**ECS (ELASTIC CONTAINER SERVICE):**

* It is container orchestration service just like docker swarm and kubernetes which is provided by aws.
* It operates on top of ec2.
* It has built-in private repository called ECR.
* It supports native integration with ELB and ASG.
* Ecs eliminates the need to install and run your own container orchestration tool, manage and scale a cluster of vms. It will take care of them automatically.

**COMPONENTS:**

* **Cluster** -
  + A group of container instances. ( basically, an ec2 with docker and ecs agent).
  + Supports multi AZ.
  + Bound to one region.
* **Task** -
  + Set of one (or) more related containers which is deployed to cluster.
  + Containers within a task are placed on same host.
* **Task Definition** -
  + Serves as template for tasks.
  + Allows you to define docker features accessible (image, vol, network etc).
  + Allows you to define cpu and memory for tasks.
  + Can be configurable using JSON.
* **Service** -
  + It is an other layer on top of tasks.
  + Allows to set how many copies of specific tasks should be running.
  + It maintains the desired number of running tasks.
  + Can be increase and decrease the number of running tasks using service auto scaling.
  + Can bind It to load balancer.

**RESOURCE ALLOCATION:**

* Each container gets portion of cpu and memory from host machine, these are reserved for each container.
* Remaining cpu and memory will be shared by other containers.
* We have to configure resource allocation in task definition.
* There are two types in memory allocation
* **Hard limit** - container is killed when its exceed the reserved amount.
* **Soft limit** - memory can be exceeded if available.
* You have to use one but You can use both limits together.

**AUTOSCALING:**

* You can use autoscaling with ecs clusters. There are two types of asg in ecs.

**Service autoscaling** - scaling the number of tasks for a service.

* Adding more containers to handle increasing load.
* It can be configured inside ecs.
* You can use cpu and memory to trigger scaling events.
* Can use custom cloud watch metrics too.

**Cluster autoscaling** - scaling the number of ec2 in cluster.

* Adding more instances to handle the load.
* Can be configured using ec2 asg.
* You can use cpu and memory to trigger scaling events.
* Service asg is faster than cluster auto scaling.

**STORAGE:**

* Docker containers are volatile.
* Docker uses union file systems for container storage.
* Data that written in **UFS** doesn’t persist.
* For this reason, we can use docker volumes to persist data and share data between containers.
* Docker volumes bypasses UFS.
* Docker volumes are local to a specific host.
* To share storage between you can use shared file systems like **EFS** and **objectiveFS**.
* You have to mount efs on local and mount to containers running on them.

**CI/CD:**

* ECS can use docker images from ecr (or) any other repository.
* You can specify which images are to deploy using task definitions.
* ECS allows you to perform rolling updates to running services.
* Updates can be done automatically using ECS API.
* You can automate this process using jenkins (or) any other CI/CD tools.

**CI/CD WORKFLOW:**

* Checkout code from version control using jenkins.
* Build a docker image.
* Push image to ECR (or) any other repo.
* Jenkins Update the task definition and service
* ECS updates containers on clusters.

**SETUP:**

* First we need to create a ECR repository and Push an docker image to ECR repository.
* Install aws cli
* Login to ECR reposiroty from EC2

**aws ecr get-login --no-include-email --region ap-south-1 (specify your region in ap-south-a)**

* once you execute the above command, you will get an output.
* Copy the output and execute it to login to ECR.
* Once you go the login succed message, you can push your images to ECR.
* To push images, you have to give tag for them.
* To tag an image -

**docker tag reddy:latest 858277153255.dkr.ecr.ap-south-1.amazonaws.com/reddy:latest**

* to push an image to ECR -

**docker push 858277153255.dkr.ecr.ap-south-1.amazonaws.com/reddy:latest**

* once it is done, you have to create two roles for ECS.
* **Ecs-instance-role** - ecs agent to make calls to ecs api on your behalf.
* **Ecs-service-role** - ecs to access other aws resources on your behalf.
* You can also use docker hub in ecs, just specify your **username/repo:tag** and it will pull image from your dockerhub account.

**CLUSTERS:**

* A cluster is a grouping on container instances. You can create multiple clusters in an account.
* An container instance is nothing but a group of ec2 instances that are running ecs agent and has been registered into a cluster.
* To create a cluster:

**Go to cluster page**

**Click create cluster**

**Select cluster template (linux, windows)**

**Cluster name - Specify a name to cluster**

**Ec2 instance type - select an ec2 type**

**Number of instances - specify no of instance to launch in your cluster.**

**EBS - specify storage.**

**Key Pair - choose key pair for ec2**

**VPC - select vpc to launch instances in it.**

* If you didn’t selected any, aws will create a new vpc.

**Container ecsinstancerole - select your ecs instance role**

**Click, create to create your cluster.**

* To delete a cluster:

**Go to clusters page**

**Select a cluster to delete.**

**Choose delete cluster.**

* The cluster will be deleted along with ecs instances.
* To scale cluster:

**Go to cluster page**

**Choose you cluster**

**Go to ecs instances tab**

**Select scale ecs instances**

**Specify your desired number**

**Choose create.**

* It will scale your ecs clusters.
* If you reduce the desired number, it will terminate instances randomly and any tasks that are running in instance will be stopped.
* There might be a situation where you need to perform updates to an container instance, update docker etc. at this type of situations ecs gives use to drain an instance from cluster without impacting tasks on cluster.
* You can drain instances and once the work is finished you can active your instance and it will add instance to cluster again and run tasks on it.

**TASK DEFINITION:**

* Here you have to define all your containers and some parameters.
* You can version these task definitions.
* To create an task definition:

**Task definition name - specify any name.**

**Network mode - container uses this networking.**

**Task size**

**Memory size - memory used by task.**

**Cpu size - cpu used by task.**

**Add container - you have to add container to run in your clusters.**

* Click on add container,

**Container name - specify a name to container.**

**Image - specify your image url (repo/namespace/image:tag)**

**Memory - specify memory type and specify size.**

**Port mappings - specify host and container port.**

* You can add health checks, volumes, networking etc.
* Once you added the image. Click, add container.
* Click create to create a task definition.

**SERVICE:**

* It allows us to set no of copies of tasks to run on clusters. If any task is deleted, it will create another task and maintain the desired state.
* You can specify load balancing and auto scaling using the service.
* It exposes api, so that you can access your containers outside of the machine.

**To create a service**

**Go to clusters page**

**Choose a cluster to create service**

**Click create service**

**Select task definition and revision**

**Select cluster**

**Specify a name to service**

**Specify no of tasks to run**

**Choose service type**

**Replica**  - maintains desired number of tasks across your cluster.

**Daemon** - maintain one copy of your tasks on each ecs instance.

**Click Next**

**Choose a load balancer if you want**

**Click Next**

**Setup service autoscaling if you want.**

**Click Next and Create service.**

* To update a service

**Go to clusters page**

**Choose a cluster to update service**

**Click update service**

**Update all the settings you need**

**Choose update service.**

* To delete a service

**Go to clusters page**

**Choose a cluster to delete service**

**Click delete service**

* It will delete the service. You can create as many services as you want in a cluster. You can choose which service that you want to run in clusters.

**ECS AGENT:**

* If you let ecs to manage the creation and deletion of clusters. The ecs agent will be installed by default.
* But, if you want to use your own ec2 as cluster, you have to launch ec2 with ecs service role and install ecs agent manually by downloading from s3 bucket from your regional (or) from docker hub.
* If you install from S3, you have to verify the validity of the agent.
* You have to install **ecs-init** and **docker**.
* Once installed, start both services.
* you have to download **agent** **tar** file and **GPG signature** to validate, which is available in aws docs.
* Refer the aws ecs documentation to install ecs container agent.

**CLOUDWATCH LOGS:**

* you can configure you rcontainer instances to send log information to cloudwatch.
* To get logs, we have to create an IAM policy to allow your container instances to use cloud watch logs API and attach policy to ecsinstancerole.
* To create a IAM policy

**Go to IAM**

**Choose Policies**

**Click, Create policy and paste the json content.**

**{**

**"Version": "2012-10-17",**

**"Statement": [**

**{**

**"Effect": "Allow",**

**"Action": [**

**"logs:CreateLogGroup",**

**"logs:CreateLogStream",**

**"logs:PutLogEvents",**

**"logs:DescribeLogStreams"**

**],**

**"Resource": [**

**"arn:aws:logs:\*:\*:\*"**

**]**

**}**

**]**

**}**

**Choose review policy and create policy by giving a name to it.**

* Now, attach this policy to ecsinstancerole.
* Go to Roles

**Select your ecsinstancerole**

**Go to permissions tab**

**Click Attach Policy**

**Select your IAM policy from list.**

**Attach Policy.**

* After attaching ecs-logs policy to ecsinstancerole, we have to install cloudwatch logs agent in container instances.
* To install cloudwatch logs agent refer to aws documentation.
* Once you install logs agent, specify the log files in **/etc/awslogs/awslogs.conf** to send logs to cloudwatch logs.
* **/var/log/docker** - docker daemon logs
* **/var/log/ecs/ecs-init.log** - logs from ecs-init upstart job
* **/var/log/ecs/ecs-agent.log** - logs from ecs container agent
* **/var/log/ecs/audit.log** - logs from IAM role task credentials provider
* By default, awslogs sends data to us-west region. To send to your specified region.
* Go to **/etc/awslogs/awscli.conf**
* Specify your region and save the file.
* Start the awslogs service and you can see your logs in cloudwatch logs.

**AUTO SCALING:**

Cluster Scaling:

* You can do scaling for clusters.

**Go to cluster page**

**Choose the cluster to scale.**

**Go to instances tab**

**Select scale ec2 instances**

**Specify the number of desired instances to scale.**

Service Scaling:

* You can do autoscaling for services based on cpu and mem.

**Go to cluster page**

**Choose the cluster to scale**

**Go to service tab**

**Select a service to scale and click update.**

**In step 3, set autoscaling**

**choose configure service autoscaling to adjust your desired count.**

**Specify min, desired and max number of tasks**

**Choose Add scaling policy to configure your scaling policy**

**Scaling type - target tracking**

**Policy name - specify any name.**

**Ecs service metric - cpu (or) ram**

**Target Value - specify threshold value**

**Scale-out period - specify time in sec**

**Scale-in period - specify time in sec**

**Review all your changes and update the service.**

* **Scale-out period** - the amount of time after an scale-out completes, before another scaling-out activity happens.
* **Scale-in period** - the amount of time after scale-in completes, before another scale-in activity happens.
* Once you have configured the service autoscaling, ecs will scale the services once the threshold value Is met.

**LOAD BALANCING:**

* You can create a load balancer and attach It to ecs.
* It will share the load across the tasks in your service.
* You have to create a load balancer before creating a service.
* In the service page, you have to select your load balancer from drop-down list.
* You can’t add load balancer once you have created a service. You have to add while creating the service.
* If a service’s task failed to met lb health check, it will be killed and restarted until your service reaches desired no of running tasks.
* You can use any load balancer type. But alb is mostly useful with ecs. It automatically add instances to target groups with dynamic host port.