* Course Overview
  + Get up and running with docker
* Introduction
* Module Intro
  + How to install docker
  + Desktop installs
    - Windows, mac
  + Server install
    - Windows, linux
  + Cloud installs
    - Aws, azure
* Docker for Windows
  + A product
  + Get docker environment locally
  + windows 10 64 bit only
  + for test and dev work
  + not for production
  + docker engine will be running on linux inside of hyper v VM
  + can only run linux containers
  + native docker for windows 10 is coming
  + turn on Hyper-v/install Hyper-V
  + program and features -> turn windows features on or off -> check ‘Windows Hypervisor Platform’
  + go to docker website
  + download for windows
  + open command line when docker is installed and running
  + to check version type ‘docker version’
* Docker for Mac
  + linux VM
  + hyper kit
  + data kit
  + moby linux
  + need a moder mac(2010 an on)
  + Os X 10.10.3 or newer
  + go to docker website and install docker for mac
  + open terminal
  + type ‘docker info’ to get information about docker set up
  + type ‘docker images’ to get list of images
* Installing Docker on Windows Server 2016
  + native windows server container
  + open power shell
  + type ‘docker version’ to get the version
  + type ‘docker info’ to get information about currently running docker
  + download a base os image
    - type ‘Install-PackageProvider ContainerImage -Force’
  + find images
    - type ‘Find-ContainerImage’
  + grab image
    - type ‘Install-ContainerImage [name]’
  + to find list of download image
    - type ‘docker images’
  + unless you tell which image version or tag to go with, then the one tagged as latest is run
  + to tag image as latest
    - type ‘docker tag [image id] [image name]:latest
  + to run container
    - type ‘docker run -it [image name] cmd’
    - -it for attach powershell terminal
    - cmd to run cmd shell process inside of it
  + if we didnt tag it as latest, you will have to add the default tag
* Installing Docker on Linux
  + on Ubuntu
  + run command ‘wget -q0- <https://get.docker.com/> | sh’
    - grab install script and pipes it through shell
  + dont want to abuse root account to run docker
* Module Summary
* What is a Container
  + container split up OS resources
    - process ids, NET, File system
    - OS virtualization and assign one to each container
  + hypervisor virtualization, virtualizes physical server resources and builds virtual machines
* The ‘docker run’ Command
  + type ‘docker version’
    - to get version of client and server version
  + type ‘docker info’
    - shows info about docker host
  + type ‘docker run [image name]’
    - to spin up container
    - client talks to deamon
    - deamon finds image on machine or look in docker hub
    - use image as template to create container
  + type ‘docker ps’
    - show container currently running
  + type ‘docker ps -a’
    - shows containers that ran but is now excited
  + type ‘docker images’
    - see images
  + images
    - tag: can version it
    - id: unique hash
    - created
    - size
* Theory of Pulling and Running
  + docker host runs docker client and docker daemon
    - docker engine
  + installing docker gives the client and daemon on same host
  + docker run makes the appropriate api calls to the daemon
  + daemon implements the docker remote api(engine api)
  + docker run starts a new container with the image specified as a template
  + daemon checks local store to see if it already has a copy
    - if it doesn’t it will search for it on docker hub
  + docker hub: docker image registry
    - <https://hubs.docker.com>
    - place to store image to use for containers
    - default but can change registry
  + daemon then pulls image locally
  + daemon then create and spin of container based on configuration in image
* Working with Images
  + Images: stopped containers
  + Containers: running images
  + type ‘docker pull [image name]’ to pull images without starting container
    - looks for image in docker hub
  + type ‘docker pull [image name]:[version number]’ to pull a specific version of an image
  + type ‘docker images’ to show list of images in local image store
  + sign up for docker hub
  + type ‘docker rmi [image]:[tag]’ to delete image from local image store
* Container Lifecycle
  + can start, stopped, restrated, removed
  + ‘docker start [container]’
  + ‘docker stop [container]’
  + ‘docker rm [container]’
  + if you just stop and start containers, the data will persist
  + data will be removed if you remove the container
  + ‘docker run -d --name [name] -p [port:number] [image name]’
    - docker run: ask daemon to create new container
    - -d: start container in detached mode
      * don’t latch it on terminal
    - --name: unique name
    - -p: map network ports
      * ex) 80:8080
      * the second part after: is the port the web server listening to
      * the first part before: port of the docker host
      * map port 80 on docker host to port 8080 inside of the container
  + top level images are stored in the root of the hub
    - ex) nginx, busybox, ubuntu, redis, alpine
  + other images you will need to specify the namespace
    - ex) dockercloud/haproxy, puhsion/baseimage
  + type ‘docker ps’ to see containers
  + type ‘docker stop [container name]’ to stop the container
  + type ‘docker start [container name] to start container
  + type ‘docker run -it --name [name] [image name] /bin/bash
    - to interact with container using a terminal
    - will be inside container in terminal
  + containers are usually single process constructs
  + to get out of container without killing type ‘Ctrl P + Q’
  + typing exit will stop the container
  + type ‘docker stop $(docker ps -aq)’ to stop all containers
    - $(): gives output of inner command
    - docker ps -aq: list all containers
    - q tell it to just return container id’s
* Lesson Recap
  + docker run: starts a new container
  + docker pull: copies images to the docker host
  + docker images: lists images on the docker hub
  + docker rmi: removes images from the docker host
  + docker ps: list running containers
  + docker stop: stops running containers
  + docker start: starts stopped containers
  + docker rm: removes(deletes) stopped containers
* Module Intro
  + swarm mode and microservices
* Swarm Mode Theory
  + true native clustering
    - previously build a bunch of docker engine and grab swarm and layer it on top
  + A cluster = A swarm
    - a collection of docker engines joined into a cluster
  + Engines in a swarm run in a swarm mode
    - optional
  + swarm consists of manager nodes and worker nodes
  + Manager nodes: maintain the warm
    - look after state of cluster
    - dispatch task to work nodes
    - recommend to have odd numbers
    - only one is a leader
  + Raft Consensus Algorithm
    - ensure distributed consensus
  + Worker nodes: accepts tasks and executes them
  + Services: declarative way of running and scaling tasks
    - require swarm mode
    - expressing desired state and having docker keeping an eye on things
    - docker will make sure actual state is same as desired state
  + Tasks: atomic unit of work assigned to a worker node
* Configuring Swarm Mode
  + requires Docker 1.12 and higher
  + type ‘docker swarm init --advertise-addr [ip]:[port] --listen-addr [ip]:[port]’
    - --advertise-addr: this is address for swarm related stuff
    - --listen-addr: what nodes listens on for swarm manager traffic
    - always add these two lines when adding containers to the swarm
  + Engine port: 2375
  + Secure Engine port: 2376
  + Swarm port: 2377
  + copy given command to add worker to the swarm and manager to the swarm
  + type ‘docker swarm join-token manager’
    - to get command to add manager to swarm
  + type ‘docker swarm join-token worker’
    - to get command to add worker to swarm
  + type the given join command on another container to have it join the swarm
    - add --advertise-addr [container ip]:[port]
    - and add --listen-addr [container ip]:[port]
  + if you give the wrong address, things are going to be unpleasant
  + from a manager node type ‘docker node ls’ to see list of containers in swarm
  + from a manager node you can promote a worker node
  + type ‘docker node promote [id of container in swarm]’
  + each manager is also a worker
* Services
  + simplifying large scale production deployment
  + declarative
  + desired state, actual state
  + type ‘docker service create --name [name] [outside port]:[inside service port] --replicate [num] [image]
    - --replicate [num]: creates num tasks in service
  + type ‘docker service ls’
    - to see list of services
  + type ‘docker service ps [service name]’
    - see details about services
  + type ‘docker service inspect’
    - see configuration of services
  + overlay network is the future
  + can hit any node in the swarm and we will get to our service
    - even if that nodes isn’t running any tasks
  + routing mesh: native container-aware load balancer
    - traffic load balance
* Scaling Services
  + type ‘shutdown -h now’ to shut down a node
  + if one node running a task shuts down, another node will run in its place
  + type ‘docker service scale [service name]=[num]’
    - it will scale up or down to num of services
  + ‘docker service update --replicas [num] [service name]’
    - will achieve the same things as ‘docker service scale [service name]=[num]
  + adding new nodes, workload will not re-balance tasks with that new node added
    - might change in the future
  + type ‘docker node ps [name]’
    - will lists tasks the node is working
* Rolling Updates
  + type ‘docker service rm [service name]’
    - to remove service from node
  + type ‘docker network create -d overlay [network name]’
    - to create overlay network for service
  + then deploy the service
  + type ‘docker service create --name [name of service] --network [network name] -p [outsideport]:[inside port] --replicas [num of service] [name of image]:[version]
    - to create a service
  + type ‘docker service inspect --pretty [service name]’
    - see important information about the service
  + can set configuration while creating service, it will be the default
  + type ‘docker service create --update-parallelism [num] --update-delay [time][unit of time]’
  + ex) ‘docker service create --update-parallelism 2 --update-delay 10m’
    - roll tasks at a time
    - wait for a 10 min delay then roll another set
  + or you can specify it while updating
  + type ‘docker service update --image [image name]:[version] --update-parallelism [num] --update-delay[time][unit of time] [name of service]’
* Stack and DABS