ECE 473/573
Cloud Computing and Cloud Native Systems
Lecture 17 Kubernetes

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

October 21, 2024

Outline

Kubernetes

Reading Assignment

- This lecture: Kubernetes
  `https://kubernetes.io/docs/concepts/`
- Next lecture: 8

Outline

Kubernetes

Kubernetes (K8s)

- An open-source container orchestration platform.
  - Developed by Google, open-sourced in 2014, now maintained by the CNCF.
  - For containerized workloads and services.
  - As a combination of Google's experience like Borg and practices from the community.
- Automate container deployment, scaling, and management.
  - With a growing ecosystem and a lot of services, support, and tools.

Features

- Service discovery and load balancing
  - Access containers with DNS name or IP address.
  - K8s redirects network traffic from containers with high loads.
- Storage orchestration
  - Support many storage options like local and cloud storage.
- Automated rollouts and rollbacks
  - Control how containers are updated for newer versions.
- Automatic bin packing
  - Improve resource utilization with predefined CPU and memory requests and limits.
- Self-healing
  - Restart and replace containers when they fail.

Features (Cont.)

- Secret and configuration management
    - Use best practices to manage and distribute sensitive information like passwords and API tokens.
- Batch execution
    - Manage batch processing works, as well as continuous integration (CI) works for development and testing.
- Horizontal scaling
    - Allow applications to adjust to dynamic loads by using more or less containers, automatically or through a UI.
- IPv4/IPv6 dual-stack
- Designed for extensibility

Architecture

- Nodes: worker machines where containers run.
- Pod: unit of application workload.
  - Consist of one or more application containers.
  - Run on the same node to meet storage, communication, and scheduling requirements.
- kubelet: an agent runs on each node.
  - Make sure containers are runing and healthy in Pods.
- kube-proxy: a network proxy runs on each node.
  - Maintain network rules on nodes.
  - Control network traffic between Pods and outside.
- Container runtime: manage actual containers.
  - e.g. Docker

Architecture (cont.)

- Control plane: components managing nodes and pods.
  - Distributed for fault-tolerance and high availability (HA).
- kube-apiserver: expose the Kubernetes API.
  - Horizontally scalable.
- etcd: consistent and highly-available key value store.
  - Kubernetes' backing store for all cluster data.
- kube-scheduler: resource manager for Pods.
  - Decide where newly created Pods run.
  - Subject to various resource requirements.
- kube-controller-manager: manage nodes and jobs.
- cloud-controller-manager: interface with cloud providers.

Networking and Services

- Each Pod has its own unique cluster-wide IP address.
  - A network setup like VMs and physical servers where Pods on different nodes can communicate with each other directly.
  - Without the need to map container ports to host ports.
  - Since containers in a Pod now share the same IP address, they should coordinate port usage to avoid conflictions.
- Service: an abstraction to expose a networked service.
  - Make Pods of the service available for clients to interact.
  - Without knowing numbers of names of Pods – Pods are ephemeral and are neither reliable nor durable.

A Service Example

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  labels:
    app: nginx
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

- What Pods does this Service consist of?
  - Pods cannot be reliably identified by names.
  - Instead, Pods are labeled by key-value pairs when defined.
  - This Service consists of all Pods with the label app:nginx as indicated by `selector`.
- K8s assigns this Service an IP address, named the cluster IP.
  - Stable as Pods are created and destroyed.
  - The Service is available at TCP `port` 80 from the cluster IP, and all traffics are forwarded to `targetPort` 80 on Pods.

Service Types

- ClusterIP: default type for Services
  - Like our example, these Services are only reachable from within the cluster.
  - Web/RESTful Services can be exposed to the public internet using Ingress or Gateway that support complex HTTP routing rules and HTTPs connections.
- NodePort: expose the Service on each node at a static port.
- LoadBalancer: expose the Service to external load balancer.
- ExternalName: integrate external services via DNS names.
  - This is different than the above three as the service doesn't run in the cluster and there is no Pods.

Workloads

- A workload is an application running on K8s.
  - Consist of Pods of containers.
- Workload resources define and manage how many of what pods should be running.
  - Make it possible to automatically restart and replace Pods when some fail.
- Workload resource types
  - ReplicaSet: for stateless Pods that are interchangeable.
  - Deployment: manage different versions of ReplicaSet.
  - StatefulSets: Pods with a persistent identifier for uniqueness and ordering, e.g. to access a persistent storage.
  - DaemonSet: ensure Pods to run on all nodes.
  - Job and CronJob: ensure Pods to terminate successfully possibly on a schedule, good for batch processing and CI.

A Deployment Example

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

- A Deployment with 3 replicas of nginx web servers.
- The label app:nginx allows the nginx service to find them.

ConfigMaps and Secrets

- ConfigMap stores non-confidential data in key-value pairs.
  - Available to Pods as environment variables, command-line arguments, configuration files, or via K8s API.
  - Help to decouple configuration from container images.
- Secret stores a small amount of sensitive data.
  - E.g. a password, a token, or a key – anything that you should not commit and push to a Git repository.
  - Secrets are only sent to Pods when necessary so they are less likely to be exposed.
  - K8s takes additional care to protect secrets for storage and during transmission.
  - Authentication and authorization need to be setup properly for a K8s cluster to ensure the security of the secrets.

Summary

- Applications and services in K8s are organized as Pods of containers running on nodes.
- Pods are usually organized into Deployments and StatefulSets, which makes it possible for K8s to manage their health and restart them as needed automatically.
- Pods are created and destroyed dynamically so we use labels to identify them and define Services to access them.
- There are a lot of K8s features we haven't covered today and won't be able to cover for our projects. Many online resources are available for you to explore further.