**Learning**

**Docker and Kubernetes**

**Through the Lens**

**of**

**Python Development**

Table of Contents

[1 General Introduction 3](#_Toc29066284)

[2 Glossary 3](#_Toc29066285)

[3 Docker 4](#_Toc29066286)

[3.1 Introduction 4](#_Toc29066287)

[3.2 Use Cases that illustrate the importance of using Docker 4](#_Toc29066288)

[3.3 Docker versus Virtual Machines (VMs) 5](#_Toc29066289)

[3.4 Does Docker help us dispense with VMs ? 5](#_Toc29066290)

[3.5 Alternatives to Docker 5](#_Toc29066291)

[4 Kubernetes 5](#_Toc29066292)

[4.1 Introduction 5](#_Toc29066293)

[4.2 Use Cases that illustrate the importance of using Kubernetes 5](#_Toc29066294)

[5 Docker and Kubernetes together 5](#_Toc29066295)

[6 Why Docker and Kubernetes with Python development 5](#_Toc29066296)

[7 A complete CI/CD example using Docker, Kubernetes and Python 5](#_Toc29066297)

[8 Using Docker to learn stuff on Windows 5](#_Toc29066298)

[8.1 Ubuntu 5](#_Toc29066299)

[9 Using Docker to run stuff on Windows 6](#_Toc29066300)

[9.1 Postgresql 6](#_Toc29066301)

[9.2 Redis 6](#_Toc29066302)

[9.3 NGINX 6](#_Toc29066303)

[9.4 Flask apps on Gunicorn 6](#_Toc29066304)

[10 Ubuntu 9](#_Toc29066305)

# General Introduction

# Glossary

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Term** | **Description** | **Additional Reading** |
|  | Docker image | * A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings. | <https://www.docker.com/resources/what-container> |
|  | Docker container | * A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. * A container is always derived from an image | <https://www.docker.com/resources/what-container>  (A container, in a way, is like a SandBox) |
|  | Virtualization |  |  |
|  | Hyper-V |  |  |
|  | Layer |  |  |
|  | Linux Containers |  |  |
|  | Container Orchestration |  |  |
|  | Virtual Machines |  |  |
|  | APT | Advanced Packaging Tool |  |
|  | Multi container Applications |  |  |
|  | Volumes | Volumes are directories (or files) that are outside of the default Union File System *[Docker’s own internal filesystem ]* and exist as normal directories and files on the host filesystem.  Another description –  A Volume is a a file system that lives on a host machine outside of any container.  Volumes are created and managed by Docker.  Volumes are:   * persistent * free-floating filesystems, separate from any one container * sharable with other containers * efficient for input and output * able to be hosted on remote cloud providers * encryptable * nameable * able to have their content pre-populated by a container * handy for testing | <https://docs.docker.com/storage/volumes/> |
|  | Docker Compose |  |  |
|  | Running Container |  |  |
|  | Image Developer |  |  |
|  | Dockerfile |  |  |
|  | Union File System |  |  |
|  | Mountpoint |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Docker

## Introduction

This document aims to introduce the reader to the use of Docker and Kubernetes (“DANK”) in modern software development. Generally speaking, people discuss DANK in the context of CI/CD, but the use is not just restricted to that, project teams can leverage the power of DANK for a variety of purposes.

Docker is built to deploy applications, not machines

## Use Cases that illustrate the importance of using Docker

|  |  |  |
| --- | --- | --- |
| **#** | **Use Case** |  |
| 1 | A set of development machines need to access Postgresql database, but can’t install it |  |
| 2 | Windows environment – Need to use a software that runs only on LINUX |  |
| 3 | Need to provide a testing environment to multiple testers |  |
| 4 | Your application runs on Python 3.7 but the production environment uses Python 2 and this can’t be changed |  |
|  |  |  |
|  |  |  |

## Docker versus Virtual Machines (VMs)

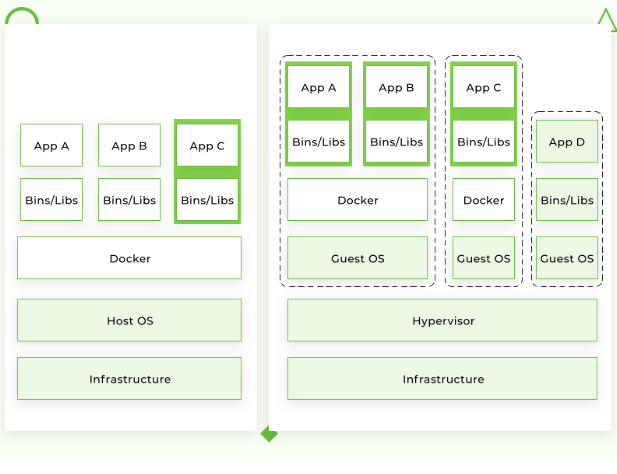


Figure – Source - <https://djangostars.com/blog/what-is-docker-and-how-to-use-it-with-python/>

## Does Docker help us dispense with VMs ?

A more appropriate response is “Can Docker work with VMs ?”.

An interesting article is <https://www.docker.com/blog/containers-and-vms-together/>

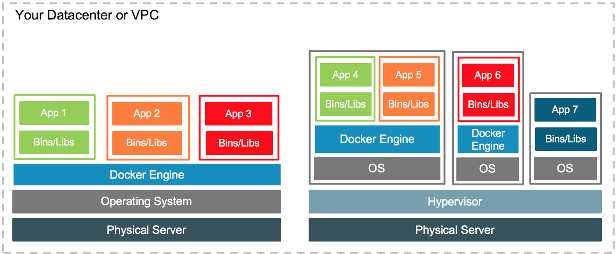


Figure – Source - <https://www.docker.com/blog/containers-and-vms-together/>

(This is the same as the above diagram !)

## Alternatives to Docker

# Kubernetes

## Introduction

## Use Cases that illustrate the importance of using Kubernetes

# Docker and Kubernetes together

# Why Docker and Kubernetes with Python development

# A complete CI/CD example using Docker, Kubernetes and Python

# Using Docker to learn stuff on Windows

## Ubuntu

# Using Docker to run stuff on Windows

## Postgresql

## Redis

## NGINX

## Flask apps on Gunicorn

| **MAIN TOPIC** | **SUB TOPIC** | **DETAILS**  **/**  **(LINKS FOR FURTHER STUDY)**  **/**  **(FEEDBACK)**  **/**  **(SAMPLE PROGRAMS)** | **(CLASSROOM EXERCISES)**  **/**  **(ASSIGNMENTS)** | **TRACKING DATA** |
| --- | --- | --- | --- | --- |
| **OVERALL CONTEXT** | WHAT ARE YOU EXPECTING ? | <Update after feedback from the students> | **N/A** | DAY 1  (<=15 mins) |
|  | MY EXPECTATIONS FROM THE STUDENTS/YOU | * Be aware of the course content (*Have all of you gone through the course details [separate doc] ?)* * Do the class room exercises * Complete your assignments * Make notes *(I do it and it helps me)* * Don’t just nod your head to what I say. Digest it slowly. Stop me if I am going too fast | **N/A** | DAY 1  (<= 15 mins) |
|  | An overview of commonly used Docker commands | *#*  *# Docker Images, Containers, docker files,* layers , docker build  #  *# Use this command to search the repository*  docker search <>  *##*  *## Launch a container*  docker **run** -t -i ubuntu /bin/bash  *## get the container IDs*  docker **ps -all**  *##*  docker ps  *##*  1. docker ps -all (use this to get the container id)  2. docker **start** <container ID>  3. docker **attach** <container id>  *## Listing all containers running or otherwise*  docker container ls -a  *## Removing a container*  docker container rm  ## Run a command in a running container  docker exec  *## Attach local standard input, output, and error streams to a running container*  docker attach |  |  |
|  | Going deeper into Docker container commands | <https://docs.docker.com/engine/reference/commandline/container/>  The main ones :-  *# start an existing docker container*  docker start [container id]  *# stop an existing docker container*  docker stop [container id]  *# Use docker attach to attach your terminal’s*  *# standard input, output, and error (or any*  *# combination of the three) to a running*  *# container using the container’s ID or name.*  *# This allows you to view its ongoing output or*  *# to control it interactively, as though the*  *# commands were running directly in your*  *# terminal*  docker **attach** <container id>  *# Kill an existing container*  docker kill [container id]  *# pause an existing docker container*  docker pause [container id]  *# Run a shell command inside a container*  docker exec {-ti} [container id] {command}  *# restart an existing docker container*  docker restart [container id]  *# Inspect an existing docker container*  docker container inspect [container id]  Note – There are many more commands. Please refer to the docker documentation |  |  |
|  |  |  |  |  |
|  | Persisting data – setting the stage for Volumes | There are two ways to persist data beyond the life of the container. One way is to *bind mount* a file system to the container.  With a bind mount, processes outside Docker also can modify the data.  <https://docs.docker.com/storage/bind-mounts/>  <https://docs.docker.com/storage/volumes/>  Volumes rather than bind mounts should be used for persisting data |  |  |
|  |  |  |  |  |
|  | Understanding Docker Volumes | See above |  |  |
|  |  |  |  |  |
|  | Volume related commands | Key commands   * docker volume create * docker volume ls * docker volume inspect * docker volume rm * docker volume prune |  |  |
|  |  |  |  |  |
|  | Run multiple containers of the same image | This is an exercise that you can try out. The Hello World image can be used | **Why would you launch multiple containers of the same image ?**  **Create a volume that is used by the different containers of the same image** |  |
|  |  |  |  |  |
|  | **Docker-compose**  **(Critical feature)** | <https://docs.docker.com/compose/>  <https://www.baeldung.com/docker-compose>  <https://www.youtube.com/watch?v=Qw9zlE3t8Ko>  Using Compose is basically a three-step process:   * Define your app’s environment with a Dockerfile so it can be reproduced anywhere. * Define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment. * Run docker-compose up and Compose starts and runs your entire app. |  |  |
|  |  |  |  |  |
|  | Run multiple containers (from different images) / Multi-container docker applications | This is achieved using Docker compose.  A thorough understanding of Docker compose is required to operate a multi container docker environment |  |  |
|  |  |  |  |  |
|  | For a single container, persisting data | This is achieved through the use of Volumes |  |  |
|  |  |  |  |  |
|  | Sharing data across multiple containers | This is achieved through the use of Volumes |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | Docker Application Development | * <https://docs.docker.com/engine/reference/commandline/image_build/> * <https://www.scalyr.com/blog/create-docker-image/> * <https://www.youtube.com/watch?v=YFl2mCHdv24> |  |  |
|  |  |  |  |  |
|  | **Docker build**  **(Critical command)** | * A **Dockerfile** is a script that contains collections of commands and instructions that will be automatically executed in sequence in the docker environment for building a new docker image |  |  |
|  |  |  |  |  |
|  | Key DockerFile commands | **FROM**  **MAINTAINER**  **RUN**  **ADD** |  |  |
|  |  |  |  |  |
|  | Networking concepts in Docker |  |  |  |
|  |  |  |  |  |
|  | Managing Application Data |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Ubuntu

|  |
| --- |
| docker container run --interactive --tty --rm ubuntu bash |
|  |
| (sudo) apt-get update |
| (sudo) apt-get install apt-utils |
| (sudo) apt-get install python3.7 |
| apt install python3-pip |
| # difference between apt-get and apt  # apt list –upgradable  #  # dpkg -l | grep systemd  #  # |
| python3/3.7 -m pip install --upgrade pip |
| # for virtual environments  apt-get install python3.7-venv |
|  |
| python3.7 -m venv /home/workspaces/my-python-sandpit/muttli  / |
| Source activate  Deactivate  From the bin folder that resides within your virtual environment |
|  |
| Apt install nginx |
|  |
| apt install docker.io  systemctl start docker  systemctl enable docker |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

## 