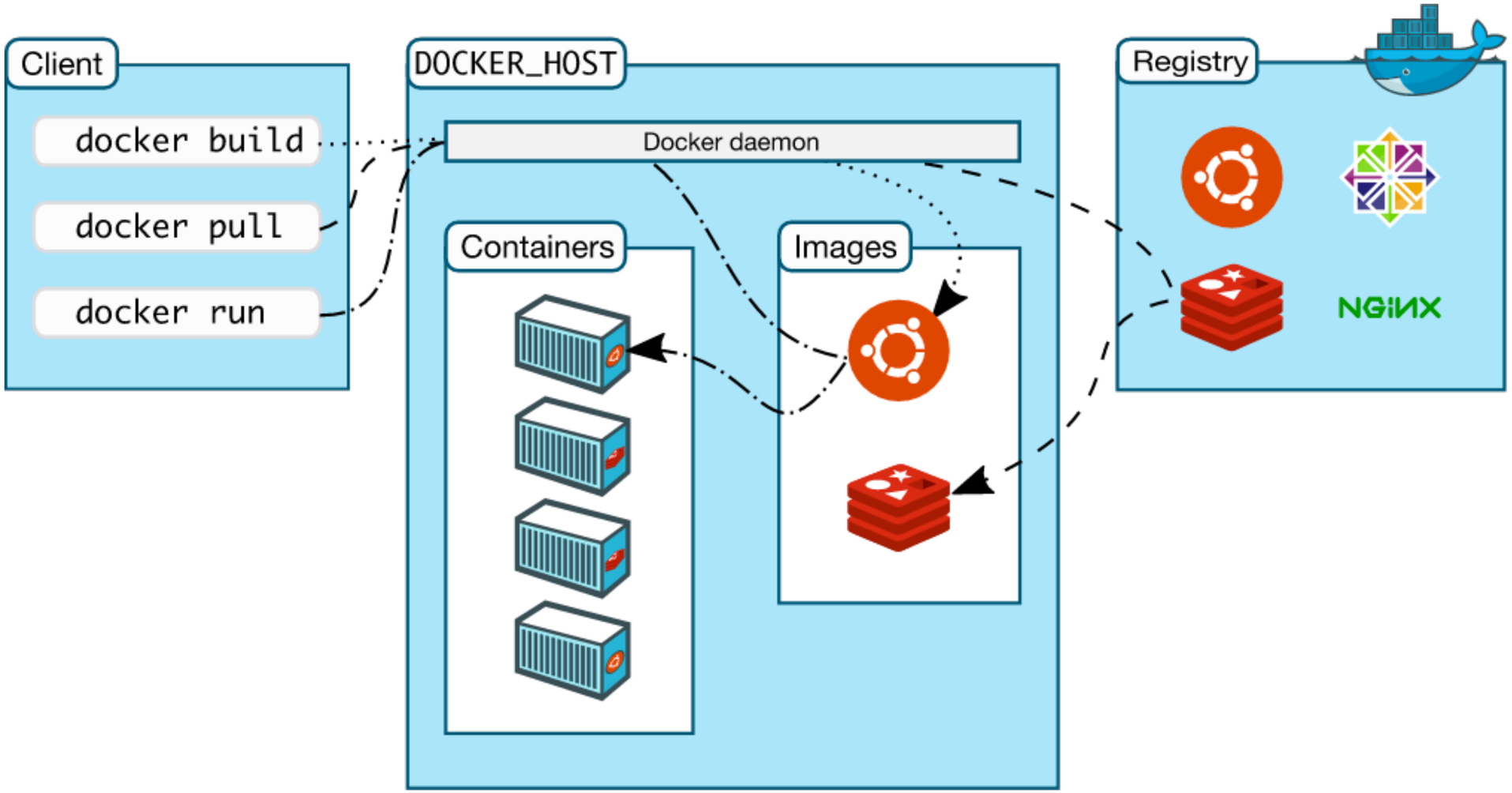
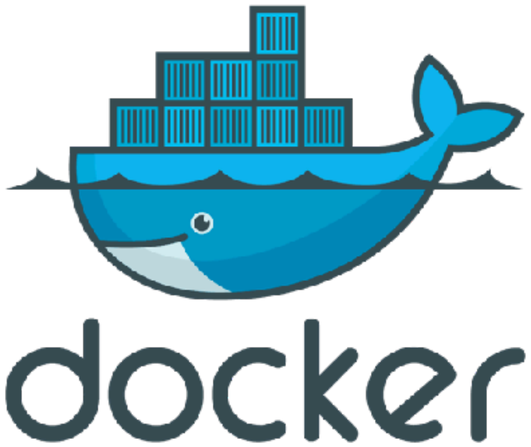
Container Orchestration on Amazon Web Services

Arun Gupta, @arungupta

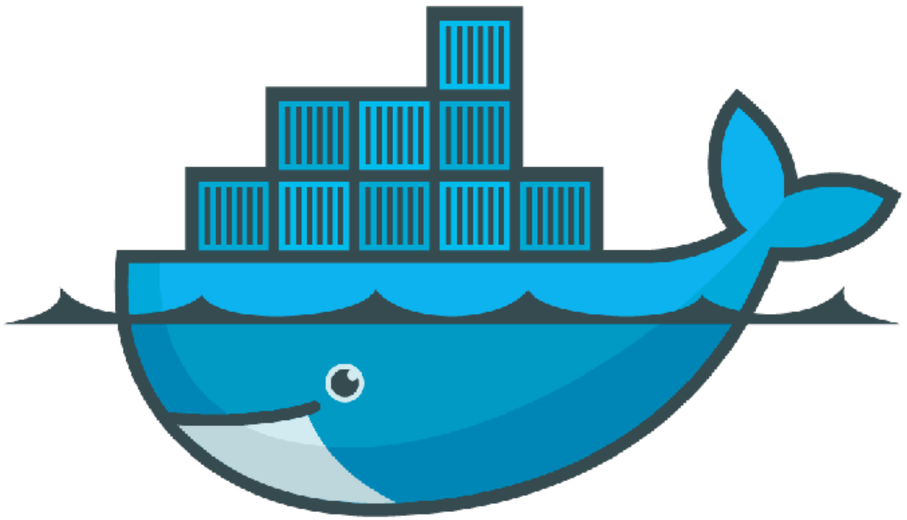
Docker Workflow





kubernetes

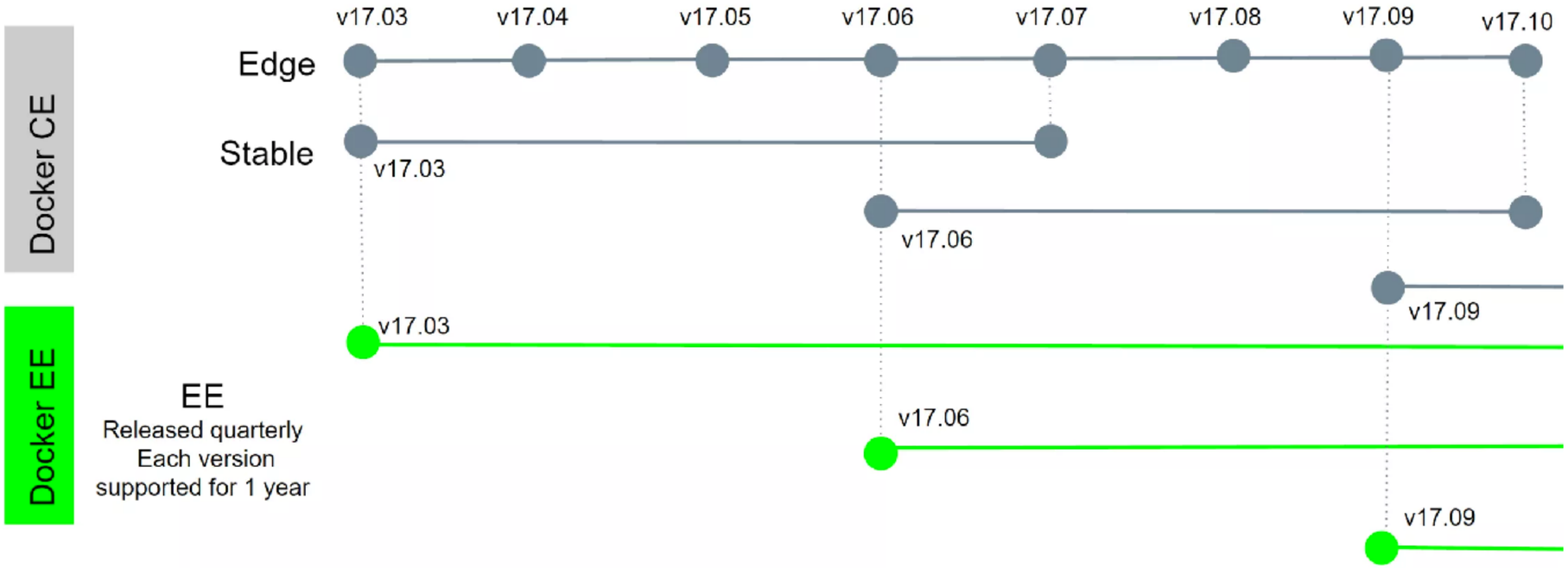
MESOSPHERE



docker

Development using Docker

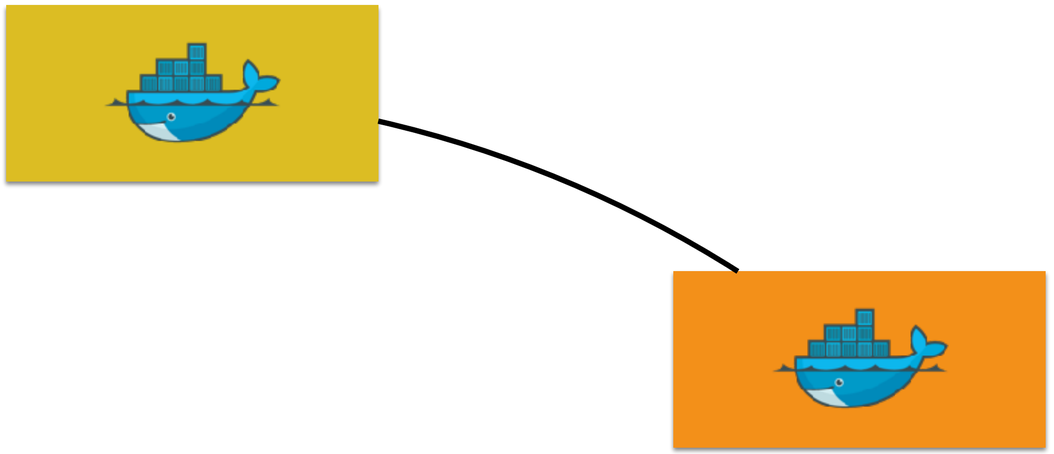
- Docker Community Edition
  - Docker for Mac/Windows/Linux and AWS/Azure
  - Monthly edge and quarterly stable releases
  - Native desktop or cloud provider experience



Swarm-mode: Initialize

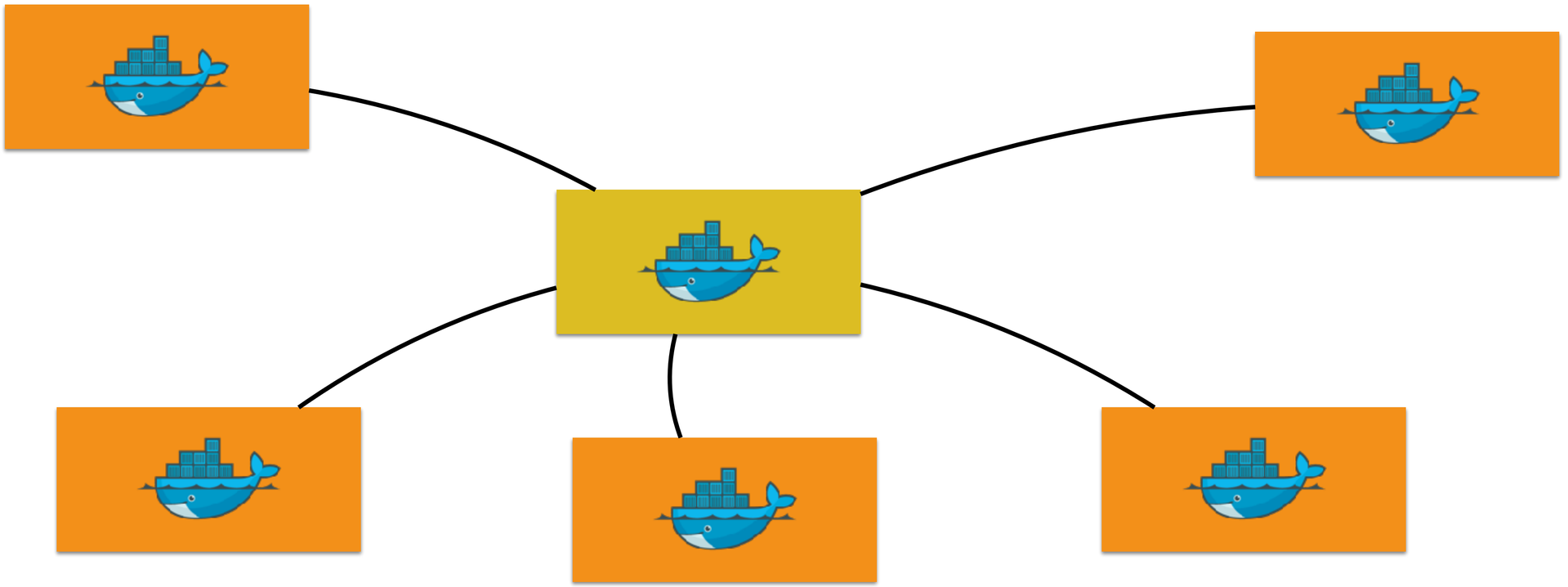
```
docker swarm init --listen-addr <ip>:2377
```

Swarm-mode: Add Worker



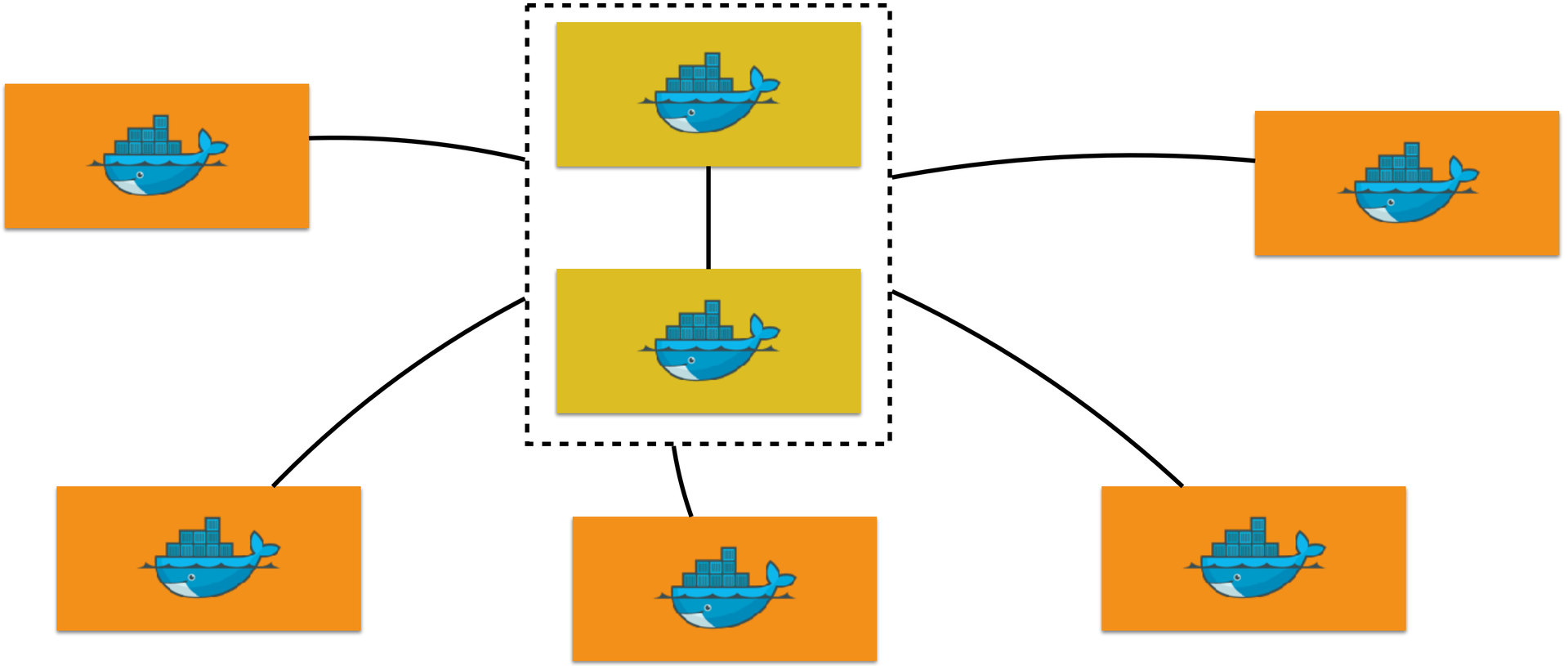
```
docker swarm join --token <token> <manager>:2377
```

Swarm-mode: Add More Workers



```
docker swarm join --token <token> <manager>:2377
```

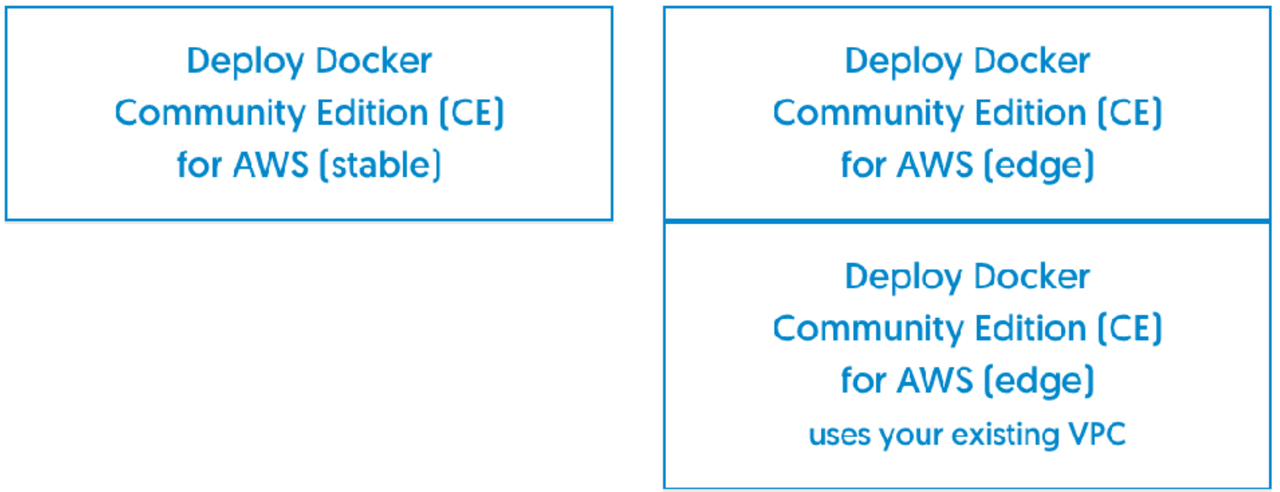
Swarm-mode: Primary/Secondary Master



```
docker swarm join --manager --token <token> --listen-addr
<master2>:2377 <master1>:2377
```

Docker for AWS

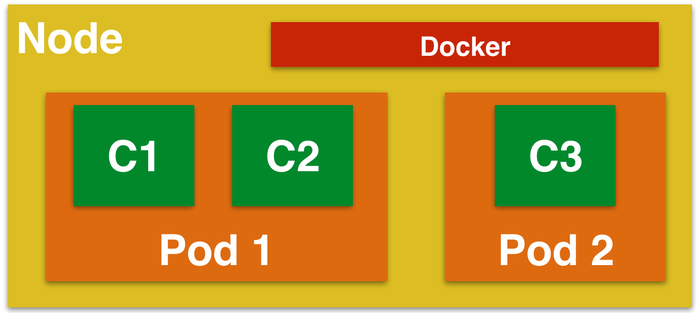
- CloudFormation template
- Integrated with AWS Infrastructure
  - Autoscaling Groups (ASG)
  - Elastic Load Balancer (ELB)
  - Elastic Block Store (EBS)

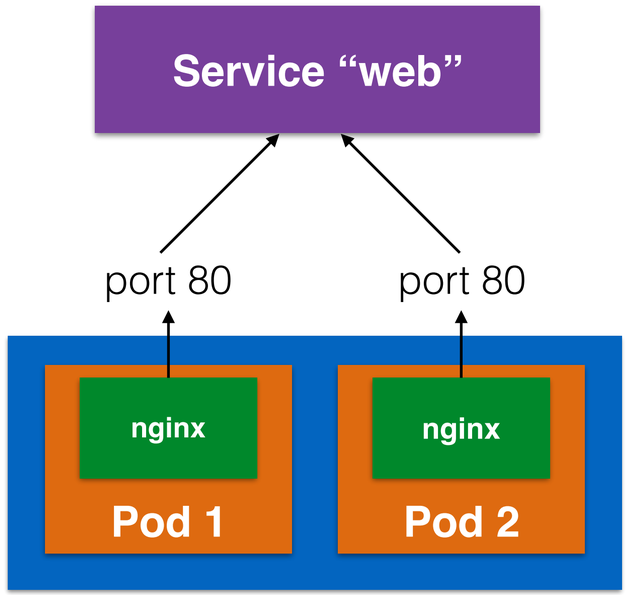


kubernetes

Kubernetes Concepts

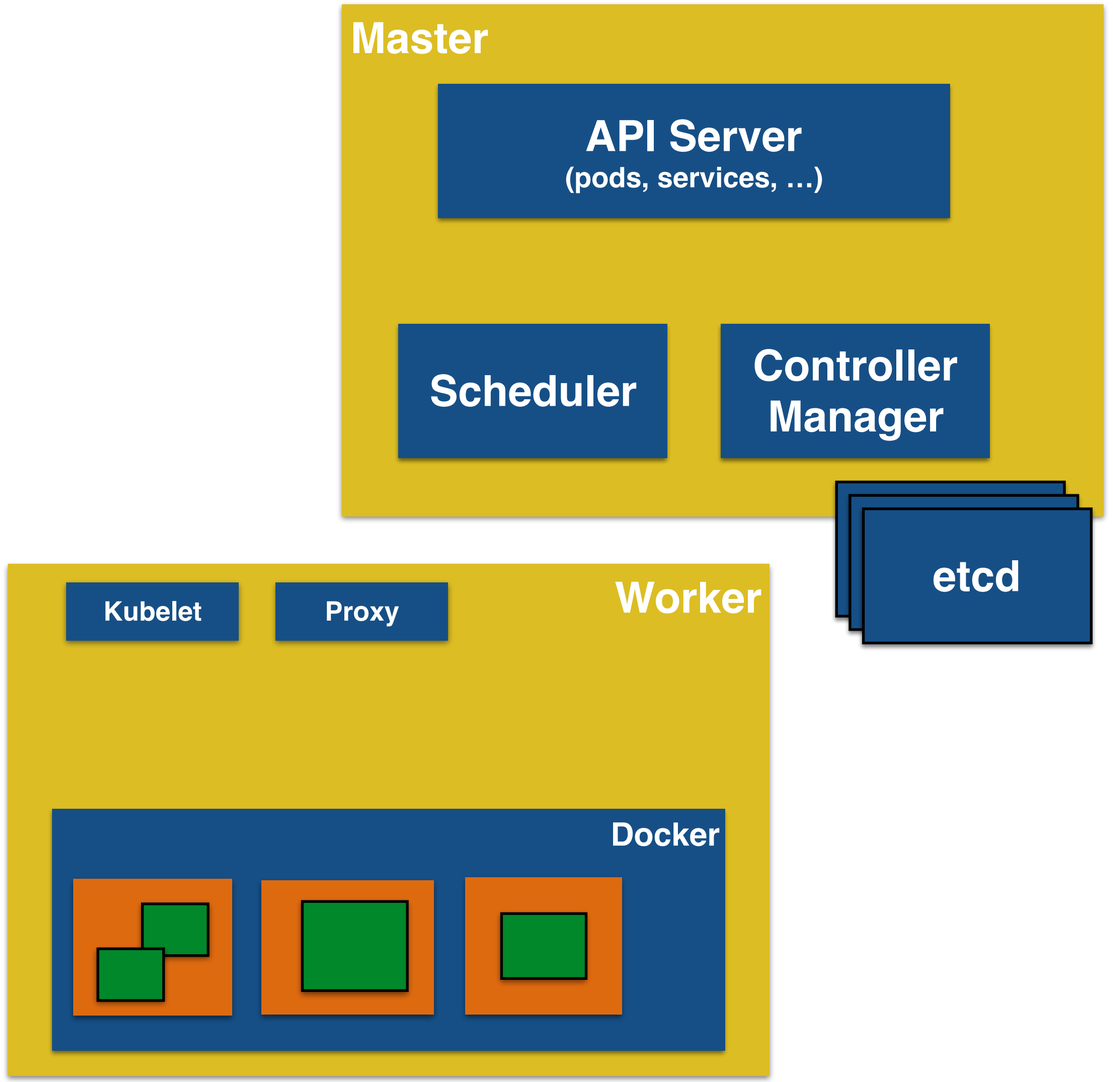
- **Pods**: collocated group of Docker containers that share an IP and storage volume
- **Service**: Single, stable name for a set of pods, also acts as LB
- **Label**: used to organize and select group of objects
- **Replica Set**: manages the lifecycle of pods and ensures specified number are running

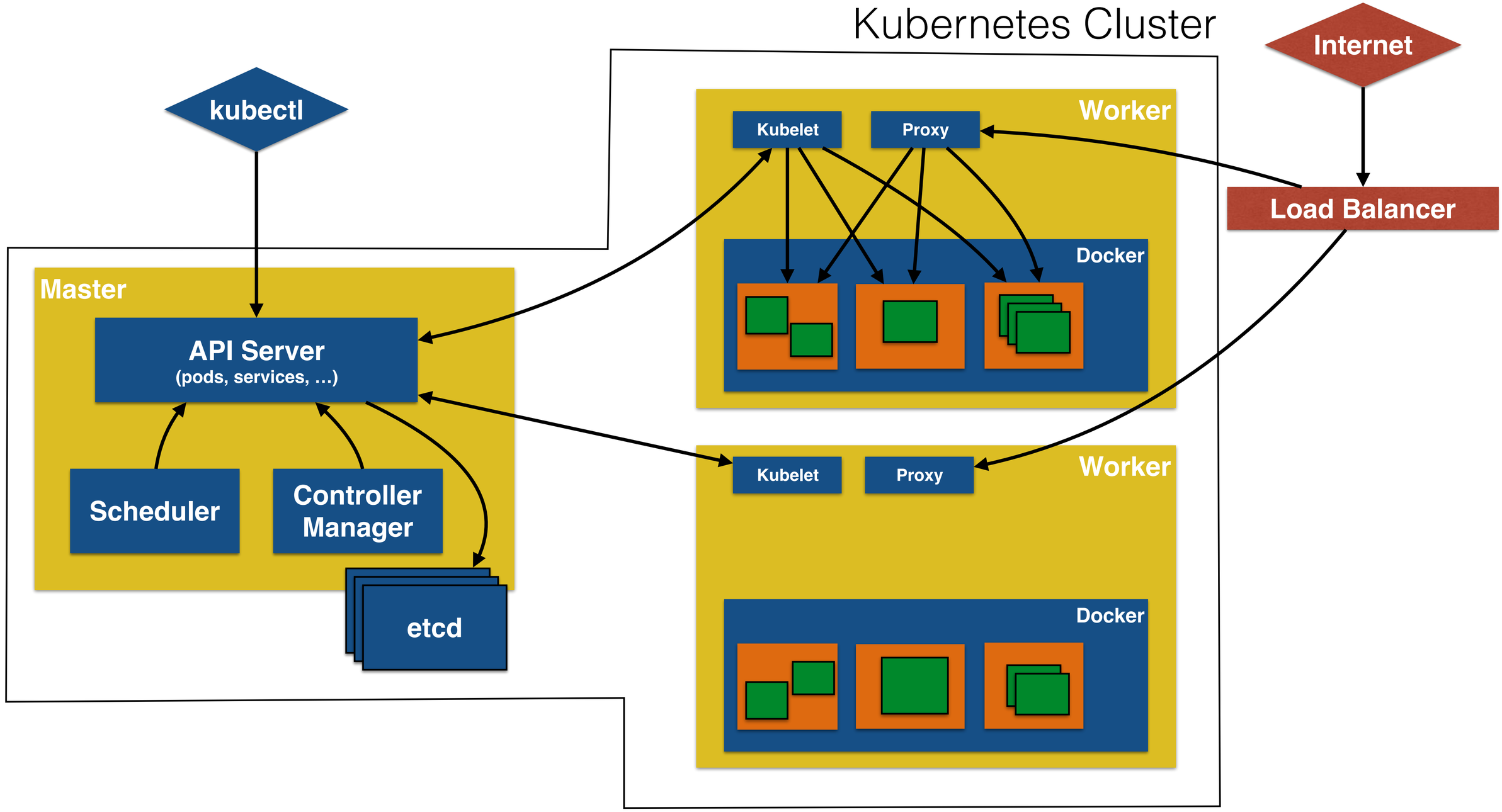




Core Concepts: Kubernetes

- **Node**: Machine or VM in the cluster
- **Master**: Central control plane, provides unified view of the cluster
  - **etcd**: distributed key-value store used to persist Kubernetes system state
- **Worker**: Docker host running *kubelet* (node agent) and *proxy* services
  - Runs pods and containers
  - Monitored by *systemd* (CentOS) or *monit* (Debian)




Kubernetes Cluster
kubectl
Internet
Load Balancer
Master
API Server
(pods, services, ...)
Scheduler
Controller Manager
etcd
Worker
Kubelet
Proxy
Docker
Worker
Kubelet
Proxy
Docker

kubectl

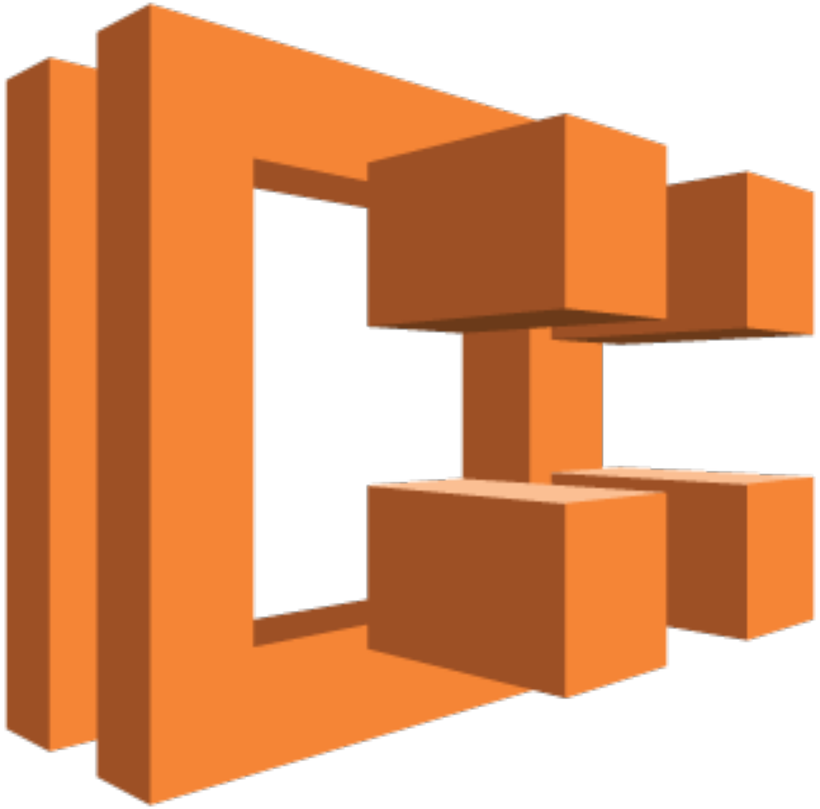
- Controls the Kubernetes cluster manager
- `kubectl get pods` or `minions`
- `kubectl create -f <filename>`
- `kubectl update` or `delete`
- `kubectl resize –replicas=3 replicaset <name>`

Kubernetes on Amazon Web Services

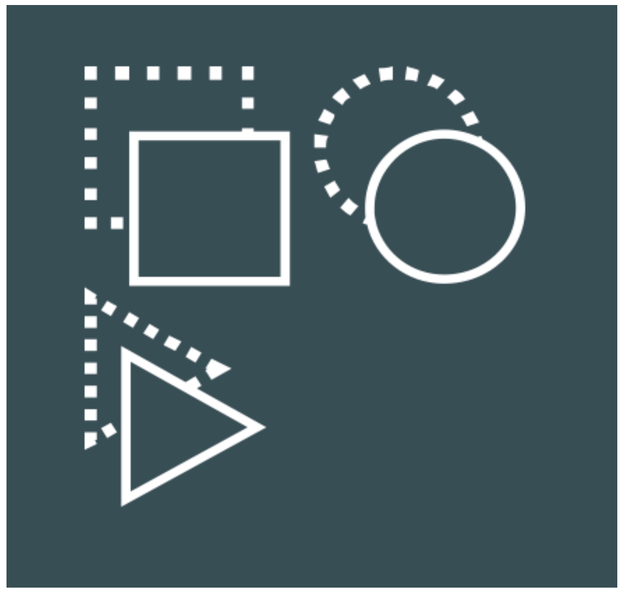
This page lists different options, in alphabetic order, of starting a Kubernetes cluster on Amazon Web Services.

- Clocker: http://www.clocker.io/tutorials/kubernetes-cluster.html
- Heptio: https://github.com/aws-quickstart/quickstart-heptio
- Juju Charms: https://jujucharms.com/canonical-kubernetes/
- Kargo: https://github.com/kubernetes-incubator/kargo
- Kismatic Enterprise Toolkit: https://github.com/apprenda/kismatic
- Kraken 2: https://github.com/samsung-cnct/k2
- kube-aws: https://github.com/kubernetes-incubator/kube-aws
- Kubeadm Quickstart: https://github.com/upmc-enterprises/kubeadm-aws
- Kubernetes Operations (kops): https://github.com/kubernetes/kops
- OpenShift: https://access.redhat.com/articles/2623521
- Stackpoint.io: https://stackpoint.io
- Tack: https://github.com/kz8s/tack
- Tectonic: http://github.com/coreos/tectonic-installer
- Weaveworks AMI: https://github.com/weaveworks/kubernetes-ami

https://github.com/kubernetes/community/blob/master/sig-aws/kubernetes-on-aws.md



Amazon EC2 Container Service

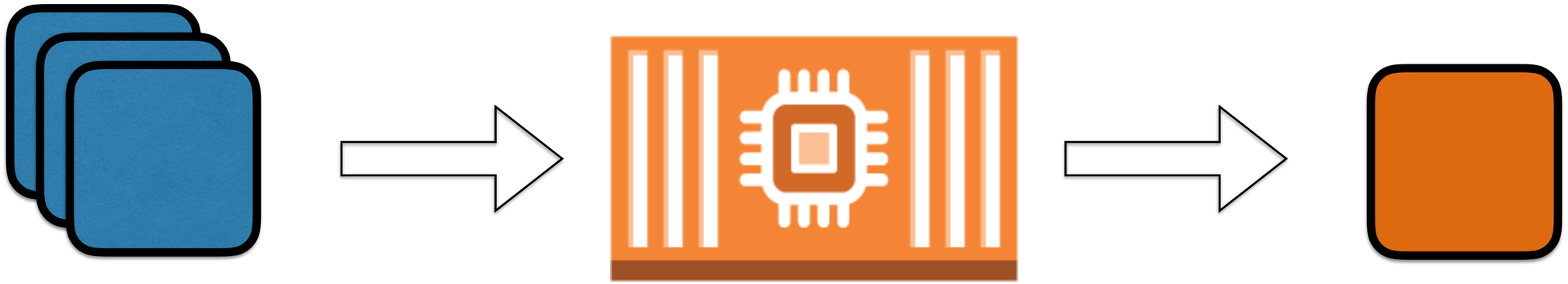


Cluster
Management

Container
Orchestration

Deep AWS
Integration

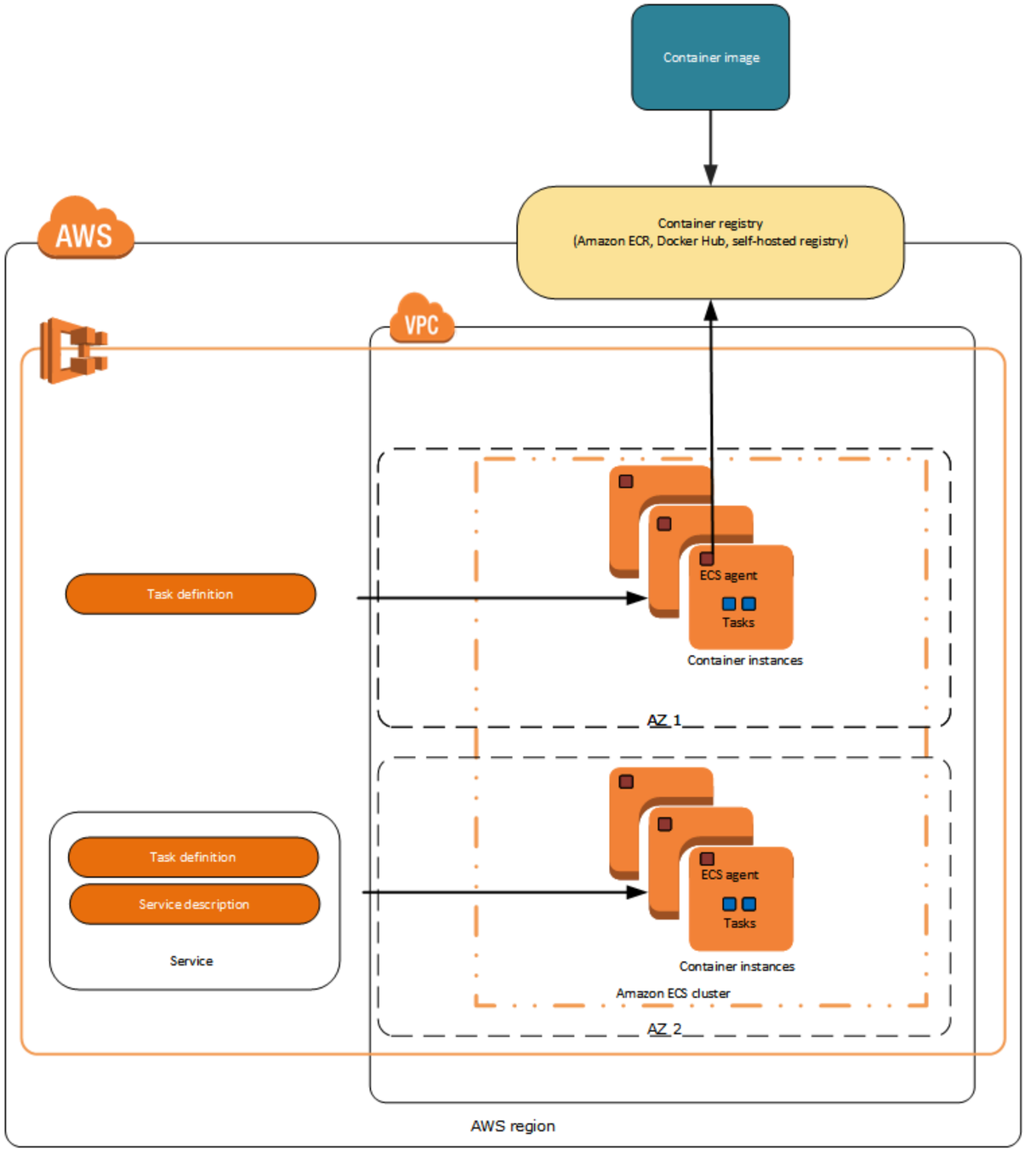
Mapping to EC2 Workloads



Instances

Services

Tasks



http://docs.aws.amazon.com/AmazonECS/latest/developerguide/

Deep Integration with AWS

- Specify IAM role used by the containers in a task
- Services deploy and scale quickly, easily extensible
  - For example, git push can trigger a deployment using CI tool
- Scale a service up or down based upon CloudWatch alarms
- Audit through CloudTrail, can track task/role association
- Application Load Balancer: Define routing rules based on content

Create a task definition

An Amazon ECS task definition is a blueprint or recipe for containers. You can modify parameters in the task definition to suit your particular application (for example, to provide more CPU resources or change the port mappings). Learn more

Task definition name* wildlfy

Container name* webapp

Image* wildfly:10.1.0.Final

Custom image format: [registry-url]/[namespace]/[image]:[tag]

Memory Limits (MB)* Hard limit 1024

Add Soft limit

Define hard and/or soft memory limits in MiB for your container. Hard and soft limits correspond to the `memory` and `memoryReservation` parameters, respectively, in task definitions.
ECS recommends 300-500 MB as a starting point for web applications.

Port mappings

Host port	*Container port*	*Protocol*
8080	8080	tcp

Add port mapping

Configure service

Create a name for your service and set the desired number of tasks to start with. A service auto-recovers any stopped tasks to maintain the desired number that you specify here. Later, you can update your service to deploy a new image or change the running number of tasks. Learn more

Service name* sample-webapp

Desired number of tasks* 2

Application Load Balancer

Create an Application Load Balancer and configure your service to run behind it. Learn more

Container name : container port : protocol No Application Load Balancer

Configure cluster

Your Amazon ECS tasks run on container instances (Amazon EC2 instances that are running the ECS container agent). Configure the instance type, instance quantity, and other details of the container instances to launch into your cluster

Cluster name* default

EC2 instance type* m3.large

Number of instances* 4

Key pair arun-west1

You will not be able to SSH into your EC2 instances without a key pair. You can create a new key pair in the EC2 console.

Security group

By default, your instances are accessible from any IP address. We recommend that you update the below security group ingress rule to allow access from known IP addresses only. ECS automatically opens up port 80 to facilitate access to the application or service you're running.

Allowed ingress source(s)* Anywhere

0.0.0.0/0

Container instance IAM role

The Amazon ECS container agent makes calls to the Amazon ECS API actions on your behalf, so container instances that run the agent require the ecsInstanceRole IAM policy and role for the service to know that the agent belongs to you. If you do not have the ecsInstanceRole already, we can create one for you.

Container instance IAM role ecsInstanceRole

Integration with other AWS services

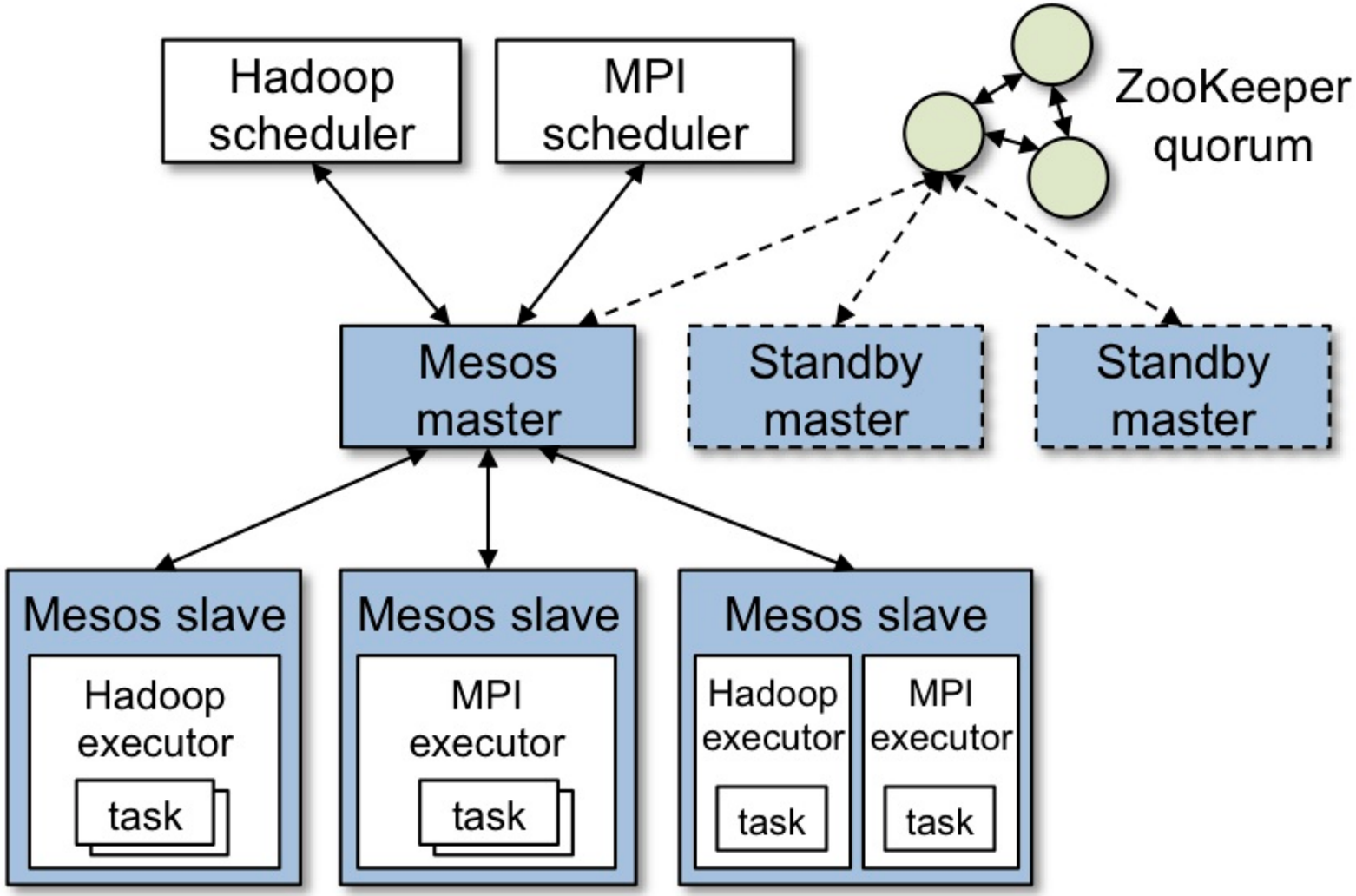
- Elastic Load Balancing
- Amazon Elastic Block Store
- Amazon Virtual Private Cloud
- Amazon CloudWatch
- AWS Identity and Access Management
- AWS CloudTrail
- AWS CodeBuild
- AWS CodePipeline

MESOSPHERE

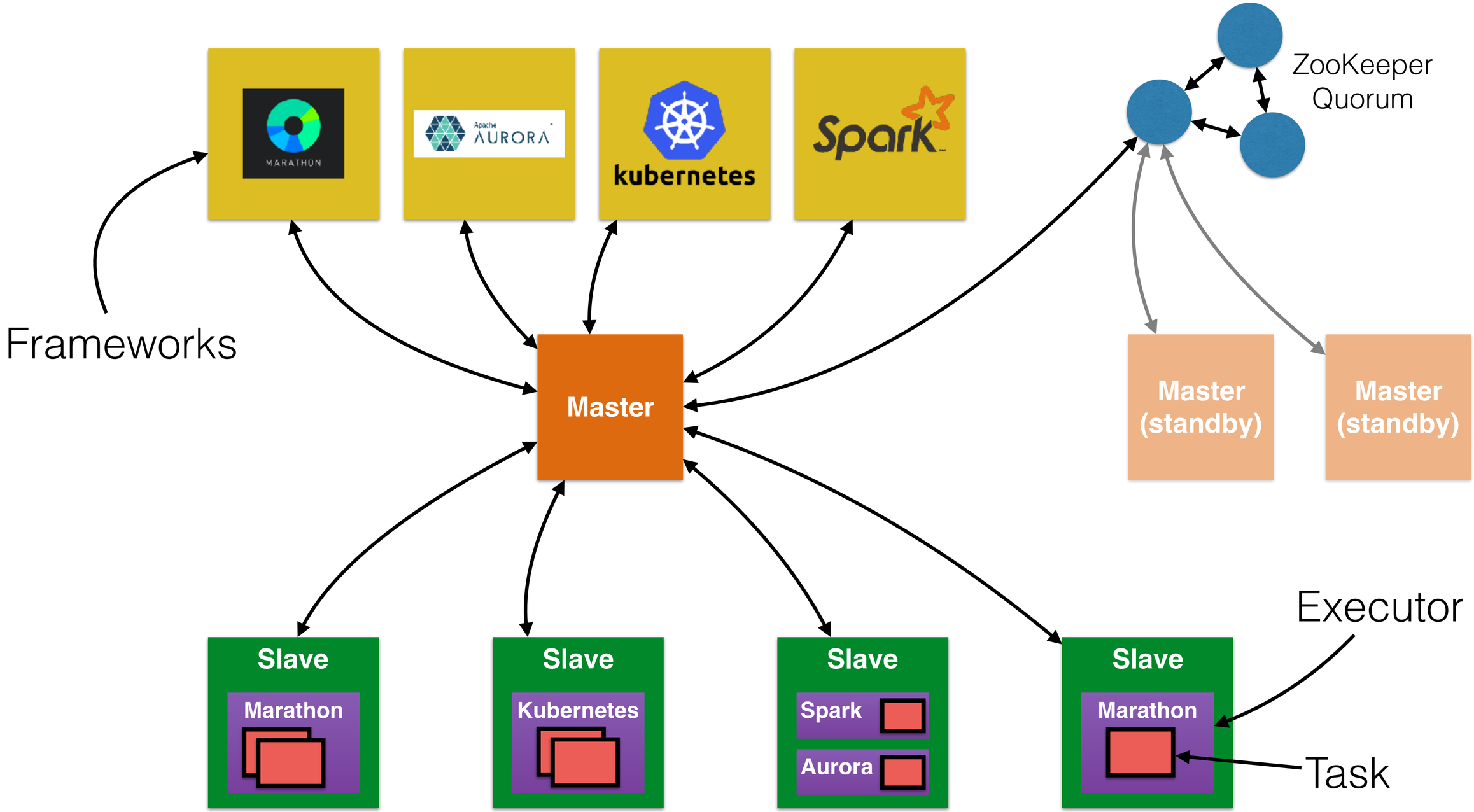
Mesos

- Open source cluster manager
  - Developed at UC Berkeley
- Provides resource isolation and sharing across distributed applications
- Run distributed systems on the same pool of nodes
  - Hadoop, Spark, Jenkins, …
- Cluster monitoring
- Tasks isolated via Linux containers

Mesos Architecture



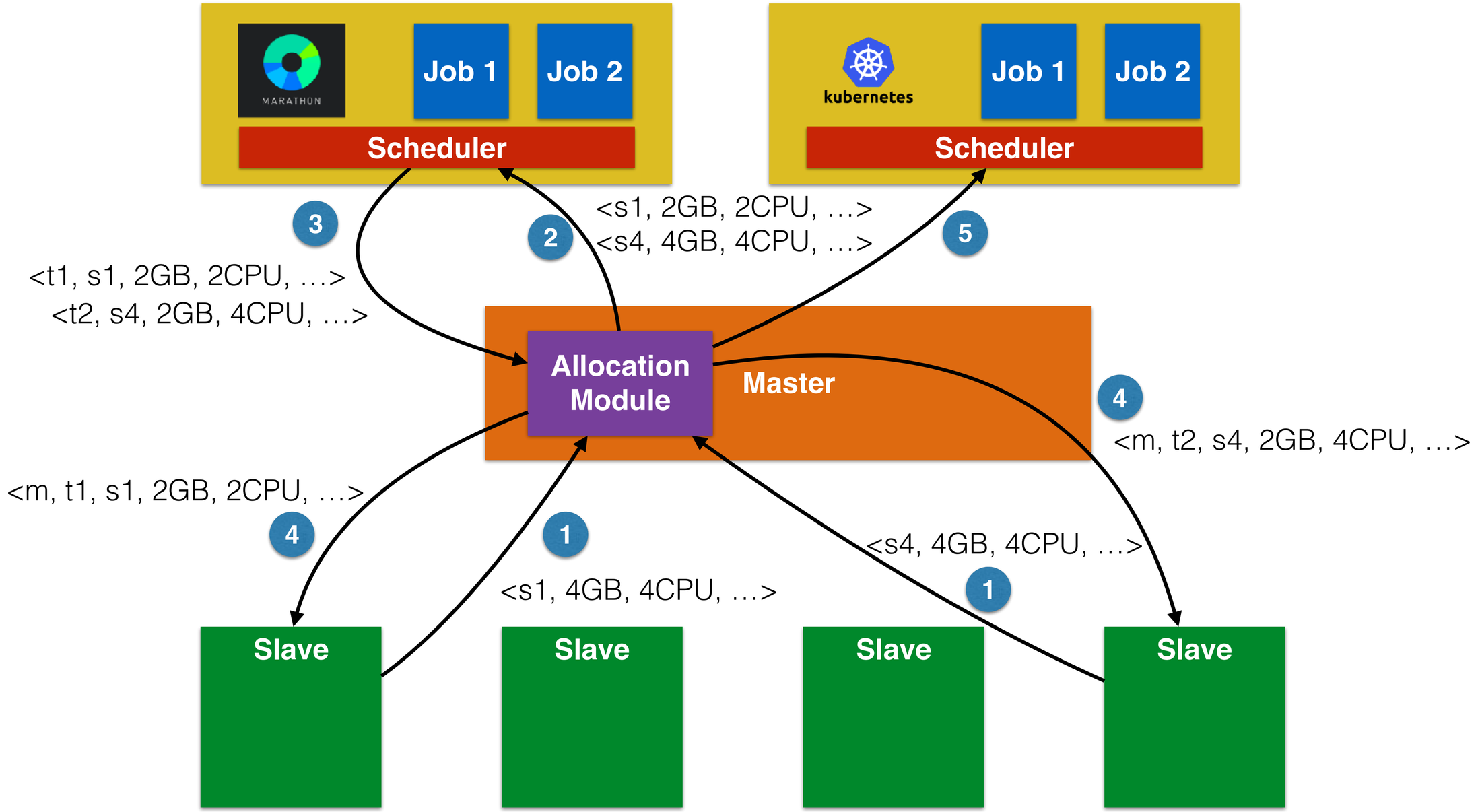
http://mesos.apache.org/documentation/latest/architecture/



http://mesos.apache.org/documentation/latest/architecture/

Frameworks

- Frameworks are targeted at a use case and domain-specific
  - Master node “offers” resources to each framework
  - Framework “accepts” the offer and execute applications
- Framework has “scheduler” and “executor”
  - Scheduler registers with the master for “offer”
  - Executor launched on slave nodes to run the task
    - Passes a description of the task to run



http://mesos.apache.org/documentation/latest/architecture/

DC/OS on AWS

- CloudFormation template
- Basic
  - Easy to get started, minimal setup required
  - Great for simple production deployment, demos and testing
  - Limited customization options
- Advanced
  - Highly customizable
  - More setup work is required

Thanks!

Arun Gupta, @arungupta
github.com/arun-gupta/docker-java/tree/master/slides