Docker POC

Docker is available in two editions**: Community Edition (CE) and Enterprise Edition (EE).**

**Docker Community Edition (CE)** is ideal for developers and small teams looking to get started with Docker and experimenting with container-based apps. Docker CE has two update channels, stable and edge.

1. **Stable:** gives you reliable updates every quarter.
2. **Edge:** gives you new features every month.

**Docker Enterprise Edition (EE)** is designed for enterprise development and IT teams who build, ship, and run business critical applications in production at scale.

Docker CE and EE are available on multiple platforms, on cloud and on-premises.

**Setting Up with Amazon ECS**

1. **Create an IAM User.**

* <http://docs.aws.amazon.com/AmazonECS/latest/developerguide/get-set-up-for-amazon-ecs.html#create-an-iam-user>

1. **Create an IAM Role for your Container Instances and Services.**

**Important**

* <http://docs.aws.amazon.com/AmazonECS/latest/developerguide/instance_IAM_role.html>
* Ensure that the AmazonEC2ContainerServiceforEC2Role managed policy is attached to the role.
* In the navigation pane, choose Roles>Click create new role> Role Name: ecsInstanceRole>Select role Type: AmazonEC2ContainerServiceforEC2Role>Choose Attach Policy.

If you are going to use S3 for store config files: Adding Amazon S3 Read- only Access to your Container Instance Role

* <http://docs.aws.amazon.com/AmazonECS/latest/developerguide/instance_IAM_role.html#container-instance-role-s3>

**Note:** If you do not launch your container instance with the proper IAM permissions, your Amazon ECS agent cannot connect to your cluster.

1. **Amazon ECS Service Scheduler IAM Role**

* <http://docs.aws.amazon.com/AmazonECS/latest/developerguide/service_IAM_role.html>
* Ensure that the AmazonEC2ContainerServiceRole managed policy is attached to the role.
* In the navigation pane, choose Roles>Click create new role> Role Name: ecsServiceRole>Select role Type: AmazonEC2ContainerServiceRole>Choose Attach Policy.

1. **Create a Key Pair**

* <http://docs.aws.amazon.com/AmazonECS/latest/developerguide/get-set-up-for-amazon-ecs.html#create-a-key-pair>

1. **Create a Virtual Private Cloud**

* <http://docs.aws.amazon.com/AmazonECS/latest/developerguide/get-set-up-for-amazon-ecs.html#create-a-vpc>

1. **Create a Security Group**

* <http://docs.aws.amazon.com/AmazonECS/latest/developerguide/get-set-up-for-amazon-ecs.html#create-a-base-security-group>
* Add HTTP port #: 80
* Add SSH port #: 22

These are both managed services on Aws for Postgres & Redis.

* You can add PostgreSQL port #: 5432 (For amazon RDS)
* You can add Redis port #: 6379 (For amazon elasticache)

1. **Install the AWS CLI**

* <http://docs.aws.amazon.com/cli/latest/userguide/installing.html>

**Cluster:**

1. Clusters is a group of container instances that act as a single computing resource.
2. When you run tasks or services they will be scheduled to run on your cluster and you can create zero or more clusters on your AWS account.
3. You can also mix and match instances types on your cluster and they can even span across multiple AZ.
4. Instances cannot join multiple clusters.

**Important:** By default, your container instance launches into your default cluster. To launch into a non-default cluster, choose the Advanced Details list. Then, paste the following script into the User data field, replacing your\_cluster\_name with the name of your cluster.

#!/bin/bash

echo ECS\_CLUSTER=your\_cluster\_name >> /etc/ecs/ecs.config

**Cluster Using AWS Cli:**

1. Create a cluster:

aws ecs create-cluster --cluster-name Docker0

{

"cluster": {

"status": "ACTIVE",

"clusterName": "Docker0",

"registeredContainerInstancesCount": 0,

"pendingTasksCount": 0,

"runningTasksCount": 0,

"activeServicesCount": 0,

"clusterArn": "arn:aws:ecs:ap-southeast-1:872623284944:cluster/Docker0"

}

}

1. List all clusters:

aws ecs list-clusters

{

"clusterArns": [

"arn:aws:ecs:ap-southeast-1:872623284944:cluster/Docker0"

]

}

1. A deeper look into the cluster's state

aws ecs describe-clusters --clusters Docker0

{

"clusters": [

{

"status": "ACTIVE",

"clusterName": "Docker0",

"registeredContainerInstancesCount": 0,

"pendingTasksCount": 0,

"runningTasksCount": 0,

"activeServicesCount": 0,

"clusterArn": "arn:aws:ecs:ap-southeast-1:872623284944:cluster/Docker0"

}

],

"failures": []

}

**Container agent:**

1. The container agent is open source tool which takes care of pluming to ensure amazon Ec2 instance can register to your cluster.
2. Allow container instances to join cluster. (It’s a tool that allows your EC2 instances could join into a cluster.)
3. Amazon ECS Container Agent Versions.

<http://docs.aws.amazon.com/AmazonECS/latest/developerguide/container_agent_versions.html>

1. Amazon ECS-Optimized AMI Container Agent Versions

<http://docs.aws.amazon.com/AmazonECS/latest/developerguide/container_agent_versions.html#ecs-optimized-ami-agent-versions>

1. Container agent is open source tool and its available repo GitHub & Docker hub.

<http://docs.aws.amazon.com/AmazonECS/latest/developerguide/ECS_agent.html>

**Container Agent GitHub repo:**

<https://github.com/aws/amazon-ecs-agent>

**Container Agent on the Docker Hub:**

<https://hub.docker.com/r/amazon/amazon­ecs­agent/>

1. Updates are quick and easy (Updating the container agent on an a existing instance it’s a pain less operation and it will not interrupt tasks , serveries that are running on that container instance.

**Container Instances:**

1. Ec2 instances registered to a cluster. (A container instances are an amazon Ec2 instances that has been register to be a part of the specific cluster.

2. Connects via container agent (Its connects to the cluster using container agent).

3. Life cycle states of container instance:

**Active and connected** (This occurs when the container agent register the container instance on to the cluster were connection status is True once this happens the container agent ready to run tasks.

**Active and disconnected** (Connection status were false. This happens when you stop the container instance. In which case, current task is that running will stop.)

**Inactive**: (This happened if you terminate or deregister your container instance. An inactive instance not be seen in part of the cluster.)

Launching an Amazon ECS-Optimized AMI

1. <http://docs.aws.amazon.com/AmazonECS/latest/developerguide/ecs-optimized_AMI_launch_latest.html>
2. <http://docs.aws.amazon.com/AmazonECS/latest/developerguide/launch_container_instance.html>

To find container instance to joined on cluster:

aws ecs list-container-instances --cluster Docker0

{

"containerInstanceArns": [

"arn:aws:ecs:ap-southeast-1:872623284944:container-instance/cf76a21d-ec4b-49cd-9716-1d876fb9fb70"

To Describe container instance:

aws ecs describe-container-instances --cluster Docker0 --container-instances "arn:aws:ecs:ap-southeast-1:872623284944:container-instance/cf76a21d-ec4b-49cd-9716-1d876fb9fb70"

{

"failures": [],

"containerInstances": [

{

"status": "ACTIVE",

"registeredAt": 1500553051.638,

"registeredResources": [

{

"integerValue": 1024,

"longValue": 0,

"type": "INTEGER",

"name": "CPU",

"doubleValue": 0.0

},

{

"integerValue": 993,

"longValue": 0,

"type": "INTEGER",

"name": "MEMORY",

"doubleValue": 0.0

},

{

"name": "PORTS",

"longValue": 0,

"doubleValue": 0.0,

"stringSetValue": [

"22",

"2376",

"2375",

"51678",

"51679"

],

"type": "STRINGSET",

"integerValue": 0

},

{

"name": "PORTS\_UDP",

"longValue": 0,

"doubleValue": 0.0,

"stringSetValue": [],

"type": "STRINGSET",

"integerValue": 0

}

],

"ec2InstanceId": "i-0083d73d6e393d6e3",

"agentConnected": true,

"containerInstanceArn": "arn:aws:ecs:ap-southeast-1:872623284944:container-instance/cf76a21d-ec4b-49cd-9716-1d876fb9fb70",

"pendingTasksCount": 0,

"remainingResources": [

{

"integerValue": 1024,

"longValue": 0,

"type": "INTEGER",

"name": "CPU",

"doubleValue": 0.0

},

{

"integerValue": 993,

"longValue": 0,

"type": "INTEGER",

"name": "MEMORY",

"doubleValue": 0.0

},

{

"name": "PORTS",

"longValue": 0,

"doubleValue": 0.0,

"stringSetValue": [

"22",

"2376",

"2375",

"51678",

"51679"

],

"type": "STRINGSET",

"integerValue": 0

},

{

"name": "PORTS\_UDP",

"longValue": 0,

"doubleValue": 0.0,

"stringSetValue": [],

"type": "STRINGSET",

"integerValue": 0

}

],

"version": 3,

"attributes": [

{

"name": "ecs.availability-zone",

"value": "ap-southeast-1a"

},

{

"name": "com.amazonaws.ecs.capability.logging-driver.syslog"

},

{

"name": "ecs.instance-type",

"value": "t2.micro"

},

{

"name": "ecs.ami-id",

"value": "ami-19f7787a"

},

{

"name": "com.amazonaws.ecs.capability.task-iam-role-network-host"

},

{

"name": "com.amazonaws.ecs.capability.logging-driver.awslogs"

},

{

"name": "com.amazonaws.ecs.capability.logging-driver.json-file"

},

{

"name": "com.amazonaws.ecs.capability.docker-remote-api.1.17"

},

{

"name": "com.amazonaws.ecs.capability.privileged-container"

},

{

"name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"

},

{

"name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"

},

{

"name": "com.amazonaws.ecs.capability.ecr-auth"

},

{

"name": "ecs.os-type",

"value": "linux"

},

{

"name": "com.amazonaws.ecs.capability.docker-remote-api.1.20"

},

{

"name": "com.amazonaws.ecs.capability.docker-remote-api.1.21"

},

{

"name": "com.amazonaws.ecs.capability.docker-remote-api.1.22"

},

{

"name": "com.amazonaws.ecs.capability.task-iam-role"

},

{

"name": "com.amazonaws.ecs.capability.docker-remote-api.1.23"

}

],

"versionInfo": {

"agentVersion": "1.14.3",

"agentHash": "15de319",

"dockerVersion": "DockerVersion: 17.03.1-ce"

},

"runningTasksCount": 0

}

]

}