# Docker Introduction

* It provides the ability to run one or more application in an isolated environment called a container
* It is a containerization ecosystem

## What is Containerization?

* Involved bundling an application together with all the configuration files, libraries, and dependencies required
* We will be using **Docker** and **Kubernetes**
* You are hosting a virtual OS
* When you create a container, the allocation of resources is dynamic
  + It will only use as many resources as possible the container needs to run

## What is Virtualization?

* It is a creation of a virtual machine that stimulates a real computer with an operating system
* When you create virtual machines, the allocation of resources is static
  + Meaning once you start a virtual machine, you cannot change the resources it currently has
* It is heavy to run

# What is the purpose of Docker?

* It allows developers to work in standardize environments using containers
  + Meaning they can work on any computer with any OS they are comfortable with and still be able to share their application to everyone
* Makes it perfect for CI/CD workflow
* It makes scaling and deployment way easier
  + Docker containers can run on most things (cloud providers, data center, virtual/physical machine)
  + You can scale up or tear down application as business dictates

# Docker artifacts/terminology

## Docker Images

* They are standalone package that includes everything we need to run an application such as code itself, runtime environment, libraries, etc.
* They are immutable file and represents an application and its virtual environment at a specific point in time
  + Great for consistency when sharing your application everyone
  + Immutable meaning once you make it you can’t change it, you’ll have to create a new version of your image if you want to update it
* Essentially, they are a virtual file storage that has a copy of your files and other development platform it needs to run those files

## Docker Container

* **An image cannot run on its own,** it needs a docker container to run the image
* Docker container is the runnable instance of a docker image

## Docker ignore

* It will ignore certain files and not copy them to the image
* Like gitignore

## Docker Registry

* It is a server-side or cloud application where you can store your images and make it easy to distribute to everyone else
* Think of github but just for docker images
* **Docker hub**

## Docker Configuration

* Contains all the information to tell the docker container how to run the image
* We used the docker run command to already configure this (mostly just changing what port the container should run on)

# Docker instructions

## From

* Initialize our build stage and sets the base image
* Essentially this is where we indicate what we need to be able create/run our application

## Workdir

* Sets the working directory
* Just creates a folder and this is where we will copy and paste our files into that folder

## Copy

* Copy and pastes files into the image’s virtual file system

## Run

* It will run what you put in the terminal

# Docker CLI

* This will build the docker image
  + Docker build -t [username]/[imageName]:[imageVersion] .
* This will run the docker image in a container
  + Docker run -d -p 5001:80 -t [username]/[imageName]:[imageVersion]

# DevOps

* It is a culture that you follow in which you continuously develop your application and continuously deploy it.
* There are three main teams that are part of DevOps
  + Developer team that are responsible for planning and building the application (by adding more features)
  + Operation team that are responsible for deploying and maintain the application (check that the new feature added and can handle real world environment)
    - The application must deal with being used by thousands if not millions of people
  + Business team verifies that the correct product is created and delivered
    - They check if the people even want that new feature

## Cycle time

* You observe what the market needs
* You then decide what options you should pursue to satisfy what the market needs
* Finally, you give the working software that should satisfy what the market needs

## Feedback loop

* Once released, you get feedback if it did satisfy what the market needs

## Validation

* Using the feedback, learn what needs to change and adapt your software to satisfy what the market needs

# Continuous Integration

* The process of automating the building and testing of your code every time someone tries to merge with the main branch of your repository.
* Some automated testing process will occur and if this fails, the system will prevent your code from merging with the main branch
* So, in this way, there is a lot less bugs that can roll out to the real application that is being used by everyone
* Essentially, we take out the human component of someone looking at the merge request and checking everything looks good.

# Continuous Delivery

* The extension of CI since all CI does is checks if you can merge with the main branch and now, we must deal with taking that main branch and releasing to everyone else
  + Anything from CI has been tested correctly
* Essentially, it is the automated process of delivering the new changes in the main branch to the people
* However, there is a release manager that will check on everything and make sure it looks good before allowing the entire process to continue
  + They might test it in a more real environment than the testing environment
    - Things like stress testing the application by emulating thousands or millions of people using it
  + They might release the application to a smaller set of people and see how it goes from there (beta testing for games)
  + They essentially have the ultimate power to push your code to production (real world)

# Continuous Deployment

* Everything is automated from building and testing your code (what CI does) to deploying your new version to the people (what CD does).
* Doesn’t have any human intervention so as soon as your unit testing passes, it will be deployed to the people
* Not really common in the industry since simple changes can cost millions of monies.
  + [Knightmare: A DevOps Cautionary Tale – Doug Seven](https://dougseven.com/2014/04/17/knightmare-a-devops-cautionary-tale/)