type you chose.
8. For **EBS storage (GiB)**, choose the size of the Amazon EBS volume to use for data storage on your container instances. You can increase the size of the data volume to allow for greater image and container storage.
9. For **Key pair**, choose an Amazon EC2 key pair to use with your container instances for SSH access. If you do not specify a key pair, you cannot connect to your container instances with SSH
10. In the **Networking** section, configure the VPC to launch your container instances into. By default, the cluster creation wizard creates a new VPC with two subnets in different Availability Zones, and a security group open to the internet on port 80. This is a basic setup that works well for an HTTP service. However, you can modify these settings by following the substeps below.

    .

    a. For **VPC**, create a new VPC, or select an existing VPC
    b. (Optional) If you chose to create a new VPC, for **CIDR Block**, select a CIDR block for your VPC.
    c. For **Subnets**, select the subnets to use for your VPC. If you chose to create a new VPC, you can keep the default settings or you can modify them to meet

your needs. If you chose to use an existing VPC, select one or more subnets in that VPC to use for your cluster.

d. For **Security group**, select the security group to attach to the container instances in your cluster. If you choose to create a new security group, you can specify a CIDR block to allow inbound traffic from. The default port 0.0.0.0/0 is open to the internet. You can also select a single port or a range of contiguous ports to open on the container instance. For more complicated security group rules, you can choose an existing security group that you have already created.

Note: You can also choose to create a new security group and then modify the rules after the cluster is created

e. In the **Container instance IAM role** section, select the IAM role to use with your container instances. If your account has the **ecsInstanceRole** that is created for you in the console first-run wizard, it is selected by default. If you do not have this role in your account, you can choose to create the role, or you can choose another IAM role to use with your container instances.

Important: The IAM role you use must have the AmazonEC2ContainerServiceforEC2Role managed policy attached to it, otherwise you will receive an error during cluster creation. If you do not launch your container instance with the proper IAM permissions, your Amazon ECS agent does not connect to your cluster.

f. If you chose the Spot Instance type earlier, the **Spot Fleet Role IAM role** section indicates that an IAM roleecsSpotFleetRole is created.

g. In the **Tags** section, specify the key and value for each tag to associate with the cluster.

h. In the **CloudWatch Container Insights** section, choose whether to enable Container Insights for the cluster.

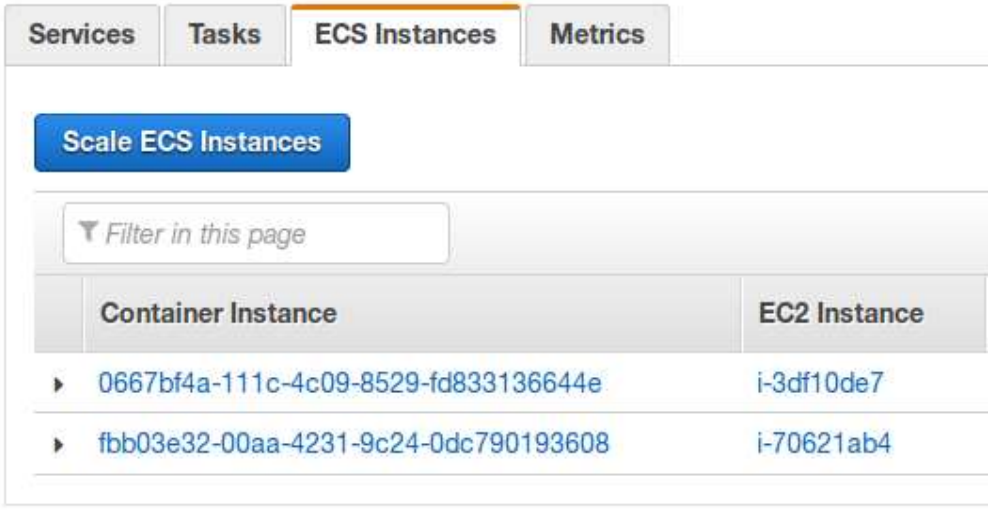
i. Choose **Create**.

4.2 Scaling a Cluster

- If you have a cluster that contains Amazon EC2 container instances, the following helps you scale the number of Amazon EC2 instances in your cluster.
- If Clusters with Fargate tasks can be scaled using Service Auto Scaling.

1. Open the Amazon ECS console at https://console.aws.amazon.com/ecs/.
2. From the navigation bar, choose the Region in which your cluster exists.

3. In the navigation pane, choose **Clusters** and select the cluster to scale.
4. On the **Cluster :** ***name*** page, choose **ECS Instances**.



If a **Scale ECS Instances** button appears, then you can scale your cluster in the next step. If not, you must manually adjust your Auto Scaling group to scale up or down your instances, or you can manually launch or terminate your container instances in the Amazon EC2 console.

5. Choose **Scale ECS Instances**.
6. For **Desired number of instances**, enter the number of instances to which to scale your cluster and choose **Scale**.

Note: If you reduce the number of container instances in your cluster, randomly selected container instances are terminated until the desired count is achieved, and any tasks that are running on terminated instances are stopped.

5. Reference:

- https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html?shortFooter=true
- Running Docker In Production Using Amazon ECS: https://www.edureka.co/blog/docker-container-in-production-amazon-ecs/
- Take Containers From Development To Amazon ECS: https://docs.bitnami.com/aws/how-to/ecs-rds-tutorial/

- Gentle Introduction to How AWS ECS Works with Example Tutorial: https://medium.com/boltops/gentle-introduction-to-how-aws-ecs-works-with-example-tutorial-cea3d27ce63d
- Deploy Docker Containers on Amazon Elastic Container Service (Amazon ECS): https://aws.amazon.com/getting-started/tutorials/deploy-docker-containers/
- https://tutorialsdojo.com/amazon-elastic-container-service-amazon-ecs/
- A beginner's guide to Amazon's Elastic Container Service: https://www.freecodecamp.org/news/amazon-ecs-terms-and-architecture-807d8c4960fd/
- your requirements extend beyond what is supported in this wizard: https://github.com/awslabs/ecs-refarch-cloudformation.