* Course Overview
  + Kubernetes building blocks
  + Role of pods
  + Services
  + Secrets
* Overview
  + Kubernetes from a developer perspective
  + Creating pods
  + Creating deployments
  + Creating services
  + Understanding storage options
  + Creating ConfigMaps and Secrets
* Introduction
  + Can use Kubernetes in non production environment
* Kubernetes Overview
  + Open source system for automating deployment, scaling and management of containerized applications
  + How are you managing containers
    - Load balance to server with containers
  + Can use docker compose to manage containers
    - Not meant for production but can be used
  + Kubernetes is the conductor of a container orchestra
  + Key Kubernetes Features
    - Service discovery, load balancing
    - Storage orchestration/volumes
    - Automate rollouts/rollbacks
    - Self healing
    - Secret and configuration management
    - Horizontal scaling
    - More features
* The Big Picture
  + Container and cluster management
  + Kubernetes open source project
  + Supported by all major cloud platforms
  + Provides a “declarative” way to define cluster’s state
  + One or more master nodes
    - Manage worker nodes
  + Together they create a cluster
  + Master will start a pod on each node
  + Pod is a way to host a container
  + Pod is suit and container is a person in the suit
  + Deployment and replicaset to deploy pods
  + Need service for pods to talk to each other and the outside world
  + Node is like a VM, can run one or more pods
  + Store and controller manager
    - Store is database for master node to track things
    - Controller manager deals with request scheduling
  + kubectl command line tool
  + kubelet registers node with cluster and talks to manager
  + container runtime
  + Kube-Proxy: unique ip address for pods
* Benefits and Use Cases
  + benefits
    - accelerate developer onboarding
    - eliminate app conflicts
    - environment consistency
    - ships software faster
  + Key Kubernetes benefits
    - orchestrate containers
    - zero downtime deployments
    - self healing
    - scale containers
  + develop use cases
    - emulate production locally
    - move from docker compose to kubernetes
    - create an end to end testing environment
    - ensure application scales properly
    - ensure secrets/config are working properly
    - performance testing scenarios
    - workload scenarios(CI/CD and more)
    - learn how to leverage deployment options
    - help devops create resources and solve problems
* Running Kubernetes Locally
  + Minikube
  + docker desktop
  + kind: kubernetes in docker
  + kubeadam: full kubernetes, administrators
  + click on docker desktop icon
    - click on on settings
    - click on kubernetes
    - check enable kubernetes
    - click apply & restart
  + if it doesnt work, click restart docker
* Getting Started with kubecti
  + kubectl command line tool
  + type ‘kubectl version’ to check kubernetes version
  + type ‘kubectl cluster-info’ to get cluster information
  + ‘kubectl get all’ retrieve information about kubernets pods, services, etc
  + ‘kubectl run [container-name] --image=[image-name]’ simple way to create a deployment for a pod
  + ‘kubectl port-forward [pod] [ports]’ to forward a port to allow external access
    - normally a pod has a cluster ip, so only items in cluster can talk to pod
    - port-forward to expose it to external items
  + ‘kubectl expose ..’ expose a port for a deployment/prod
  + ‘kubectl create [resource]’ create a resource
  + ‘kubectl apply [resource’ create or modify a resource
  + ‘kubectl get pods’ to get all the pods
  + ‘kubectl get services’ to get all the services
* Web UI Dashboard
  + Optional
  + Visual dashboard to inspect nodes, etc
  + Steps
    - ‘kubectl apply -f [dashboard-yaml-url]’
    - ‘kubectl describe secret -n kube-sytem’
    - Locate the Kubernetes.io/service-account-token and copy it
    - kubectl proxy
    - visit the dashboard url and login using the token
  + type ‘kubectl apply -f [dashboard yaml link]’ to apply yml to set up for dashboard
  + then type ‘kubectl describe secret -n kube-system’
    - copy the first token
  + then run ‘kubectl proxy’ opens port to connect to dashboard
  + then open the link they specify under the proxy command on the documentation
    - <http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/>.
  + select token and paste in token you copied earlier
* Summary
  + Kubernetes provides container orchestration capabilities
  + Used for production, emulating production, testing and more
  + Several options to run Kubernetes locally
  + Interact with Kubernetes using kubectl
* Introduction
  + Pod core concepts
  + Create a pod
  + Kubectl and pods
  + Yaml fundamentals
  + Defining a pod with yaml
    - Declarative approach
  + Pod health
* Pod Core Concepts
  + Pod is the basic execution unit of a kubernetes application- the smallest and simplest unit in the Kubernetes object model that you create or deploy
  + Pods runs containers
    - Can have more than one container
  + Environment for containers
  + Organize application “parts” into pods(server, caching, APIs, database)
  + Pods have IP, memory, volumes, etc. shared across containers
  + Scale horizontally by adding pod replicas
  + Pods live and die but never come back to life
  + Master node schedules pod on a node
  + Pods can be scaled to have replicas
    - Kubernetes load balance between them
    - Kubernetes manages the health of the pods
  + Pods within a node have unique ip address, default is cluster ip
    - Containers within pods have their own unique ports
  + Pods containers share the same network namespace
  + Pods containers have the same loopback network interface
  + Containers processes need to bind to different ports within a pod
  + Ports can be reused by containers in separate pods
  + Pods do not span nodes
* Creating a Pod
  + There are several different ways to schedule a pod
    - kubectl run command
      * imperative way
    - kubectl create/apply command with a yaml file
      * declarative way
  + ‘kubectl run [podname] --image=nginx:alpine’ to run nginx:alpine container in a pod
  + ‘kubectl get pods’ list only pods
  + ‘kubectl get all’ list all resources
  + The ‘kubectl get’ command can be used to retrieve information about pods and many other Kubernetes objects
  + Pods and containers are only accessible within the Kubernetes cluster by default
    - Cluster ip address
  + One way to expose a container port externally: ‘kubectl port-forward’
  + ‘kubectl port-forward [name of pod] [host port/external port]:[container port/internal port]’
    - enables pod container to be called externally
  + ‘kubectl delete pod [name of pod]’ to delete pod
    - Pod will be delete but pod will be recreated
    - Kubernetes wants to maintain state
  + ‘kubectl delete deployment [name of deployment’ delete deployment that manages the pod
    - Pod will not be recreated
  + Running a pod will cause a deployment to be created
  + To delete a pod use kubectl delete pod or find the deployment and use kubectl delete deployment
* Kubectl and Pods
  + Deployment and replicas sets are responsible for actually deploying pods
  + A deployment is responsible for making sure that the current state is maintained
* YAML Fundamentals