**Kubernetes**

**Microservices Architecture:**

- A microservices architecture consists of a collection of small services. Each service is self-contained and should implement a single business module.

- Each service is a separate codebase, which can be managed by a small development team.

- services can be deployed independently. A team can update an existing service without rebuilding and redeploying the entire application.

- Services communicate with each other by using well-defined APIs. Internal implementation details of each service are hidden from other services.

Containers:

- Containers are an application-centric method to deliver high-performing, scalable on any infrastructure of you choice. Conatiners are best suited to deliver microservices by providing protable, isolated virtual environments for applications to run without interference from other running applications.

Container orchestration:

- Container orchestrators are tools whixh group systems together to form clusters where containers’ deployment and management is automated at scale while meeting the requirements mentioned above.

- Marathon

- Azure service Fabric

- Docker swarm

- Nomad

- Amazon Elastic Container Service (ECS)

- Kubernetes

Kubernetes as a service:

- Amazon Elastic Kubernetes Service (Amazon EKS)

- Azure Kubernetes Service (AKS)

- Google Kubernetes Engine (GKE)

**what is kubernetes?**

- kubernetes is an open-source system for automating deployment, scaling and management of containerizd applications.

- you can think of kubernetes as the **pilot** on a ship of containers.

Features of kubernetes

- **Scalability**: Kubernetes provides horizontal scaling of pods on the basis of CPU utilization. The threshold for CPU usage is configurable and kubernetes will automatically start new pods if the threshold is reached.

\* vertical scaling: means to modify the attributed resources (like CPU or RAM) of each node in the cluster (vertical Pod autoscaler)

\* Horizontal scaling: means modifying the compute resources of an existing cluster, for example by adding new nodes to it or by adding new pods by increasing the replica count of pods (Horizontal Pod autoscaler).

- self-healing: kubernetes automatically replaces and reschedules containers from failed nodes. It kills and restarts containers unresponsive to health checks, based on existing rules/policy. it also prevents traffic from being routed to unresponsive containers.

- Automated rollouts and rollbacks: kubernetes samelessly rolls out and rolls back application updates and configuration changes, constantly monitoring the application’s health to prevent any downtime.

- Secret and configuration management: Kubernetes manages sensitive data and configuration details for an application separately from the container image. Secrets consist of sensitive/confidential information passed to the application without revealing the sensitive content to the stack configuration, like github.

Kubernetes Architecture:

\* worker nodes:

- worker node provides a running environment for client applications. through containerized microservices, these applications are encapsulated in Pods.

- Pods are the smallest deployable units of computing that you can create and manage in kubernetes.

- A Pod is a group of one or more containers with shared storage and network resources.

worker node components:

- Container runtime:

+ CRI (container runtime interface)

- containerd

- CRI-O

- Node Agent (kubelet):

+ agent running on each node and communicates with the control plane components from the master node. It receives Pod definitions, primarily from the API server. and interacts with the container runtime on the node to run containers associated with the Pod. It also monitors the health and resources of Pods running containers.

+ The kubelet connects to container runtimes though a plugin based interface: the Container Runtime Interface (CRI)

- Proxy (kube-proxy):

+ network agent which runs on each node responsible for dynamic updates and maintenance of all networking rules on the node. It abstracts the details of pods networking and forwards connection requests to pods.

+ the kube-proxy is responsible for TCP, UDP and SCTP stream forwarding or round-robin forwarding across a set of Pod backends, and it implements forwarding rules defined by users through service API objects.

- Addons (DNS, Dashboard User Interface, Monitoring and loggings):

+ Addons are cluster features and functionality not yet available in kubernetes therefore implemented through 3rd-party pods and services.

- master nodes:

+ The master node provides a ruuning environment for the control plane responsible for managing the state of a kubernetes cluster, and it is the brain behind all operations inside the cluster.

+ In order to communicate with the kubernetes cluster, users send requests to the control plane via a command line interface (CLI) tool, a web User-Interface (Web UI) Dashboard or application Programming interface (API)

+ to persist the kubernetes cluster’s state all cluster configuration data is saved to etcd