Kubernetes

AGENDA

- Introduction to Kubernetes
- Advantages of Kubernetes
- Kubernetes Architecture
- Components of Kubernetes
- Kubernetes Deployment Models
- Getting Up and Running with Kubernetes

Containers

Containers and container platforms provide a lot more advantages over traditional virtualization

Isolation is done on the kernel level with need of Guest OS

Containers are much more efficient, fast, and lightweight.

Allows for apps to become encapsulated in self-contained environments comes with advantages, such as quicker deployments, scalability, and closer parity between development environments.

Docker is currently the most popular container platform.

Docker Engine, is a runtime which allows you to build and run containers.

Need for container orchestration

While Docker provided an open standard for packaging and distributing containerized applications, there arose a new problem.

1. How to bring multiple hosts together and make them cluster ?
2. How to schedule the containers to run on specific hosts ?
3. How can containers communicate to each other?
4. Who will make sure that the container has the dependent storage, when it is scheduled on a specific host ?

Solutions for orchestrating containers soon emerged and Kubenetes is most popular of all and clear winner.

Kubernetes – The Gold Standard

- Born from Google internal Project in mid 2014
- 1.0 Released in 2015
- Google partnered with the Linux foundation to form the Cloud Native Computing Foundation to offer kubernetes as open standard
- frequently called as k8s
- Kubernetes is derived from Greek – “helmsman” or “pilot”
- Kubernetes is designed to work in multiple environments, including bare metal, on-premises VMs, and public clouds.

Kubernetes only happens to have a lot of pieces that you need to put together to create something cool

What’s Kubernetes

You can think of Kubernetes as a scheduler.

Kubernetes inspects your infrastructure (bare metal or cloud, public or private) and measures CPU and memory for each computer.

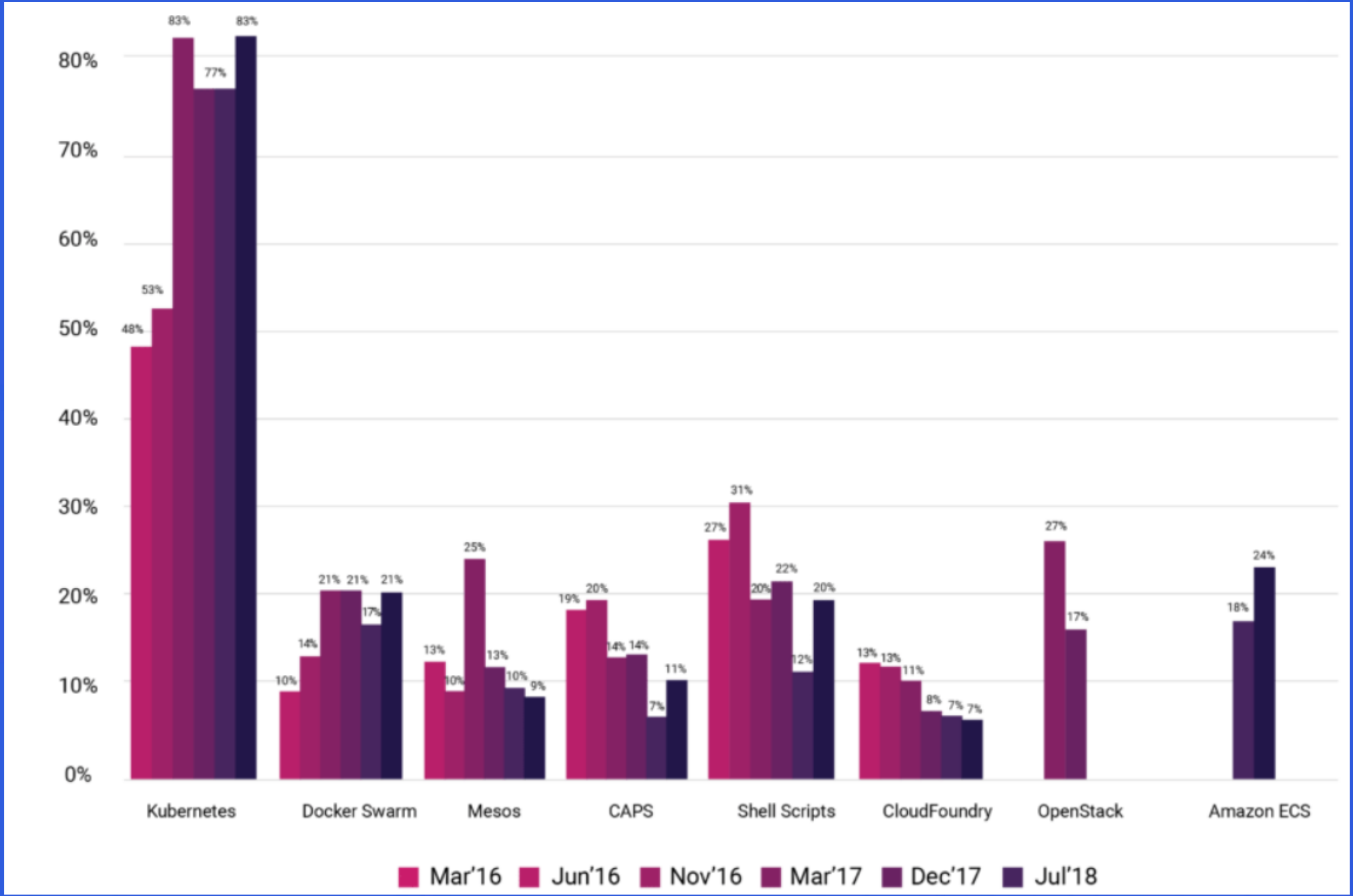
When you request to deploy a container, Kubernetes identifies the memory requirements for your container and finds the best server that satisfies your request.

You don’t decide where the application is deployed. The data centre is abstracted away from you.

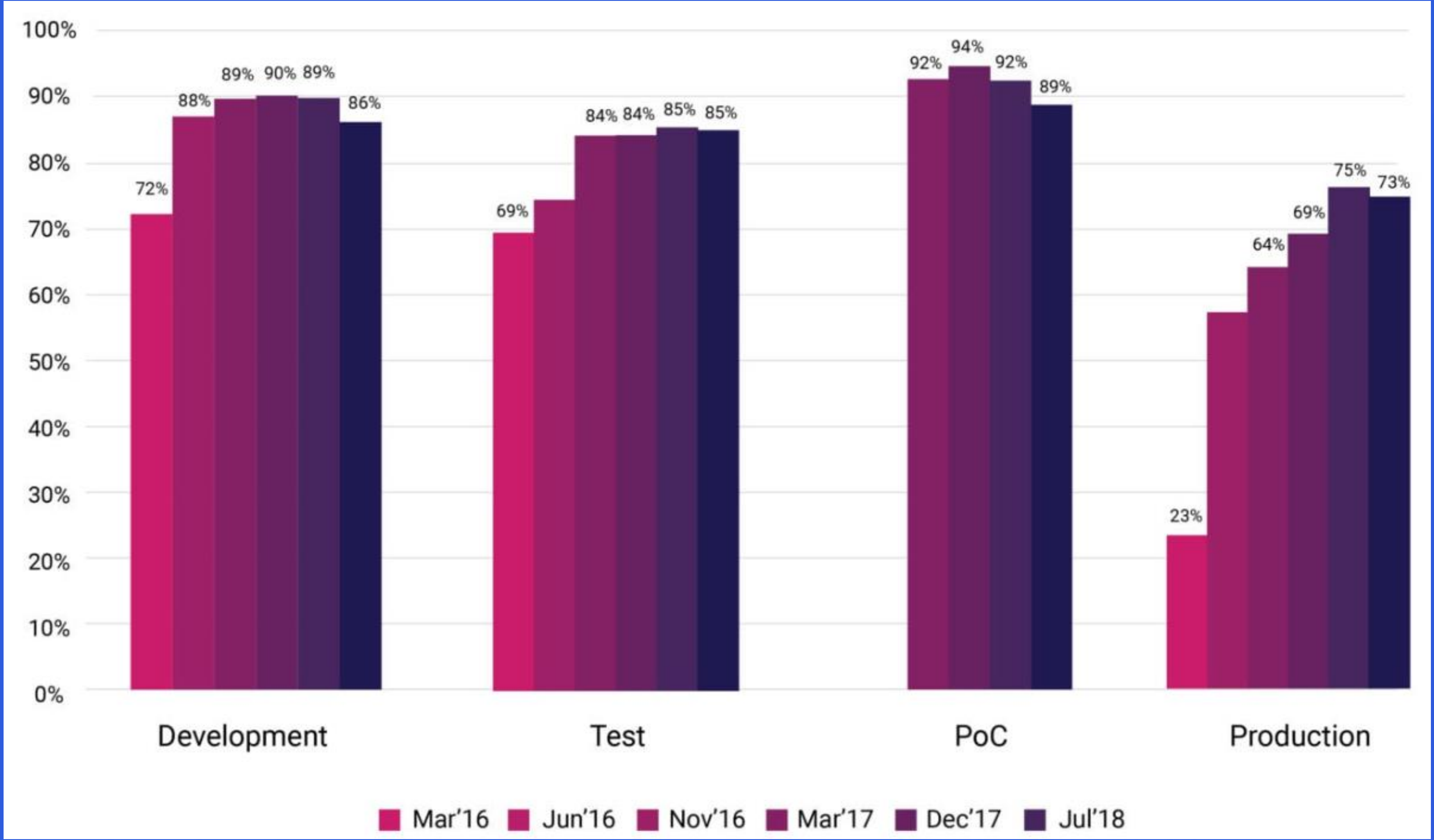
In other words, Kubernetes will play Tetris with your infrastructure.
Docker containers are the blocks; servers are the boards, and Kubernetes is the player.

Everything you do in Kubernetes is one API call away from you.

Container Orchestration Tools Usage



companies uses containers in env’s:



What does Kubernetes do

- Provisioning and deployment of containers
- Automated deployment and replication of containers across host infra
- Online scale-in or scale-out of container clusters
- Load balancing over groups of containers
- Rolling upgrades and downgrade of application containers
- Resilience, with automated rescheduling of failed containers
- Controlled exposure of services running in a container to outside world
- Service Discovery
- Horizontal Scaling, Self-Healing, Batch Processing, Secrets/Configuration, Scheduling Affinity, Storage Solutions

Kubernetes from a Developer Perspective

- Everything is an Object
- An object represents a desired state
- YAML is the way to define the state of object
- We define the Kubernetes config using these objects in YAML files
- We also can write the Kubernetes config in JSON
- Applications will continue to run if Kubernetes is down
- K8s is a Client-Server Architecture
- We need to Install Master components and Node components

Kubernetes Architecture

Master Components:

Kube-apiserver

Etcd

Kube-scheduler

Kube-controller-manager

Node Components:

Kubelet

Kube-proxy

Container runtime

Kube-apiserver

The API Server is the main management entry point to entire cluster.

In short, it processes REST operations, validates them, and updates the corresponding objects in etcd.

The API Server is the only Kubernetes component that connects to etcd;

all the other components must go through the API Server to work with the cluster state.

ETCD

Etcd is a distributed key value store that provides reliable way of storing data across a cluster of machines.

In Kubernetes, etcd reliably stores the configuration data of the Kubernetes cluster.

It representing the state of the cluster (what nodes exist in the cluster, what pods should be running, which nodes they are running on, and a whole lot more) at any given point of time

Scheduler

The scheduler’s main job is to allocate what node the pods needs to be created. It registers with api server for any newly created object/resource.

Scheduler figures out what node the pods needs to be created, using an algorithm. It checks whether the worker node has desired capacity or not. (CPU and Memory)

It checks whether the resource specification targeted any specific nodes with labels or affinity rules or any specific volumes like SSD.

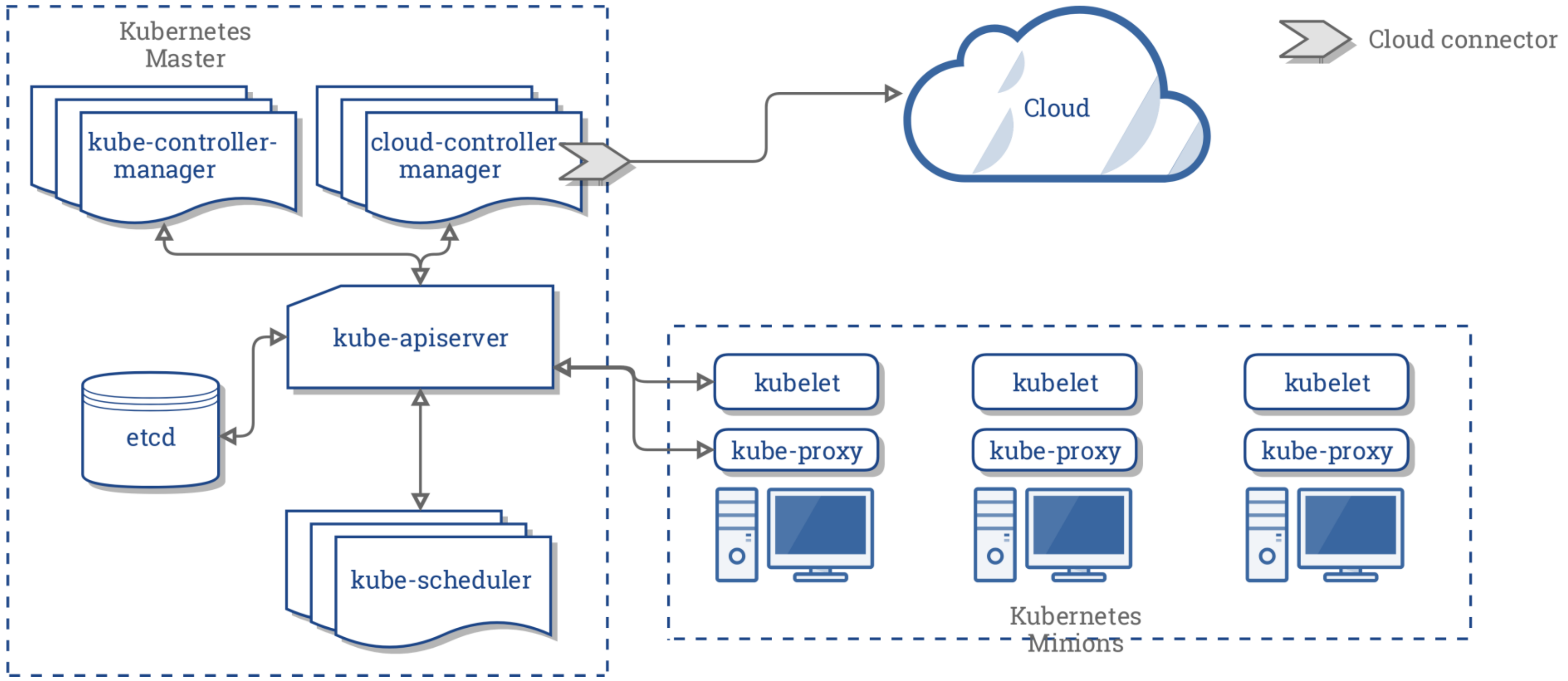
Controller Manager

Controller Manager watches the state of the cluster through the API Server watch feature and, when it gets notified, its responsible to makes the necessary changes attempting to move the current state towards the desired state.

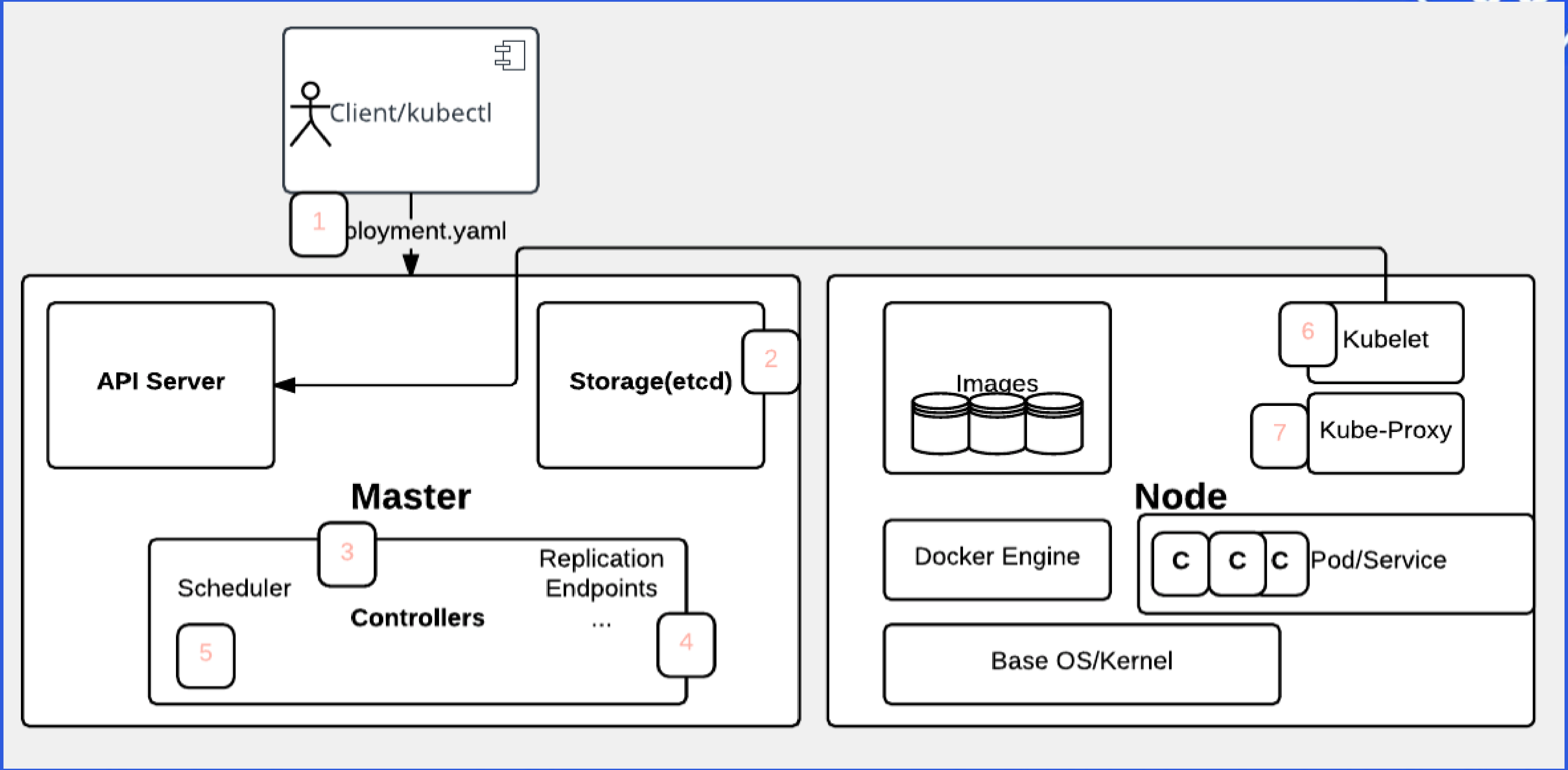
Controller Manager receives the desired state as specified in the resource specification.

All controllers watch the API Server for changes to resources/objects and perform necessary actions like create/update/delete of the resource.

Kubernetes Architecture



Kubernetes Deployment FLOW



Kubernetes Deployment FLOW

1. The user deploys a app using the kubectl CLI. Kubectl sends the request to the API server.
2. The API server receives the request and stores it in the data store (etcd). After the request is written to the data store, the API server is done with the request.
3. Watchers detect the resource changes and send notifications to the Controller to act on those changes.
4. The Controller detects the new app and creates new pods to match the desired number of instances. Any changes to the stored model will be used to create or delete pods.
5. The Scheduler assigns new pods to a node based on specific criteria. The Scheduler decides on whether to run pods on specific nodes in the cluster.
6. A Kubelet on a node detects a pod with an assignment to itself and deploys the requested containers through the container runtime, for example, Docker. Each node watches the storage to see what pods it is assigned to run. The node takes necessary actions on the resources assigned to it such as to create or delete pods.
7. Kubeproxy manages network traffic for the pods, including service discovery and load balancing. Kubeproxy is responsible for communication between pods that want to interact.

Kubernetes Node Components

Kubernetes node runs the Kubelet and Service Proxy components as well as a container engine such as Docker or Rocket, which in turns provides a container run time environment for your containerized applications

Kubelet
Kube-proxy
Container RUntime

Kubelet

The Kubelet is one of the most important components in Kubernetes.

It's an agent that runs on each node and is responsible for watching the API Server for pods that are bound to its node and making sure those pods are running (it talks to the Docker daemon using the API over the Docker socket to manipulate containers lifecycle).
It then reports back to the API Server the status of changes regarding those pods.

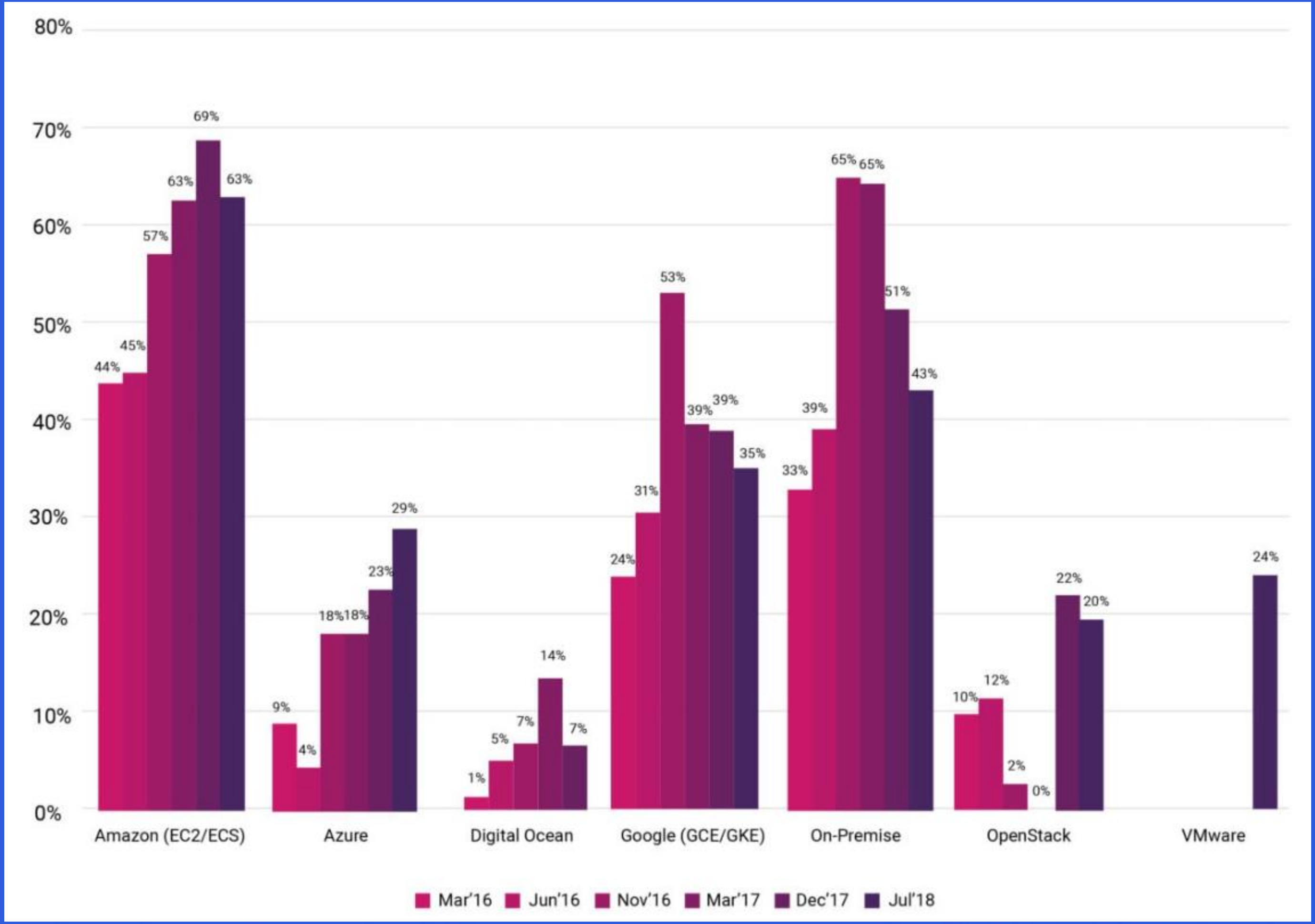
The main Kubelet responsibilities include:
1. Run the pods containers.
2. Report the status of the node and each pod to the API Server.
3. Run container probes.
4. Retrieve container metrics from cAdvisor, aggregate and expose them through the Kubelet Summary API for components (such as Heapster) to consume.

Where does Kubernetes run

Kubernetes runs any where Linux runs

- Your Laptop
- Globally distributed data centers
- Major Cloud Providers
- Any where in between and any combination

where do companies run Kubernetes



Kubernetes cloud offerings

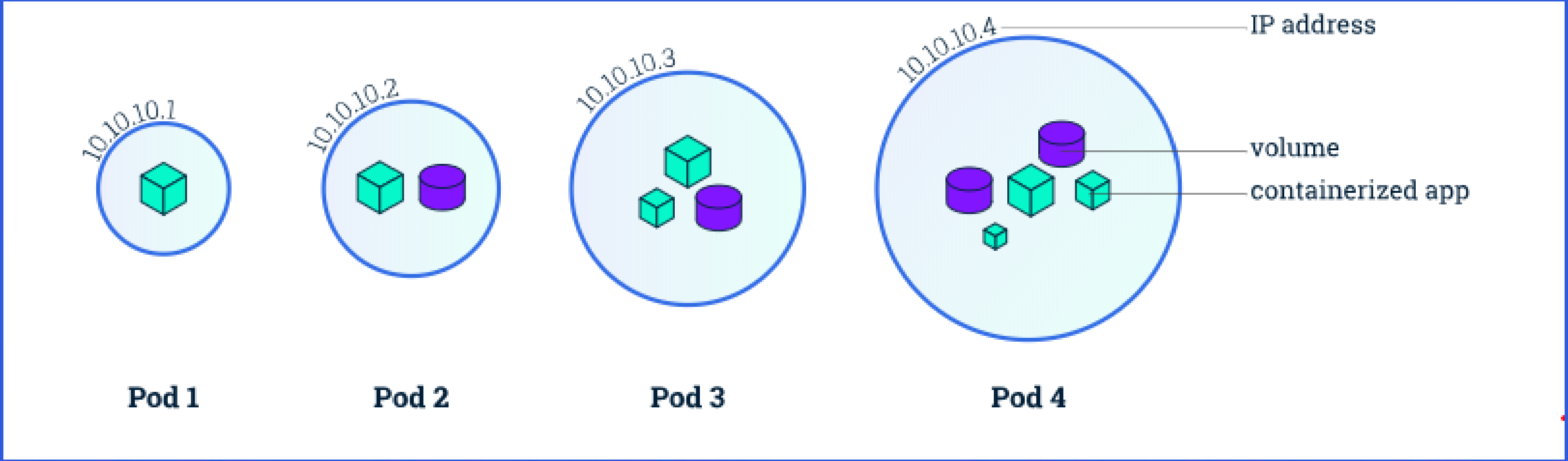
All major cloud providers are offering plenty of Kubernetes-as-a-Service offerings.
These PaaS based solutions fully abstracts the management, scaling, and security of your Kubernetes cluster, across multiple zones even, so you can focus strictly on your applications and microservices

Amazon EKS
Google Cloud Kubernetes Engine
Azure Kubernetes Service (AKS)

Kubernetes Pod

- Group of one or more containers that are always co-located, co-scheduled, and run in a shared context
- Containers in the same pod have the same hostname
- Each pod is isolated by
  - Process ID (PID) namespace
  - Network namespace
  - Inter Process Communication (IPC) namespace
  - Unix Time Sharing (UTS) namespace
- Alternative to a VM with multiple processes

Kubernetes Pod



Kubernetes Networking Model

Kubernetes aims to keep the networking architecture as simple as possible for the end user. It simplifies the cluster networking by creating a flat network structure that frees users from setting up complex network rules.

The Kubernetes platform sets the following networking rules:

1. All containers should communicate with each other without NAT.
2. All nodes should communicate with all containers without NAT.
3. The IP as seen by one container is the same as seen by the other container (in other words, Kubernetes bars any IP masquerading).
4. Pods can communicate regardless of what Node they sit on.

These are achieved using any CNI based network model

SUMMARY

- Kubernetes Master runs the API, Scheduler and Controller services
- Each Node is responsible for running one or more Pods
- Pods are the unit of deployment in Kubernetes
- Labels associate one Kubernetes object with the other
- Replication Controller ensures high availability of Pods
- Services expose Pods to internal and external consumers

Q & A

Controller Manager

Controller Manager watches the state of the cluster through the API Server watch feature and, when it gets notified, its responsible to makes the necessary changes attempting to move the current state towards the desired state.

Controller Manager receives the desired state as specified in the resource specification.

There are several different controllers available under controller manager. Some of them are DeploymentControllers, StatefulSet Controllers, Namespace Controllers, PersistentVolume Controllers etc.

All controllers watch the API Server for changes to resources/objects and perform necessary actions like create/update/delete of the resource.

Scheduler

The scheduler's main job is to allocate what node the pods needs to be created. It registers with api server for any newly created object/resource.

Scheduler figures out what node the pods needs to be created, using an algorithm. It checks whether the worker node has desired capacity or not.

It checks whether the resource specification targeted any specific nodes with labels or affinity rules or any specific volumes like SSD.

Finally after figuring out the node the scheduler will just update the resource specification and send it API Server.

The Api Server updates the resource specification and stores into etcd. The Api Server notifies the kubelet for the worker node selected by scheduler (using watch mechanism).

ETCD

etcd is a distributed key value store that provides reliable way of storing data across a cluster of machines.

In Kubernetes, etcd reliably stores the configuration data of the Kubernetes cluster, representing the state of the cluster (what nodes exist in the cluster, what pods should be running, which nodes they are running on, and a whole lot more) at any given point of time

Etcd also implements a watch feature, which provides an event-based interface for asynchronously monitoring changes to keys. Once a key is changed, its “watchers” get notified.

This is a crucial feature in the context of Kubernetes, as the API Server component heavily relies on this to get notified and call the appropriate business logic components to move the current state towards the desired state.

Kube-apiserver

The API Server is the main management entry point of the entire cluster.
In short, it processes REST operations, validates them, and updates the corresponding objects in etcd.

The API Server serves up the Kubernetes API

The API Server is the only Kubernetes component that connects to etcd; all the other components must go through the API Server to work with the cluster state.

The API Server is also responsible for the authentication and authorization mechanism. All API clients should be authenticated in order to interact with the API Server.

The API Server also implements a watch mechanism (similar to etcd) for clients to watch for changes. This allows components such as the Scheduler and Controller Manager to interact with the API Server in a loosely coupled manner.

Kubernetes basic concepts

POD
Service
Volume
Namespace

https://lagu456.site/introduction-to-kubernetes-concepts-and-architecture-with-marcus-maxwell-mp3/7EY7rEutzdY

Kubernetes advanced concepts

ReplicaSet/Replication Controller
Deployment
StatefulSet
DeamonSet
Job

https://lagu456.site/introduction-to-kubernetes-concepts-and-architecture-with-marcus-maxwell-mp3/7EY7rEutzdY

Labels & Selectors

- Key/value pairs associated with Kubernetes objects
- Used to organize and select subsets of objects
- Attached to objects at creation time but modified at any time.
- Labels are the essential glue to associate one API object with other
    - Replication Controller -> Pods
    - Service -> Pods
    - Pods -> Nodes

Services

- An abstraction to define a logical set of Pods bound by a policy to access them
- Services are exposed through internal and external endpoints
- Services can also point to non-Kubernetes endpoints through a Virtual-IP-Bridge
- Supports TCP and UDP
- Interfaces with kube-proxy to manipulate iptables
- Service can be exposed internal or external to the cluster

Replication Controller

- Ensures that a Pod or homogeneous set of Pods are always up and available
- Always maintains desired number of Pods
    - If there are excess Pods, they get killed
    - New pods are launched when they fail, get deleted, or terminated
- Creating a replication controller with a count of 1 ensures that a Pod is always available
- Replication Controller and Pods are associated through Labels

K8s - State

Stateless Applications

- Usually as a Deployment of Pod replicas accessed via a Service

Stateful Applications

- StatefulSets
  - Stable Storage
  - Stable network identifiers
  - Ordered deployment & Scaling
  - Ordered rolling Updates