* Docker
  + What is Docker
    - The number one containerization technology
    - Containerization is a process of building and managing virtual operating systems that generally have the code for some piece of software installed and configured
    - Is we have application that manages all containers- docker daemon connect host to container( any process that is supposed to be constantly running, generally low priority)
    - Docker daemon allows the containers to share some parts of their os ( kernal access to hardware) take turns using resources, but every single container will think that it is completely isolated
    - Because our container can share resources, each container is much more lightweight compared to traditional VMs
    - Easier to build than VMs
    - Easier to Share than VMs
    - Easier to Extend a previous Docker image
  + How does it compare to VMs
    - Virtual operating system
    - Some hypervisor ( like docker daemon ) virtualbox or vmware - connect host to virtual
    - Choose hardware to partition over to the VM, cpu cores, RAM, Disk space
    - Much less efficient use of resources compared to containerization
  + Why use it
    - Easier and better than VMs
    - Docker is by far the king when it comes to containerization
    - By building a virtual os, we can complete configure our application with everything it needs to be able to run ( this is potentially a lot of configuration we might have to do) this is going to be building a docker image
    - Once I have a docker image, any install of docker anywhere can run that image
    - Once we move to new machines, application can be live in seconds
    - Generally speed of service should be about the same, maybe a tiny bit slower
  + What is dockerd
    - The docker daemon, its the process that we interact with the most
    - Commands we write get sent here.
    - It manages all containers and there shared resources
    - It builds a virtual network for all the containers
    - It acts like a router between the host computer and the containers
  + What are docker commands
    - Docker -----management\_command some actual command
    - Docker image
      * List
      * Build
      * Pull
      * Push
      * Save
      * tag
    - Docker container
      * Run
      * Pause
      * Stop
      * Rm
      * Exec - allows us to execute a command in the container
  + What is a docker image
    - Is a file that docker can take and turn into a running container
  + What is a docker file
    - Dockerfile is a file we can make that programmatically builds a docker image from another docker image
    - Bunch of commands
      * 2 absolutely required
      * FROM - specify what image you are coming from ( base image is called scratch)
      * CMD/ENTRYPOINT - how the image should start
      * EXPOSE - make a port accessible to other containers
      * ARG - for passing in values on build
      * ENV - set an environment variable ( sometimes combined with ARG)
      * ADD - adding in packages ( probably don’t use)
      * COPY - moving files
  + How do I make an image
    - Docker image build PATH\_TO\_DOCKERFILE
    - -t to apply a tag
    - --build-arg for buildargs
    - Docker image tag dockerID newname:newtag
  + What is Dockerhub
    - Is a repository full of free to use Docker Images
    - This is where most of our images we pull FROM come from
    - We can push our own images to dockerhub
  + Other image commands
    - Docker pull - to pull an image from dockerhub
    - Docker push - push an image to dockerhub
    - Docker login - auth with dockerhub
  + What is a docker container
    - A currently managed image that can be run by the docker daemon
  + How do I get one running
    - Docker container run
    - -p EC2Port:ContainerPort port binding - bind a port on the host computer to a port on the container itself
    - -d detached, this makes the container run in the background
  + Other container commands
    - Docker container stop
    - Docker container start
    - Docker container rm - remove, you can only remove a stopped container
    - Docker container logs
  + Docker image best practices
    - <https://cloud.google.com/solutions/best-practices-for-building-containers>
  + Where can I learn more
    - <https://labs.play-with-docker.com/>
    - <https://training.play-with-docker.com/>
* Kuberenetes
  + What is orchestration
    - Container orchestration is the idea of having some greater management system that can deal with the creation and lifecycle of many different containers and can manage how they interact with each other
    - Docker Swarm (made by Docker) - 4 months ago, docker swarm is deprecated
    - Kubernetes - K8s - ( made by google )
  + What are some of the benefits it can give us
    - One, it can manage multi server deployments
    - Two, powerful controllers that allow us to set the desired state of the system - k8s will always move containers into that desired state
      * Self healing containers
    - Three, can handle service registry and gateway and load balancing
    - Four, very easy advanced deployment strategies
  + What are the pieces of the kubernetes engine
    - Kubernetes cluster
      * Some number of nodes, some of those nodes being workers and at least one node running the kubernetes control plane to act as the manager
    - Node
      * It represents an actual server that has hardware
      * A node generally has its own host OS
    - Kubernetes Control Plane
      * This is what makes the manager node work
      * Kube-api-server
        + This is the main process, it interacts with all other processes,
        + This is our entry point into the kubernetes cluster
      * Kube-controller manager
        + This is the piece in charge of all controller objects
        + A controller object is in charge of defining a piece of the desired state in the cluster
      * Kube-scheduler
        + Watches all objects and schedules those objects to nodes for them to execute
      * Etcd
        + This a little database, key value store, store all of the data for running the cluster
    - Node Components
      * Kubelet
        + Makes sure pods are running, keeps track of state, reports back to the control plane
      * Kube-proxy
        + Managing networking for the node and redirection of requests to appropriate pods
      * Container runtime
        + We need Docker, or containerd or some other runtime
    - Kube-dns - (service registry)
      * Generally on manager node, internal domain name service for our kubernetes objects
  + What are some objects we can make in kubernetes
    - Pod
      * Is some number 1-infinity of containers that are directly related to each other
      * Discrete piece of application
      * Basic pod - 1 container for some webservice
      * Advanced pod - 2 container 1 for webservice and 1 for a containerized DB for that webservice
      * Pods are ephemeral, they will be created and destroyed regularly
    - Service
      * A controller object
      * An object that wraps all copies of the same pod behind one ip address, generally with the purpose of making it accessible
      * Static ip address
      * Redirect traffic to an available pod
    - Volume
      * Permanent data storage that we can attach to a pod or a set of pods
    - Namespace
      * Like a db schema but for kubernetes
      * Different objects in different namespaces and we don’t interact between namespaces
      * To achieve multi-tenancy
    - Deployment
      * One of our most common objects
      * Controller object
      * To describe how to create a set of pods
      * Rules for replicas
      * Deployments have name
      * Rules for making each pod
      * It makes identical pods
    - ReplicaSet
      * Is a controller for managing the number of pods that needs to exist
      * When we make a deployment we also make a replica set
    - Job
      * Like a deployment
      * Run many pods and configure their running
      * Jobs are planned to terminate
      * Complicated multi pod one time process
    - Ingress
      * To be the entrypoint for all services in our cluster
      * I don’t need to expose any service to the outside internet, I only need to expose the ingress point
      * Nginx default web server, used very commonly for proxy server
  + How do I make a kubernetes object
    - One we need kubectl
    - Kubectl is the cli for interacting with a kubernetes cluster
    - Most of the time, we will use kubectl apply -f some\_config.yaml
    - General rules to yaml
      * Make apiversion
      * Kind
      * Metadata
      * Spec
  + Types of Services
    - ClusterIp
      * This will make a service for a set of pods and give them one static ip address, but that address will only be accessible in the cluster
    - NodePort
      * Will do the same as above, but it will redirect traffic from one port on the node itself to the clusterip
      * Accessible to world
      * Not recommended because it is hard to maintain and scale
    - LoadBalancer - general uses a cloud providers technology to build you a load balancer
  + Where can I learn more
    - <https://kubernetes.io/docs/home/>
  + Minikube
    - A tool for building local kubernetes clusters
    - It requires at least 2 cpu cores and 2 ish GB of RAM