Container

* package of applications and dependencies
* shares kernel of the infra/machine its been running on
* Container is a running instance of a container image

**Container Orchestration - Kubernetes**

**Cluster**:

|  |  |
| --- | --- |
| **Control plan**(multiple or single master nodes) | Nodes- ec2 machines |

**Worker nodes**

Single node Example:

Pre requisite and Instru: in EC2(node)

[

install **container run time(cri-container run time interface - containereD)** - for Kubernetes to work with containers - **connection**

Install runc, CNI plugin

**Node agent: Kubelet** will be in all the nodes(master/worker)

* registers the node to identify itself to the cluster
* Takes instructions to start/edit containers and takes care of governance of the container health
* Kublet works with CRI to launch pods and containers

Kubeadm - to bootstrap the cluster

Kubectl - to commuicate with the cluster

]

Api version:

rbac.authorization.k8s.io/v1 - roles

**Control Plane (Master Node):** The "brains" of the cluster, responsible for maintaining the desired state.

* **kube-apiserver:** The front-end of the control plane, exposing the Kubernetes API. All communication with the cluster goes through here.
* **etcd:** A distributed, consistent, and highly available key-value store where all cluster data is stored. Think of it as Kubernetes' single source of truth.
* **kube-scheduler:** Watches for newly created Pods and assigns them to a healthy Node based on resource requirements and constraints.
* **kube-controller-manager:** Runs various controllers (e.g., ReplicationController, EndpointsController) that continuously monitor the state of the cluster and make changes to move the actual state towards the desired state.

**Worker Nodes:** The "brawn" of the cluster, where your containerized applications actually run.

* **kubelet:** An agent that runs on each Node and communicates with the Control Plane. It ensures that containers are running in a Pod.
* **kube-proxy:** A network proxy that maintains network rules on Nodes, enabling network communication to your Pods from inside or outside the cluster. It handles Service load balancing.
* **Container Runtime:** The software responsible for running containers (e.g., Docker, containerd, CRI-O). You're already familiar with Docker, so this part should feel comfortable!

**Key Kubernetes Objects (Building Blocks):**

* **Pod:** The smallest deployable unit in Kubernetes. It's an abstraction over a container, representing a single instance of a running process in your cluster. A Pod can contain one or more containers that share network and storage resources.
* **ReplicaSet:** Ensures a specified number of Pod replicas are running at any given time.
* **Deployment:** A higher-level abstraction that manages ReplicaSets, enabling declarative updates to Pods and ReplicaSets. This is how you manage your application's lifecycle, including rolling updates and rollbacks.
* **Service:** An abstract way to expose an application running on a set of Pods as a network service. It provides a stable IP address and DNS name for a set of Pods, even if the underlying Pods change.
  + **ClusterIP:** Exposes the Service on an internal IP in the cluster. Only reachable from within the cluster.
  + **NodePort:** Exposes the Service on a static port on each Node's IP. This makes the Service accessible from outside the cluster.
  + **LoadBalancer:** Exposes the Service externally using a cloud provider's load balancer, assigning a public IP. (Leverages your AWS knowledge!)
  + **ExternalName:** Maps the Service to a DNS name outside the cluster.
* **Namespace:** A way to divide cluster resources among multiple users or teams. Provides scope for names.
* **ConfigMap:** Used to store non-sensitive configuration data as key-value pairs.
* **Secret:** Similar to ConfigMap but designed for sensitive data (e.g., passwords, API keys). Kubernetes provides mechanisms to keep them secure.
* **Volume:** A directory, possibly with some data in it, that is accessible to the containers in a Pod.