* Course Overview
* Course Intro
  + Containers vs virtual machine
  + Docker
  + Kubernetes
  + Workloads
  + Orchestration
* Containers: Primer
* The Bad Old Days
  + Application for the most part on servers
  + One server per app
* Hello VMware
  + Can runs tons of apps on a single server
  + Not a perfect solution
* Vm Warts
  + Server will create a VM for each app it needs to run
  + Each one is a slice of the physical server hardware
  + Each VM will need its own dedicated operating system
  + Each OS will take a sizable portion of the resources to run
  + License cost, admin(patching, etc)
* Containers
  + Instead of multiple VM and multiple OS
  + Install one OS then on top of that create containers for each application
  + Each container is a slice of the OS
  + Inside the containers we run apps
  + Containers are smaller and more efficient than virtual machines
  + Virtual machine: software construct that acts as physical machine
    - OS steals resources
  + Containers are ideal for spinning things up and tearing things down on demand
* Container Demo
  + Docker can run on VMs
  + Docker does linux and windows
  + Docker on linux only runs linux apps
  + Docker on windows only runs window apps
  + May be possible to run your linux apps on docker on windows
  + Dock image: prepacked application or a vm template
    - Got everything wrapped up into a bundle that you need to run an application
  + Command line to fire up container from the image
    - docker container run -d - -name web -p 8080:8080 nigelpoulton/ctr-demo:1
    - docker container run -d is command fire up contain
    - - - name web sets the name of the contain to web
    - -p 8080:080 sets the network port to expose
    - nigelpulton/ctr-demo is the name of the image
  + Command line to stop container
    - docker stop [name of container]
    - ex) docker stop web
  + command line to start container again
    - docker container start [name of container]
    - ex) docker container start web
* Recap
  + Monolith/Legacy App has everything baked into single binary(program)
    - All functions in one program
    - Nightmare to update and fix
    - Have to update the entire codebase
  + Cloud native, microservice
    - Break out components and make each on its own app or service
    - The components still take to each other
  + Containers pave way for cloud service and microservice
  + Containers and virtualization usually are used together
* Docker
  + Docker inc, docker project
* Docker, Inc
  + Start as company(dotcloud) that provided developer platform on top of AWS
  + Had been using containers to build their platform on top of AWS
  + Had tech as internal tool to help spin up and manager container
  + Turn that tool to Docker
* Docker the Technology
  + Containers are like fast light weight VMs
  + Docker makes running apps inside of containers really easy
  + Community Edition(CE)
    - Open source
    - Free
    - Can contribute back
  + Enterprise Edition(EE)
    - Additional features
    - Official support
  + Containerizing Apps: apps running in containers
* Docker Demo
  + cloud native apps are built from lots of small parts
    - Web, Api, catalog, cart
  + in cloud microservice world
    - each service is coded separately
    - each one lives its own container
  + can update/fix service independent of the rest
  + take code -> build docker image -> push image to registry -> start a container from it
  + install docker
  + docker image build -t nigelpoultron/ctr-demo:2
    - build image from all files in current directory
    - with the name “nigelpultron/ctr-demo” with a tag of 2
  + docker takes source code and package it as a image
  + docker image push nigelpoulton/ctr-demo:2
    - pushes the image to docker hub
  + docker container run -d --name web -p 8000:808 nigelpoulton/ctr-demo:2
    - fires up the container from the image
    - give it a name
    - set the port where it will be available
    - from the image specified
* Docker Summary
  + Docker Swarm
  + Kubernetes
* Kubernetes
* Kubernetes: History
  + each google search runs in its own container
  + google made kubernetes, and made it open source
  + is backed by almost everyone
  + extensive features
  + kubernete(k8s)
* Kubernetes: The Short & Skinny
  + docker: provides mechanics for starting, stoping, and deleting individual containers
  + kubernetes: scheduling, scaling, healing, updating containers without downtime
    - the conductor in a musical
    - the musical are made up of containers
  + have a kubernetes node cluster to host our application
  + nodes have kubernetes software and container runtime
  + K8s control plane makes the decisions
  + decides which nodes to run stuff on
  + it always watching and if more is needed it will spin up more containers
  + if load decreases it will scale back down
  + docker does the low level of spinning up or down
  + but it only does it when kubernetes tells it too
  + thus kubernetes is managing a bunch of docker nodes
* Recap
* Preparing to Thrive in a Container World
* Individual Preparedness
  + Docker on desktop or docker on cloud
  + Play with Docker
    - <https://play-with-docker.com>
  + play with kubernetes
    - http://play-with-k8s.com
* Organization Preparedness
  + CI/CD workflows
  + Infrastructure services
* Recap
* Suitable Workloads
  + Amazing at Stateless
  + Can be used for Stateful
  + Stateful app/service
    - has to remember stuff(database)
    - data needs to persist
  + Stateless app/service
    - does not remember stuff
  + stateful restaurant remembers you(fancy restaurant)
  + stateless restaurant doesn’t remember you(mcdonalds)
  + docker, kubernetes are good at doing both
* Low-hanging fruit
  + cloud native & microservices
  + ability to adapt and change is vital to modern businesses
  + hypervisors revolutionized IT
  + Containers
* State & Legacy Apps
  + Stateful app is hard
  + Some legacy app can be migrated directly to containers
    - Temporary
* Recap
* Enterprise and Production Readiness
* Docker
  + Community Edition(CE)
  + Enterprise Edition(EE)
* Kubernetes
  + On Premises
  + Cloud
    - AWS EKS
    - Azure AKS
    - Google GKE
  + Alpha
    - Off by default
    - Early code
    - Uncertain future
    - Not for production
  + Beta
    - On by default
    - Becoming stable
    - Promising future
    - Some details may change
  + GA
    - Production ready
    - Solid future
    - Stable
* Ecosystem
* Summary
* A Word on Orchestration
  + Orchestration
    - Coach of football team tells them what to do
    - Coach orchestrates team to do things
* Orchestration
  + Apps comprised of
    - Multiple parts(services)
    - Multiple requirements
  + Game plan
    - Describes the app
  + Document Game Plan in version control system
  + Key to automation
    - Ordered startups
    - Intelligent scheduling
  + Give game plan to orchestrator(K8s)
  + Let orchestrator deploy and manage app
  + Orchestration
    - Define our app
    - How all parts interaction
    - Provision infrastructure
    - Deploy and manage the app
  + Main orchestrator is Kubernetes
* Summary
  + Each services run in its own containers
  + If services needs to be scaled up we just add more containers for it
  + Orchestrator to manage everything
    - Kubernetes
  + Application
    - Orchestrator, app manifest, services
  + Sports team
    - Coach, game plan, players
* What next