# Container Orchestration

* Best way to scaling your app in this modern world.
* This will essentially automatically deploy more containers depending on the amount of workload
  + Think back about what a container is and what an image is
* The big idea:
  + We will create an image of our frontend and an image of our backend
  + We will have containers running both of those images
  + If workload for the frontend container is increasing, container orchestration will automatically create even more frontend containers using the same image and easily scale our application. (It can also do the reverse process and take down containers)
  + If workload for the backend container is increasing, then more backend containers will be deployed to compensate.

## Deployment

* Deploying the containers on a server.

## Scaling

* Easily create more containers using the same image.

## Network

* Built-in load balancer and service discoverability (was discussed last notes)

## Monitoring

* Service health checks
* Checking if the containers are still working
* Can also deploy a new container if one of them isn’t working to compensate

# Introduction to Kurbenetes

* It is a container orchestration tool
* It gives us the tools to essentially setup everything and manage containers
* We can scale those containers manually
* We can check their health
* We can setup the images to be run on those containers

## Kubernetes Architecture

## Containers

* Container is the running instance of an image.

## Pods

* They are the smallest unit in Kurbenetes
* May be made up of one or more containers
* The pod will keep on running and never stopping and if one pod somehow fails a new pod will come up to replace it
* They provide necessary environment variables needed to run the container
* Each pod will have a unique IP address, storage, config information for the containers

## Node

* Either a physical or virtual machine that contains necessary services to run multiple pods
* Managed by the control panel that allows container orchestration to take place and managing the different containers that is held by the pods

## Cluster

* A set of nodes that run containerized apps (a group of nodes).
  + In a cluster, there is a master node and one or more worker nodes
* So essentially your MSA app will run across a cluster

## Master Node

* Controls the state of the cluster
* Origin of all task assignments
* Coordinate processes such as:
  + Scheduling and scaling app
  + Maintains the cluster’s health/state
  + Implements updates

## Worker Nodes

* Contains the pods that runs your microservice
* Perform tasks assigned by the master node
* Composed of: kubelet, container runtime, kube-proxy

### Kubelet

* Primary node agent
* Registers the node with the master node
* Used by the master node to monitor the node
  + Ensures that the node is working properly

### Container runtime

* Software responsible for running containers
* Ex: Docker or Azure container registries

### Kube-proxy

* Network proxy
* Maintains network rules on nodes, these network rules allow network communication to your pods from network sessions inside or outside of your cluster