# Kubernetes kubectl and Deployment

.NET CORE

*The kubectl command line tool lets you control Kubernetes clusters.*

HTTPS://KUBERNETES.IO/DOCS/REFERENCE/KUBECTL/OVERVIEW/

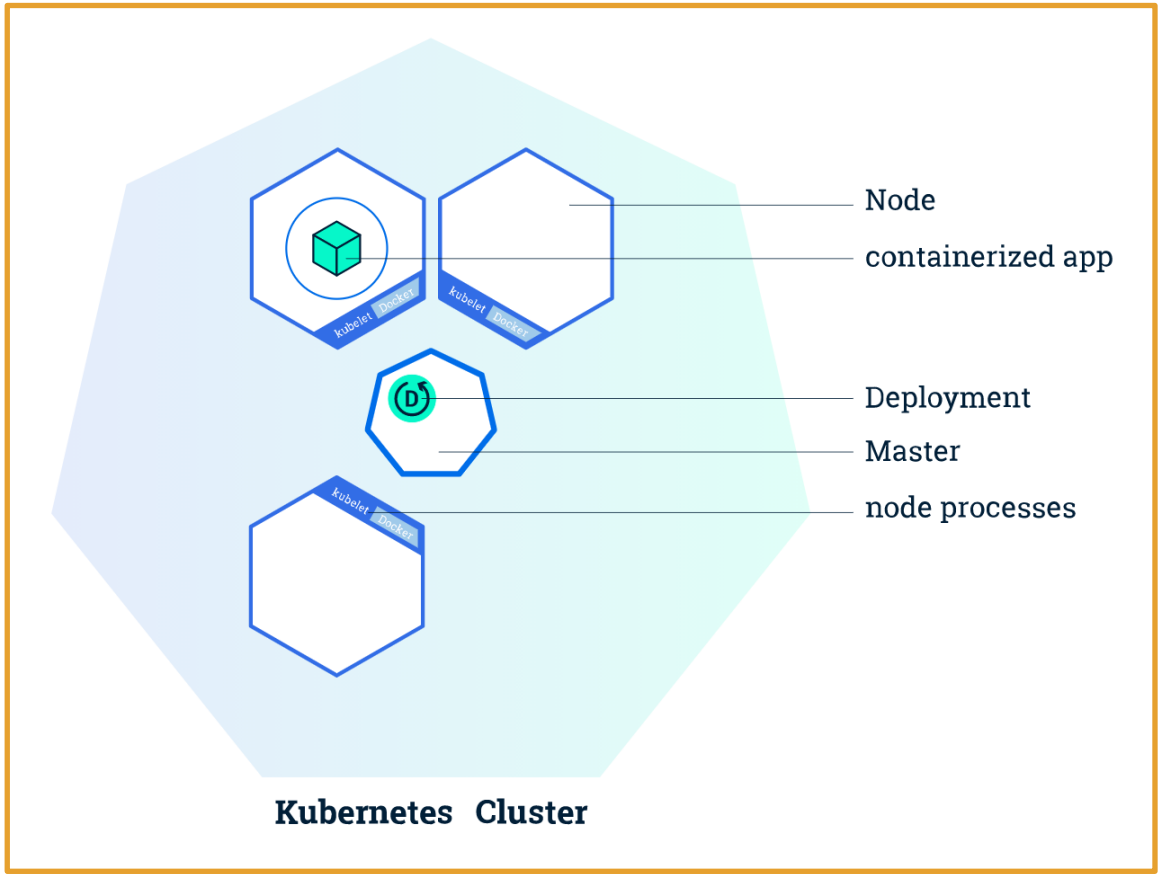
# Kubectl (say, “Cube CTL”)

https://kubernetes.io/docs/reference/kubectl/overview/

***Kubectl*** is the command line tool used to control Kubernetes clusters. ***Kubectl*** looks for a file named ***config*** in the $HOME/.kube directory. You can specify other kubeconfig files by setting the KUBECONFIG environment variable or by setting the --kubeconfig flag.

***Kubectl*** uses the ***Kubernetes API*** to interact with the cluster. The following syntax is used in command line to communicate through ***kubectl***:

- kubectl [command] [TYPE] [NAME] [flags]



# Kubectl (say, “Cube CTL”)

https://kubernetes.io/docs/tasks/kubectl/install/
https://kubernetes.io/docs/reference/kubectl/cheatsheet/

---

***Kubectl*** uses the ***Kubernetes API*** to interact with the cluster. The following syntax is used in command line to communicate through ***kubectl***:

•kubectl [command] [TYPE] [NAME] [flags]

| Command | Usage |
|---|---|
| [command] | Specifies the operation to perform on resources (create, get, describe, delete.) |
| [type] | Specifies the (case-insensitive) resource type. |
| [name] | Specifies the case-sensitive name of the resource. If the name is omitted, details for all resources are displayed. |
| [flags] | Specifies optional flags. |

# Deployment (1/2)

https://kubernetes.io/docs/tutorials/hello-minikube/

A *Deployment* is the recommended way to manage the creation and scaling of *Pods*.

The *Deployment Controller*:

- uses a *Deployment YAML* to change an unacceptable state to a desired state at a controlled rate.
- manages *ReplicaSets*,
- provides declarative updates to *Pods*,
- checks on *pod* health
- restarts terminated *pods*.

*Deployments* should be used instead of directly using *ReplicaSets* unless custom update orchestration is not required or updates themselves are not required.

```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
```

# Deployment Example (2/2)

https://kubernetes.io/docs/tutorials/hello-minikube/

```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
```

Create one container and name it nginx using the field, .spec.template.spec.containers[0].name

Deployment named nginx-deployment is created

The Deployment creates three replicated Pods

defines how the Deployment finds which Pods to manage

Pods run one container, nginx, on the nginx Docker Hub image, version 1.14.2

# ReplicaSet

https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/

The purpose of a ***ReplicaSet*** is to maintain a stable set of replica ***Pods*** running at a given time. It is often used to guarantee the availability of a specified number of identical ***Pods***.

A ***ReplicaSet*** will dynamically drive the ***cluster*** back to the predetermined desired state via creation of new ***Pods*** to keep an application running.

Use kubectl apply -f [URL] to apply a template.

***Pods*** can also be added without a template.

A ReplicaSet is defined with fields, including:

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  # modify replicas according to your case
  replicas: 3
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
```

replicas indicating how many Pods it should be maintaining

selector that specifies how to identify Pods it can acquire

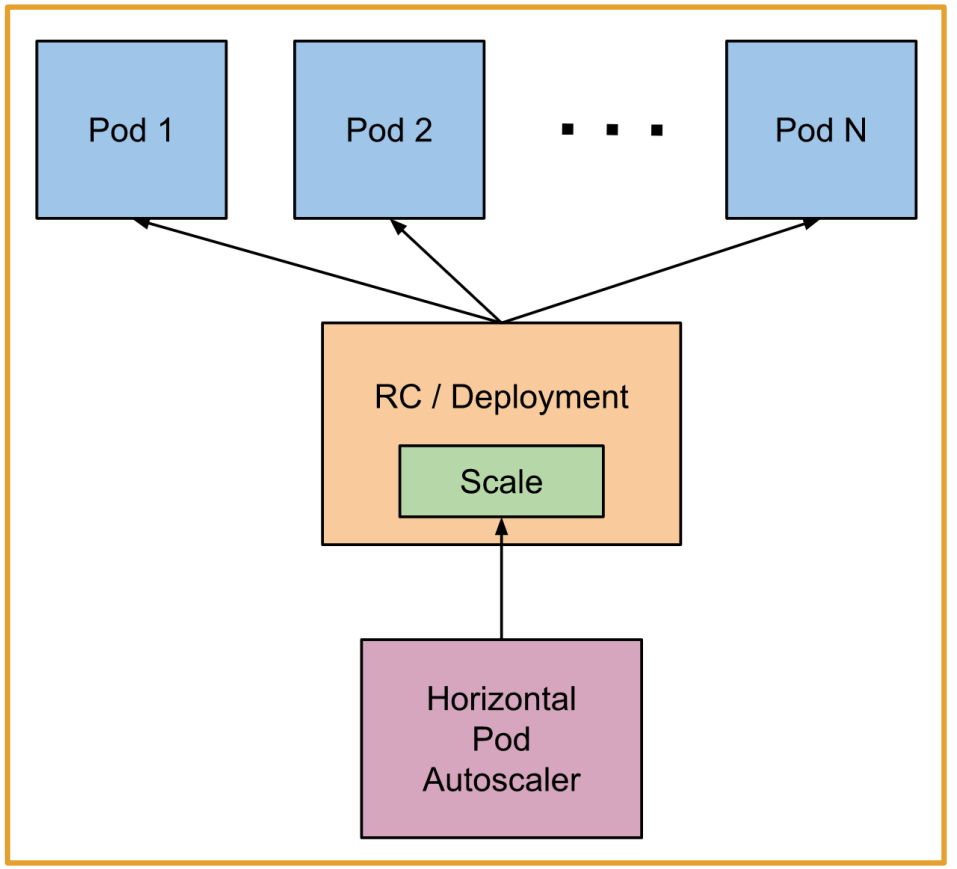
pod template specifying the data of new Pods it should create

# AutoScaling

https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

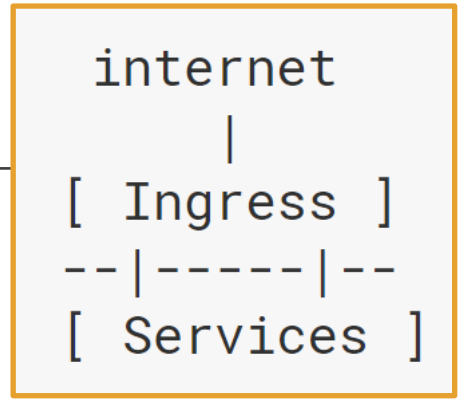
The ***Horizontal Pod Autoscaler*** is an API resource in the Kubernetes autoscaling API group which automatically scales the number of ***pods*** in a ***replication controller***, ***deployment***, ***replica set***, or ***stateful set*** based on CPU utilization.

The ***Horizontal Pod Autoscaler*** is implemented as a ***Kubernetes API*** resource and a ***controller*** in a control loop, with a period maintained by the controller manager. The ***Kubernetes API*** resource determines the behavior of the controller.



# Ingress (1/2)

https://kubernetes.io/docs/concepts/services-networking/ingress/




Linguistically, ***Ingress*** refers to the right to enter a property.

In Kubernetes, '***Ingress***' exposes HTTP routes from outside the ***cluster*** to services within the ***cluster***. It can give ***Services*** externally-reachable URLs, load balance traffic, terminate SSL/TLS, and offers name based virtual hosting.

Traffic routing is controlled by rules defined on the ***Ingress resource***. An ***Ingress Resource*** is (usually) a YAML file defining the rules for data accessing structures in a ***cluster***.

```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          serviceName: test
          servicePort: 80
```

# Ingress (2/2)

https://kubernetes.io/docs/concepts/services-networking/ingress/

As with all other Kubernetes resources, an ***Ingress*** needs ***apiVersion***, ***kind***, and ***metadata*** fields.

```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          serviceName: test
          servicePort: 80
```

The name of an Ingress object must be a valid DNS subdomain name

Ingress frequently uses annotations to configure some options depending on the Ingress controller

Each HTTP ***rule*** contains:

1. An optional ***host***. If no ***host*** is specified, the rule applies to all inbound HTTP traffic through the IP address specified.
2. A list of ***paths***.
3. A ***backend*** defines a ***serviceName*** and ***servicePort*** for each path. It is a combination of ***Service*** and ***port*** names.

The Ingress spec has all the information needed to configure a load balancer or proxy server.

# MiniKube

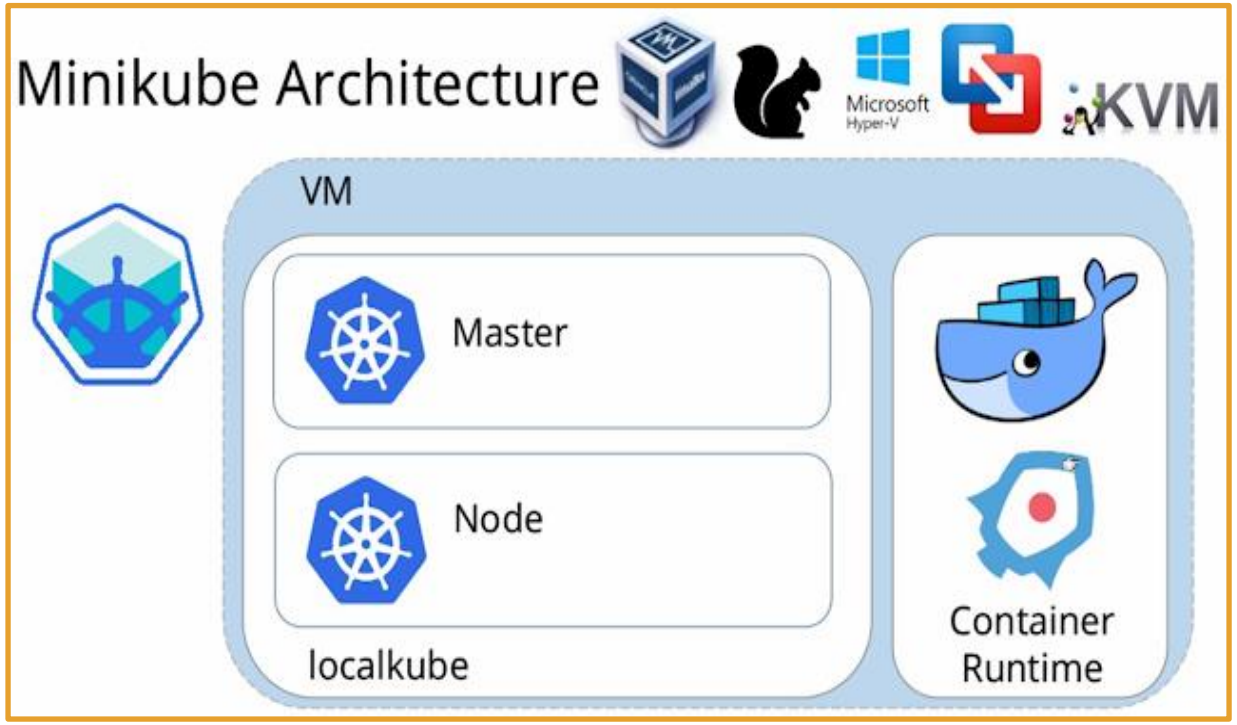
https://kubernetes.io/docs/setup/learning-environment/minikube/
https://kubernetes.io/docs/tasks/tools/install-minikube/
https://kubernetes.io/docs/tasks/tools/#minikube

---

## *Minikube:*

- is a tool that lets you run Kubernetes locally.
- runs a single-node ***Kubernetes cluster***
- runs on your personal computer (Windows, macOS, Linux PC).
- is great for trying out Kubernetes, or for daily development work.

The ***Minikube CLI*** provides basic bootstrapping operations for working with your cluster, including ***start***, ***stop***, ***status***, and ***delete***.



# Azure Kubernetes Service (AKS)

https://kubernetes.io/docs/setup/production-environment/turnkey/azure/#azure-kubernetes-service-aks
https://github.com/Azure/aks-engine/blob/master/docs/tutorials/README.md
https://docs.microsoft.com/en-us/azure/aks/intro-kubernetes

---

- The Azure Kubernetes Service (AKS) offers simple deployments for Kubernetes clusters.
- AKS makes it simple to deploy a managed Kubernetes cluster in Azure.
- AKS handles much of the complexity and operational overhead of managing Kubernetes.
- Azure handles critical tasks like health monitoring.
- The Kubernetes masters are managed by Azure. You only manage and maintain the agent nodes.
- AKS lets you integrate with Azure Active Directory and use Kubernetes role-based access controls.

# Assignment

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Create 7 quiz questions along with four plausible answers each to go along with the 6 chapters of the (above) tutorial and the lecture pdf. That means one question from each chapter and 1 from the pdf from each associate.

These questions will be included in a Google Forms quiz that you will take Monday morning.

This will serve as a primer for QC on Tuesday. Make sure to include a full range of questions from definitions of elements of Kubernetes structural questions to process to data flow to specific commands in the CTL, etc.

# Hello-Node Tutorial Step-By-Step(1/2)

https://kubernetes.io/docs/tutorials/hello-minikube/

---

- Create a Deployment that manages a Pod which will run a container based on the provided Docker Image with:
  - kubectl create deployment hello-node –image=k8s.gcr.io/echoserver:1.4
- See the deployment with:
  - kubectl get deployments.
- See the Pod with:
  - kubectl get pods.
- See cluster events with a:
  - kubectl get events.
- See the kubectl configuration with:
  - kubectl config view.
- Expose the ***Pod*** as a Kubernetes ***Service*** to make it visible from outside the ***Cluster*** with type=LoadBalancer as the expose keyword.
  - kubectl expose deployment hello-node --type=LoadBalancer --port=8080. (more on the next slide.)

# Hello-Node Tutorial Step-By-Step(2/2)

https://kubernetes.io/docs/tutorials/hello-minikube/

- View the service you just created with:
  - kubectl get services.
- External cloud providers get an external IP to access the service.
- Select + → Select Port to View on Host 1 → Enter the 5 digit port # after the:
- Take a look at available Add-Ons with:
  - minikube addons list.
- Enable the metrics-server add-on with:
  - minikube addons enable metrics-server.
- View the Pod with the service you just created with:
  - kubectl get pod, svc –n kube-system.

# Hello-Node Tutorial Step-By-Step(3/3)

https://kubernetes.io/docs/tutorials/hello-minikube/

- Disable the metrics-server with:
  - minikube addons disable metrics-server.
- Delete the service with:
  - kubectl delete service hello-node
- Delete the deployment with:
  - kubectl delete deployment hello-node
- (optional) stop Minikube with:
  - minikube stop
- (optional)Delete the Minikube VM with:
  - minikube delete

# More Tutorials

https://kubernetes.io/docs/tutorials/

---

- https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-prepare-app
- https://kubernetes.io/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/
- https://www.digitalocean.com/community/curriculums/kubernetes-for-full-stack-developers
- https://cloud.google.com/kubernetes-engine/kubernetes-comic/