

# Docker and Kubernetes

...

Hands-on Workshop

# Authors



Rutuja Joshi  
Senior Principal Software Engineer  
Oracle Analytics Cloud  
rutujaj@gmail.com



Anita Carey  
Staff Engineer, Site Reliability  
LinkedIn  
anitacarey1101@gmail.com

# Agenda

Download Code and Instructions from: [git@github.com:coder-lgtm/docker-k8s.git](https://github.com/coder-lgtm/docker-k8s.git)

- Introduction to Docker
- Virtualization technology concepts
- Launching a web application locally using Docker
- Introduction to Kubernetes
- Re-launching the same application in the Public Cloud using Kubernetes
- Scaling up / scaling down the website
- Q & A

# What is Docker

- PaaS: Platform-as-a-Service offering - provides a platform to package software and dependencies together.
- Application containerization - is an OS-level virtualization method used to deploy and run distributed applications without launching an entire virtual machine (VM) for each app. Multiple isolated applications or services run on a single host and access the same OS kernel.

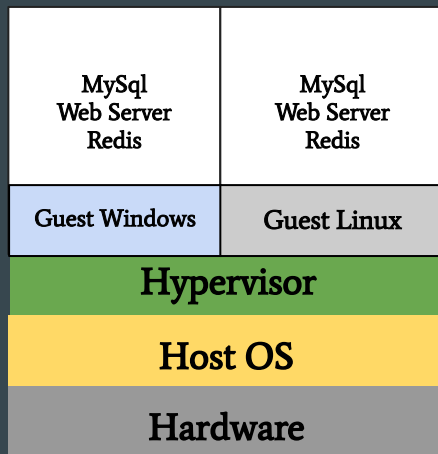
# Why Docker

- **For Development:** Developers can collaborate with each other over shared software code by eliminating “runs on my machine but not yours” problem
- **For Deployments:** Helps save computing power by running multiple containers (and hence multiple software such as a web server and database) side by side on one host machine
- **For Delivery:** Enterprises use docker to ship software in agile fashion with confidence that it will work on various operating systems without issues
- **How do we share Docker Images?** One example: <https://hub.docker.com/>  
Like Github, a public container repository provided by Docker

# Docker Terminology

- **Docker Image:** Lightweight and standalone executable that forms the package, which can include code, dependencies, settings, and system tools.
- **Docker Machine:** A tool for provisioning and managing your Dockerized hosts.
- **Docker Container:** Runtime environment for the Docker Image
- **Docker Engine:** Runs on Docker hosts (e.g., Virtualbox); this is a tool provides the Docker command line interface (CLI) and Docker server-side APIs to manage the lifecycle of Docker containers

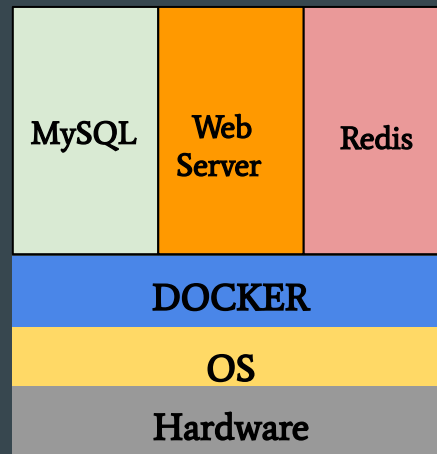
# Docker Vs Virtual Machines



Guest OS sits on Hypervisor, does not share resources

Mins to launch each Guest OS, size is in GBs

**Virtual Machine**



Docker containers. Share the same OS Layer/ Kernel

Milliseconds to launch, size is in MBs

**Docker**

# How to work with Docker? (One of the many ways...)

- Docker works on top of Linux Kernel
- How do you make it work on MacOS? Install Docker Toolbox
- The Toolbox installs and configures: VirtualBox and Docker Machine
- The Docker Machine provisions Docker hosts with the Docker Engine
- The Docker Engine offers command line interface that allows us to build and run Docker containers on top of VirtualBox

```
> docker-machine env
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://192.168.99.101:2376"
export DOCKER_CERT_PATH="/Users/rj/.docker/machine/machines/default"
export DOCKER_MACHINE_NAME="default"
# Run this command to configure your shell:
# eval $(docker-machine env)
```



# Demo Time!

- Prerequisites: follow the README.md in the [git@github.com:coder-lgtm/docker-k8s.git](https://github.com/coder-lgtm/docker-k8s.git)
  - Create a small webapp
  - Install Docker and Virtualbox
  - Validate installation
- Demo:



- Python Flask App
- Dockerize it
- Run it
- Learn about various useful Docker commands

# Kubernetes

- Kubernetes provides Container Orchestration
- Why Container Orchestration?
  - If you have only one instance of your app, if it goes down, you have an outage
  - What if some of the running containers crash?
  - How do you make your application robust?
    - Auto scaling?
    - Load balancing?
    - Self-healing?
- Kubernetes is one orchestration mechanism - other options include Docker Swarm and Mesos
  - Kubernetes is the most popular option and available on all public cloud platforms.

# Kubernetes Terminology

- Cluster: A collection of nodes that Kubernetes uses to run instances of your application
- Node: A physical machine / host in a cluster
- Pod: A unit of execution on a node. Containers run in a pod on a node. One node can have multiple pods running on it
- kubectl: The command-line tool to run Kubernetes
- Rolling Update: Incrementally updating pod instances with code updates
- Auto Scaling: Kubernetes will automatically scale up your cluster by adding instances when the load increases and scale it down when load decreases (to save you money)

# Demo Time!

- Prerequisites: follow the README.md in the [git@github.com:coder-lgtm/docker-k8s.git](https://github.com/coder-lgtm/docker-k8s)
  - Set up a GCP project
- Demo:
  - How does kubectl explore nodes / pods?
  - What processes run on the pods?
  - How do you access containers running on the pod?
  - How to scale up / scale down?
  - How to set up auto scaling?

# Summary

Docker and Kubernetes offer a fundamental shift to software development with a new level of portability and velocity.

Docker is a PaaS offering that lets you build and manage the lifecycle of a “container” to simplify and accelerate software development.

Kubernetes is an orchestration tool for Docker. Production environment workloads require effective management of the containers and Kubernetes is a brilliant tool for that - it offers load management, auto-scaling, self-healing, creation, deletion, and rolling updates (new code release, geographic migration, etc.).

Thank you!

Q & A