Architecture of Enterprise Applications 23
Virtualization & Container

Haopeng Chen

***RE**liable, **IN**telligent and **S**calable Systems Group (**REINS**)*
Shanghai Jiao Tong University
Shanghai, China
http://reins.se.sjtu.edu.cn/~chenhp
e-mail: chen-hp@sjtu.edu.cn

Contents and Objectives

REliable, INtelligent & Scalable Systems

- Kubernetes
  - Basic Concepts
  - Minikube

- Objectives
  - 能够根据系统容器化部署的需求，了解K8S的基本原理

Kubernetes

REliable, INtelligent & Scalable Systems

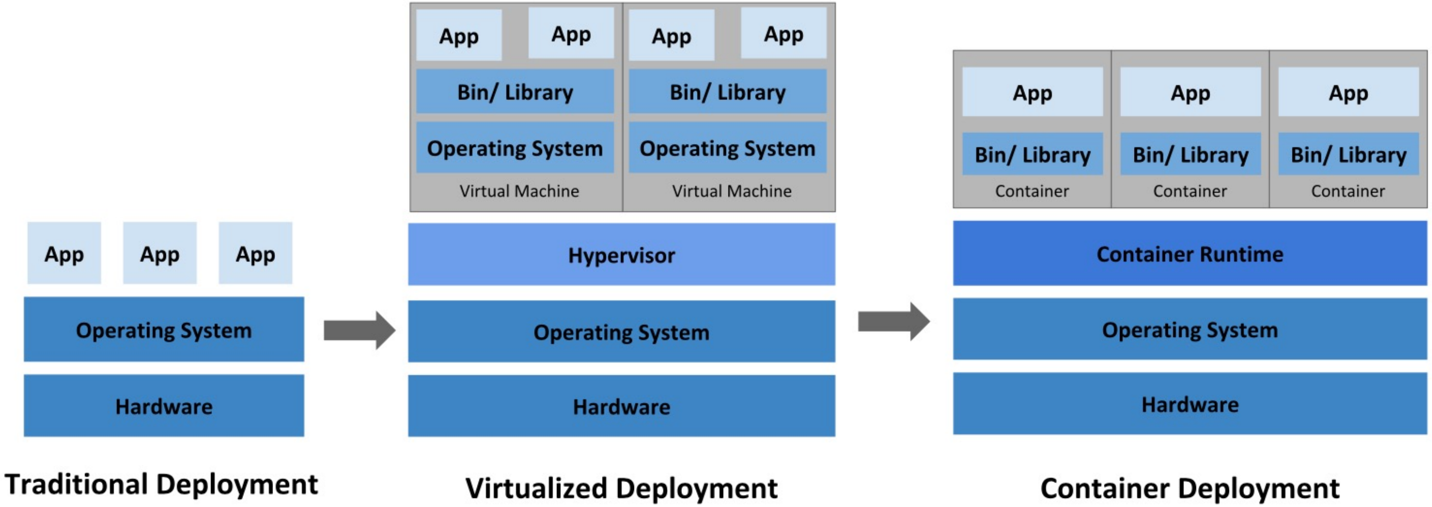
- **What is Kubernetes?**
  - https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/
  - Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.
  - It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.
  - The name Kubernetes originates from Greek, meaning helmsman or pilot.
  - Google open-sourced the Kubernetes project in 2014.
    - Kubernetes combines over 15 years of Google's experience running production workloads at scale with best-of-breed ideas and practices from the community.

kubernetes

Kubernetes

REliable, INtelligent & Scalable Systems

- **What is Kubernetes?**
  - Containers are a good way to bundle and run your applications. In a production environment, you need to manage the containers that run the applications and ensure that there is no downtime.
    - For example, if a container goes down, another container needs to start. Wouldn't it be easier if this behavior was handled by a system?

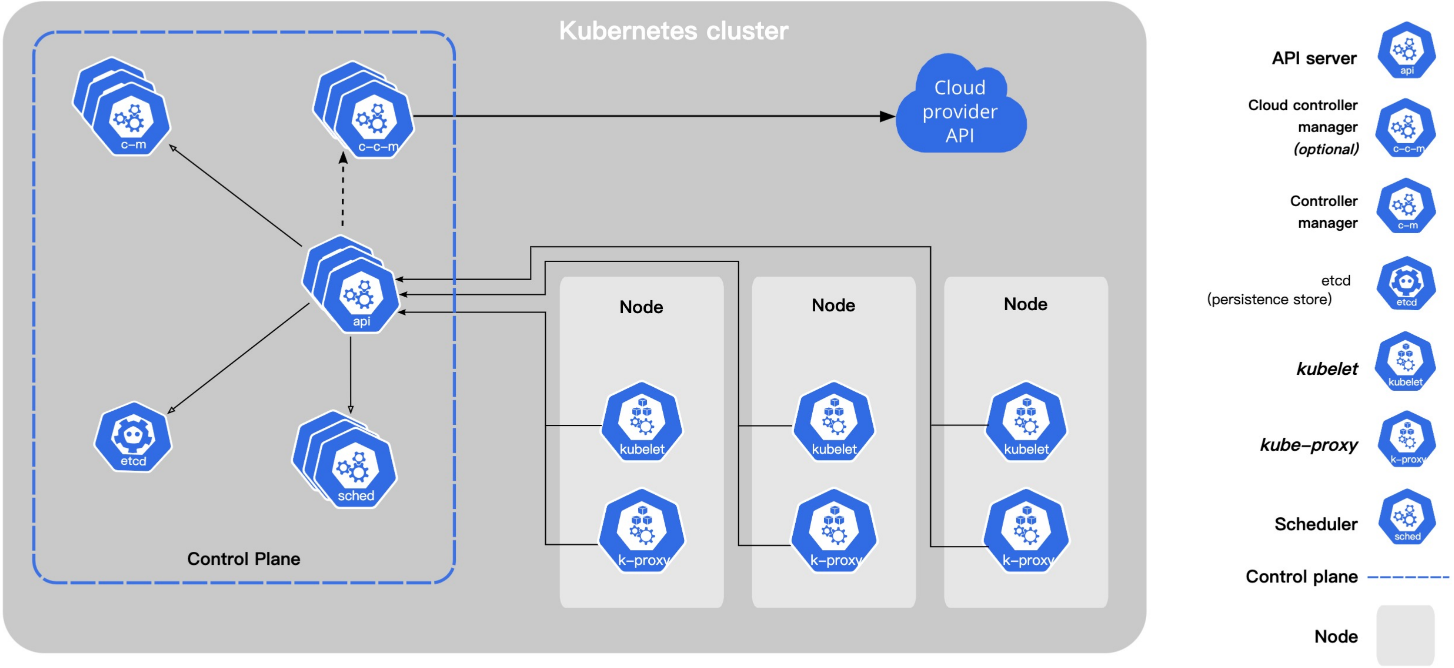


Kubernetes

- Kubernetes provides you with:
  - Service discovery and load balancing.
  - Storage orchestration.
  - Automated rollouts and rollbacks.
  - Automatic bin packing.
  - Self-healing .
  - Secret and configuration management.

Kubernetes Component

- When you deploy Kubernetes, you get a cluster.
  - A Kubernetes cluster consists of a set of worker machines, called nodes, that run containerized applications.
  - Every cluster has at least one worker node.



Control Plane’s Components

- The control plane's components
  - make global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events (for example, starting up a new pod when a deployment's replicas field is unsatisfied).
  - Control plane components can be run on any machine in the cluster.
  - However, for simplicity, set up scripts typically start all control plane components on the same machine, and do not run user containers on this machine.

- kube-apiserver
  - The API server is a component of the Kubernetes control plane that exposes the Kubernetes API. The API server is the front end for the Kubernetes control plane.
  - The main implementation of a Kubernetes API server is kube-apiserver. kube-apiserver is designed to scale horizontally—that is, it scales by deploying more instances. You can run several instances of kube-apiserver and balance traffic between those instances.

Control Plane's Components

REliable, INtelligent & Scalable Systems

- etcd
  - Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.
  - If your Kubernetes cluster uses etcd as its backing store, make sure you have a back up plan for those data.

- kube-scheduler
  - Control plane component that watches for newly created Pods with no assigned node, and selects a node for them to run on.
  - Factors taken into account for scheduling decisions include: individual and collective resource requirements, hardware/software/policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference, and deadlines.

Control Plane's Components

REliable, INtelligent & Scalable Systems

- kube-controller-manager
  - Control plane component that runs controller processes.
  - Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.
  - Some types of these controllers are:
    - Node controller: Responsible for noticing and responding when nodes go down.
    - Job controller: Watches for Job objects that represent one-off tasks, then creates Pods to run those tasks to completion.
    - Endpoints controller: Populates the Endpoints object (that is, joins Services & Pods).
    - Service Account & Token controllers: Create default accounts and API access tokens for new namespaces.

Control Plane's Components

- cloud-controller-manager
  - A Kubernetes control plane component that embeds cloud-specific control logic.
  - The cloud controller manager lets you link your cluster into your cloud provider's API, and separates out the components that interact with that cloud platform from components that only interact with your cluster.
  - The cloud-controller-manager only runs controllers that are specific to your cloud provider.
    - If you are running Kubernetes on your own premises, or in a learning environment inside your own PC, the cluster does not have a cloud controller manager.
  - As with the kube-controller-manager, the cloud-controller-manager combines several logically independent control loops into a single binary that you run as a single process.
    - You can scale horizontally (run more than one copy) to improve performance or to help tolerate failures.
  - The following controllers can have cloud provider dependencies:
    - Node controller: For checking the cloud provider to determine if a node has been deleted in the cloud after it stops responding
    - Route controller: For setting up routes in the underlying cloud infrastructure
    - Service controller: For creating, updating and deleting cloud provider load balancers

Node Components

- Node components
  - run on every node, maintaining running pods and providing the Kubernetes runtime environment.
- kubelet
  - An agent that runs on each node in the cluster.
  - It makes sure that containers are running in a Pod.
  - The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy.
  - The kubelet doesn't manage containers which were not created by Kubernetes.

Node Components

- kube-proxy
  - kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept.
  - kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.
  - kube-proxy uses the operating system packet filtering layer if there is one and it's available. Otherwise, kube-proxy forwards the traffic itself.

- Container runtime
  - The container runtime is the software that is responsible for running containers.
  - Kubernetes supports several container runtimes: Docker, containerd, CRI-O, and any implementation of the Kubernetes CRI (Container Runtime Interface).

Workload

REliable, INtelligent & Scalable Systems

- A workload is an application running on Kubernetes.
  - Whether your workload is a single component or several that work together, on Kubernetes you run it inside a set of *pods*.
  - In Kubernetes, a Pod represents a set of running containers on your cluster.
- Kubernetes pods have a defined lifecycle.
  - For example, once a pod is running in your cluster then a critical fault on the node where that pod is running means that all the pods on that node fail.
  - Kubernetes treats that level of failure as final: you would need to create a new Pod to recover, even if the node later becomes healthy.
- However, to make life considerably easier, you don't need to manage each Pod directly.
  - Instead, you can use *workload resources* that manage a set of pods on your behalf.
  - These resources configure controllers that make sure the right number of the right kind of pod are running, to match the state you specified.

Workload

- Kubernetes provides several built-in workload resources:
  - Deployment and ReplicaSet (replacing the legacy resource ReplicationController).
    - Deployment is a good fit for managing a stateless application workload on your cluster, where any Pod in the Deployment is interchangeable and can be replaced if needed.
  - StatefulSet lets you run one or more related Pods that do track state somehow.
    - For example, if your workload records data persistently, you can run a StatefulSet that matches each Pod with a PersistentVolume. Your code, running in the Pods for that StatefulSet, can replicate data to other Pods in the same StatefulSet to improve overall resilience.
  - DaemonSet defines Pods that provide node-local facilities.
    - These might be fundamental to the operation of your cluster, such as a networking helper tool, or be part of an add-on. Every time you add a node to your cluster that matches the specification in a DaemonSet, the control plane schedules a Pod for that DaemonSet onto the new node.
  - Job and CronJob define tasks that run to completion and then stop.
    - Jobs represent one-off tasks, whereas CronJobs recur according to a schedule.

Pods

- *Pods* are the smallest deployable units of computing that you can create and manage in Kubernetes.
  - A *Pod* (as in a pod of whales or pea pod) is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers.
  - A Pod's contents are always co-located and co-scheduled, and run in a shared context.
  - A Pod models an application-specific "logical host": it contains one or more application containers which are relatively tightly coupled.
  - In non-cloud contexts, applications executed on the same physical or virtual machine are analogous to cloud applications executed on the same logical host.
- As well as application containers, a Pod can contain init containers that run during Pod startup.
  - You can also inject ephemeral containers for debugging if your cluster offers this.

Deployment

REliable, INtelligent & Scalable Systems

- A *Deployment* provides declarative updates for Pods and ReplicaSets.
  - You describe a *desired state* in a Deployment, and the Deployment Controller changes the actual state to the desired state at a controlled rate.

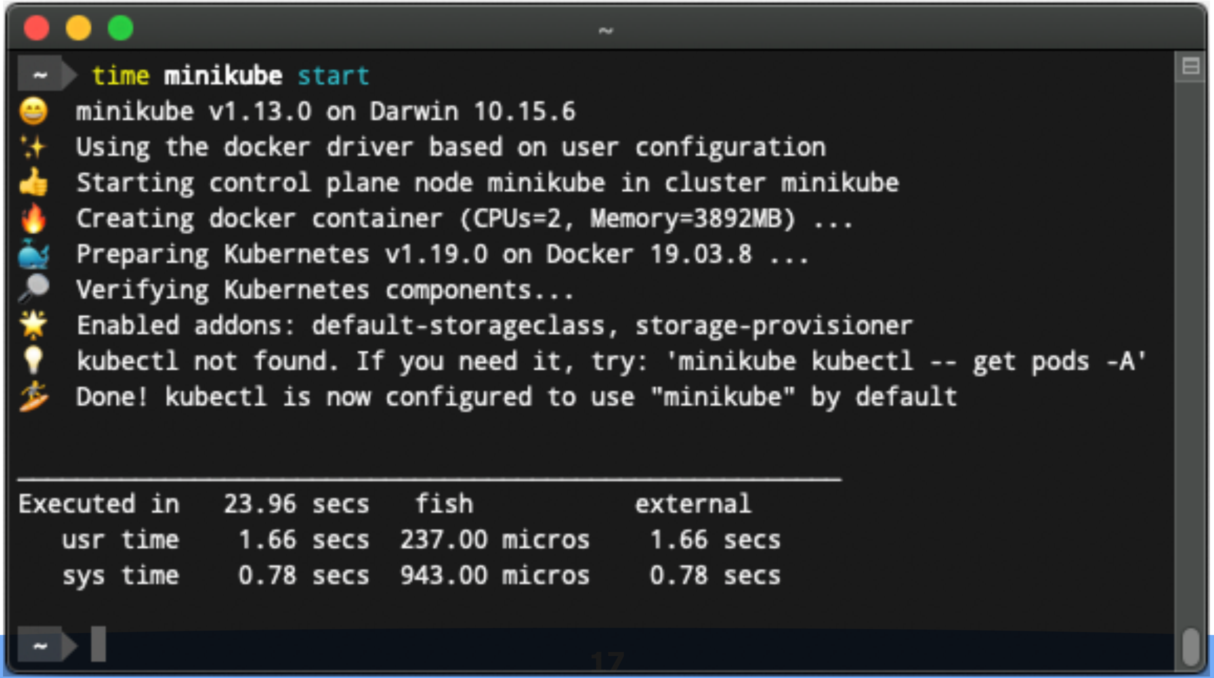
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2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

controllers/nginx-deployment.yaml

- A Deployment named nginx-deployment is created, indicated by the .metadata.name field.
- The Deployment creates three replicated Pods, indicated by the .spec.replicas field.
- The .spec.selector field defines how the Deployment finds which Pods to manage. In this case, you select a label that is defined in the Pod template (app: nginx).
- The template field contains the following sub-fields:
  - The Pods are labeled app: nginx using the .metadata.labels field.
  - The Pod template's specification, or .template.spec field, indicates that the Pods run one container, nginx, which runs the nginx Docker Hub image at version 1.14.2.
  - Create one container and name it nginx using the .spec.template.spec.containers[0].name field.

minikube

- minikube quickly sets up a local Kubernetes cluster on macOS, Linux, and Windows. We proudly focus on helping application developers and new Kubernetes users.
- https://minikube.sigs.k8s.io/docs/



Installation

Operating system: Linux | macOS | Windows

Architecture: x86-64 | ARM64

Release type: Stable | Beta

Installer type: Binary download | Homebrew

To install the latest minikube stable release on x86-64 macOS using binary download:

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-darwin-amd64
sudo install minikube-darwin-amd64 /usr/local/bin/minikube
```

minikube

REliable, INtelligent & Scalable Systems

- Start your cluster
  - From a terminal with administrator access (but not logged in as root), run:
  - $ minikube start

- Interact with your cluster
  - If you already have kubectl installed, you can now use it to access your shiny new cluster:
  - $ kubectl get po -A
  - Alternatively, minikube can download the appropriate version of kubectl and you should be able to use it like this:
  - $ minikube kubectl -- get po -A
  - Initially, some services such as the storage-provisioner, may not yet be in a Running state.
  - For additional insight into your cluster state, minikube bundles the Kubernetes Dashboard, allowing you to get easily acclimated to your new environment:
  - $ minikube dashboard

Minikube - Deploy applications

- Create a sample deployment and expose it on port 8080:
  - kubectl create deployment hello-minikube --image=k8s.gcr.io/echoserver:1.4
  - kubectl expose deployment hello-minikube --type=NodePort --port=8080
- It may take a moment, but your deployment will soon show up when you run:
  - kubectl get services hello-minikube
- The easiest way to access this service is to let minikube launch a web browser for you:
  - minikube service hello-minikube
- Alternatively, use kubectl to forward the port:
  - kubectl port-forward service/hello-minikube 7080:8080
- Tada! Your application is now available at http://localhost:7080/.

Minikube - LoadBalancer deployments

REliable, INtelligent & Scalable Systems

- To access a LoadBalancer deployment, use the "minikube tunnel" command. Here is an example deployment:
  - kubectl create deployment balanced --image=k8s.gcr.io/echoserver:1.4
  - kubectl expose deployment balanced --type=LoadBalancer --port=8080
- In another window, start the tunnel to create a routable IP for the 'balanced' deployment:
  - minikube tunnel
- To find the routable IP, run this command and examine the EXTERNAL-IP column:
  - kubectl get services balanced
- Your deployment is now available at <EXTERNAL-IP>:8080

Minikube - Manage your cluster

- Pause Kubernetes without impacting deployed applications:
  - minikube pause
- Unpause a paused instance:
  - minikube unpause
- Halt the cluster:
  - minikube stop
- Increase the default memory limit (requires a restart):
  - minikube config set memory 16384
- Browse the catalog of easily installed Kubernetes services:
  - minikube addons list
- Create a second cluster running an older Kubernetes release:
  - minikube start -p aged --kubernetes-version=v1.16.1
- Delete all of the minikube clusters:
  - minikube delete --all

References

- Kubernetes Documentation
  - https://kubernetes.io/docs/home/
- Kubernetes中文手册
  - https://www.kubernetes.org.cn/docs
  - http://docs.kubernetes.org.cn/227.html
- 推荐一款Kubernetes神器“minikube”
  - https://zhuanlan.zhihu.com/p/112755080
- External IP is pending——ambassador学习
  - https://blog.csdn.net/TTT12137/article/details/116989159

Thank You!