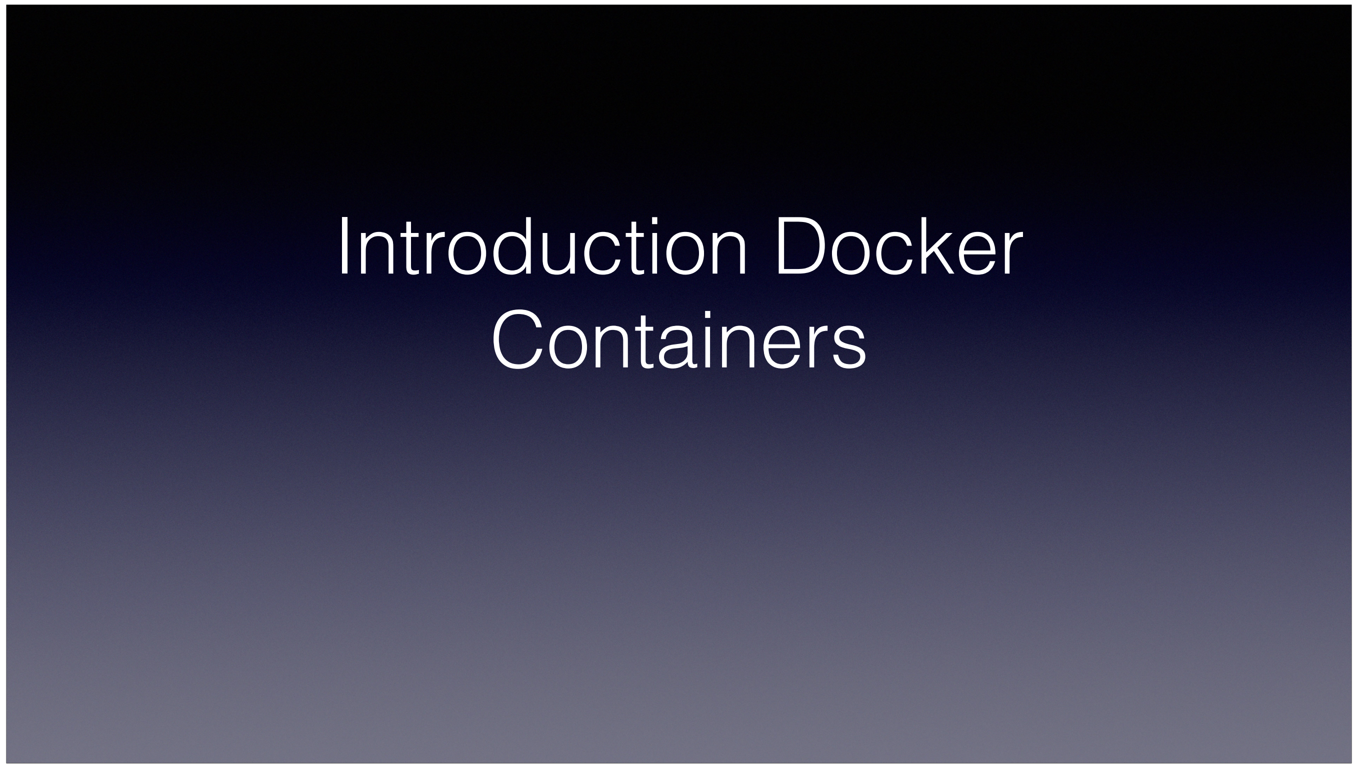

# Introduction Docker Containers

# Containers

- Virtualize an operating system
- Lighter weight compared to VM
- Uses host OS to processes and threads via kernel facilities like Linux cgroups.
- Host OS filesystems mapped into the the container
- System and OS libraries provided by container's view of filesystem

## Docker

- Docker is very popular container system for Linux OS
- Versions for Windows and MacOS
- Installs as a daemon process and is run as root
- Daemon will start a subprocess for each container

## Dockerfile

- Dockerfile's are a set of instructions for building a container
- Usually start with a base image
- Adds layers to the container file system image
- Installs or copies files into locations
- Designed to run one process

## Container Context

- Dockerfiles configure the environment within the container
- Containers usually start with a bare-bones base image: just the barest system libraries and tools any app may need
- Directives will then install additional libraries and tools onto the container base image
- Environment variables, user settings, and port mappings are provided
- When running a container you can mount-bind local directories into the container
- When running you will map ports from your host to the container ports

## Attaching to a Container

- Containers can attach other processes
- Attaching gives another process access to the same file system but not access to the processes running in the container
- This is not an OS.

## Containers working together

- Containers can communicate with each other via network protocols
- Often applications are designed as a set of relatively independent components, each running in its own container. (Micro-services)
- Containers can be orchestrated together via Kubernetes or similar tools.
- Docker supports Docker Compose files for built in orchestration