

# Learning Session Series

**Topic: MLOps for AI Engineer and Data Scientist**

**Sub-topic: Containers for ML Deployment**

**Speaker: Joseph Itopa Abubakar**

December 08<sup>th</sup>, 2021





# Learning section objective:

- ❖ Gain an introductory knowledge about Web APIs.
- ❖ Understand Basic Concept of Docker and how to containerize a simple Flask/Fast Api.
- ❖ To have fundamental knowledge of kubernertes
- ❖ To learn the concept of continuous delivery in MLOps.



# Agenda



- Introduction to WebApi
- Introduction to Docker
- Introduction to Kubernetes
- Containerize web apps using docker
- An introduction to automating ML deployment workflow

# Introduction to Web APIs



## ✓ **What is Web Api?**

Web Api is a type of Apis that was derived from the concept of web development, to aid functionality for http clients or web browsers.

## ✓ **Features of Web Api**

- Web Apis are client-side oriented.
- Most times access is via URI using http requests.
- Web Api uses REST for it interactions.

# Introduction to Web APIs

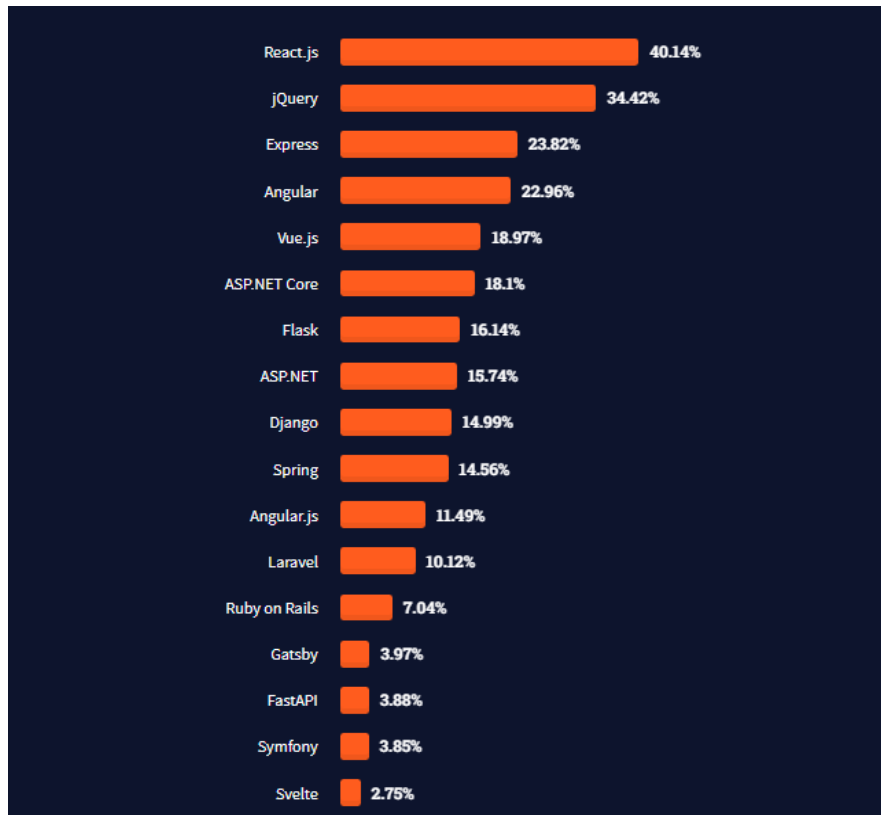


Figure 1: 2021 Survey on web frameworks popularity [StackOverflow]

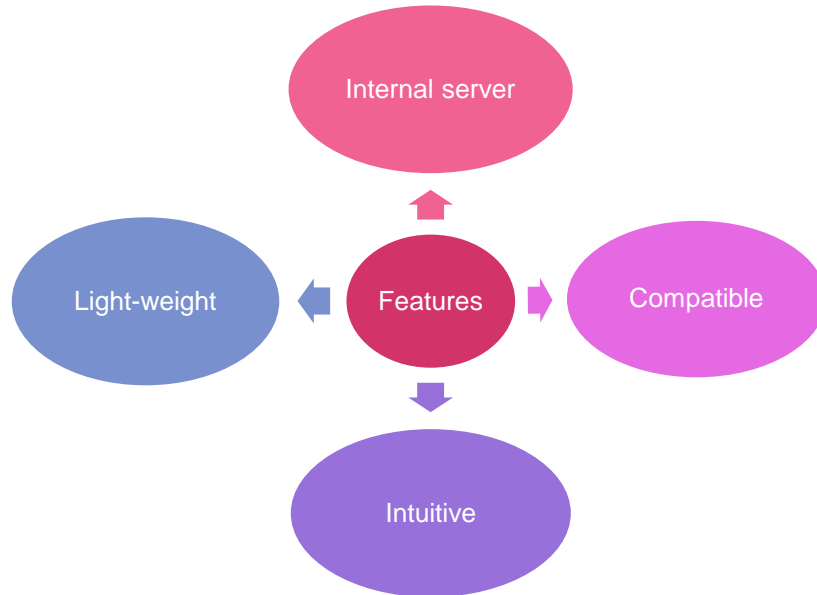
Reference: <https://insights.stackoverflow.com/survey/2021/>

# Introduction to Web APIs



## What is FlaskApi?

- According to wikipedia, Flask is a “*micro web framework written in python for developing APIs*”.

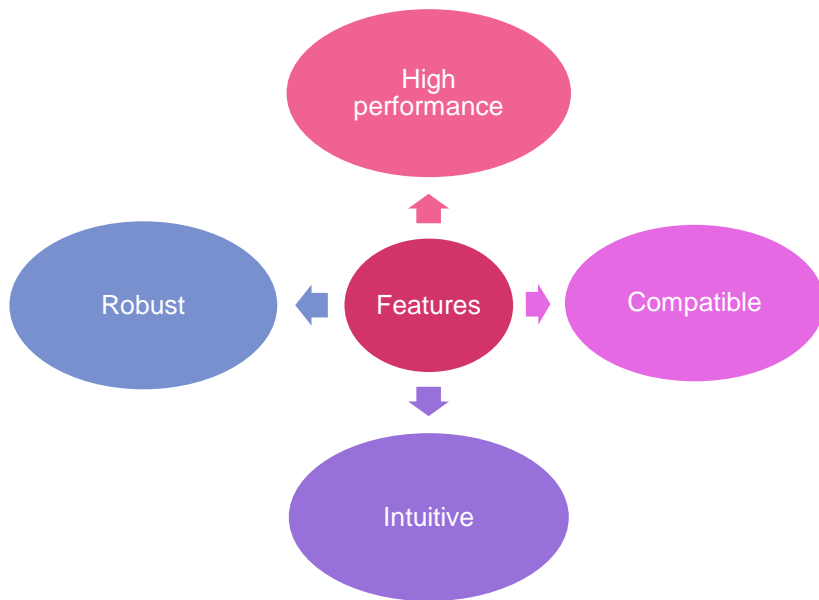


# Introduction to Web APIs



## What is FastApi?

- According to FastApi documentation, FastApi is a “*modern high-performance web framework for building APIs with Python 3.6 and above.*”





# Flask API vs Fast API



- ☐ No data validation
- ☐ Does not support Async
- ☐ Has larger community

- ☐ Possess data validation
- ☐ Async code(async/wait) support
- ☐ Has a small community



# Introduction to Web APIs

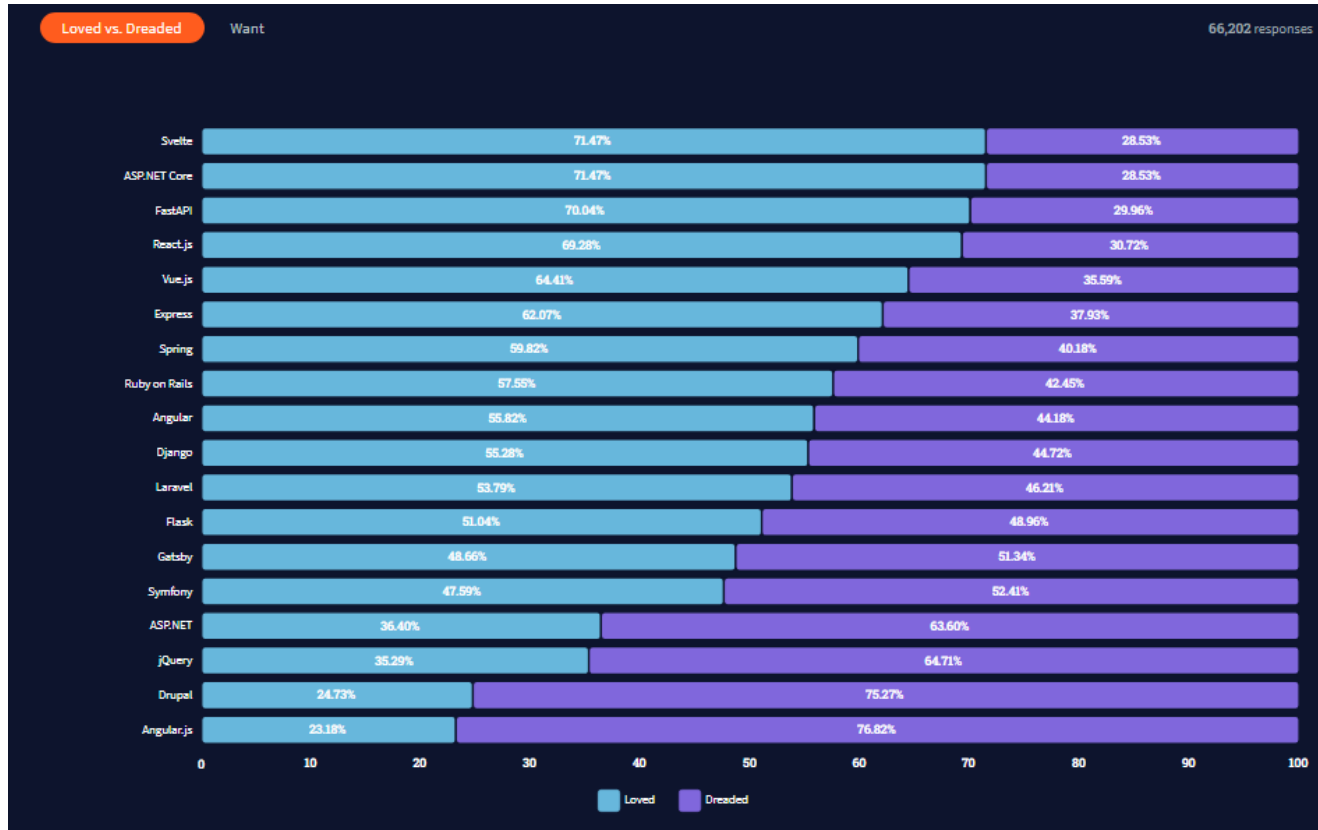


Figure 2: 2021 survey on Loved vs Dreaded Web frameworks [Stackoverflow]

Reference: <https://insights.stackoverflow.com/survey/2021/>

# Tea Break | Quiz

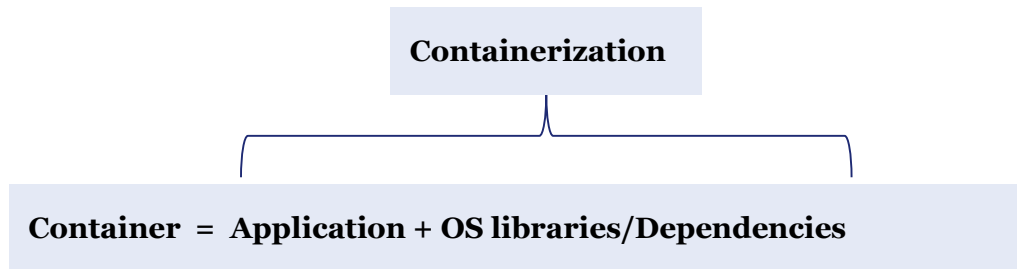
- 1) Which of the following is a way of accessing REST api?  
a) URI using http requests; b) command prompt; c) others
  
- 2) Which of the web api frameworks below handles data validation  
a) Fast api b) Flask api c) All of the above

# Introduction to Docker



## ✓ What is containerization?

According to IBM, “Containerization is the packaging of software code with just the operating system (OS) libraries and dependencies required to run the code to create a single lightweight executable—called a container—that runs consistently on **any** infrastructure.”



Reference:

<https://www.ibm.com/cloud/learn/containerization#toc-what-is-co-r25Smlqq>

# Introduction to Docker



Figure 3: Benefit of containers

Reference:

<https://www.ibm.com/cloud/learn/containerization#toc-what-is-co-r25Smlqq>



# Introduction to Docker

- ✓ What is containerization?
- ✓ **Containerization vs Virtualization**

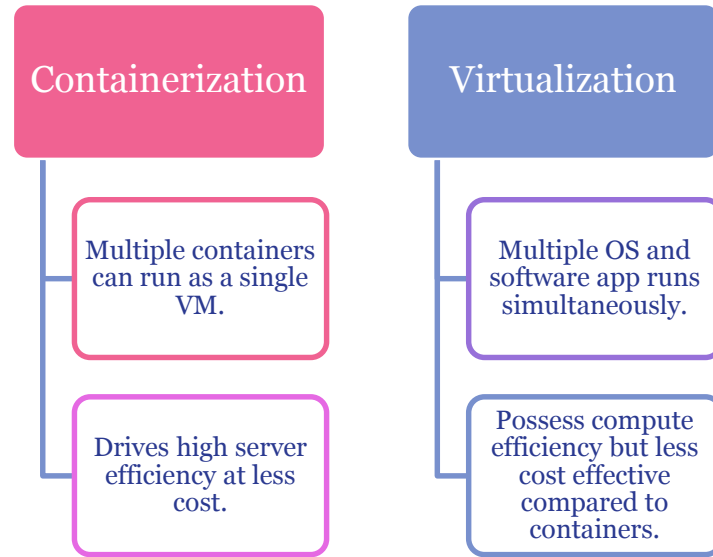
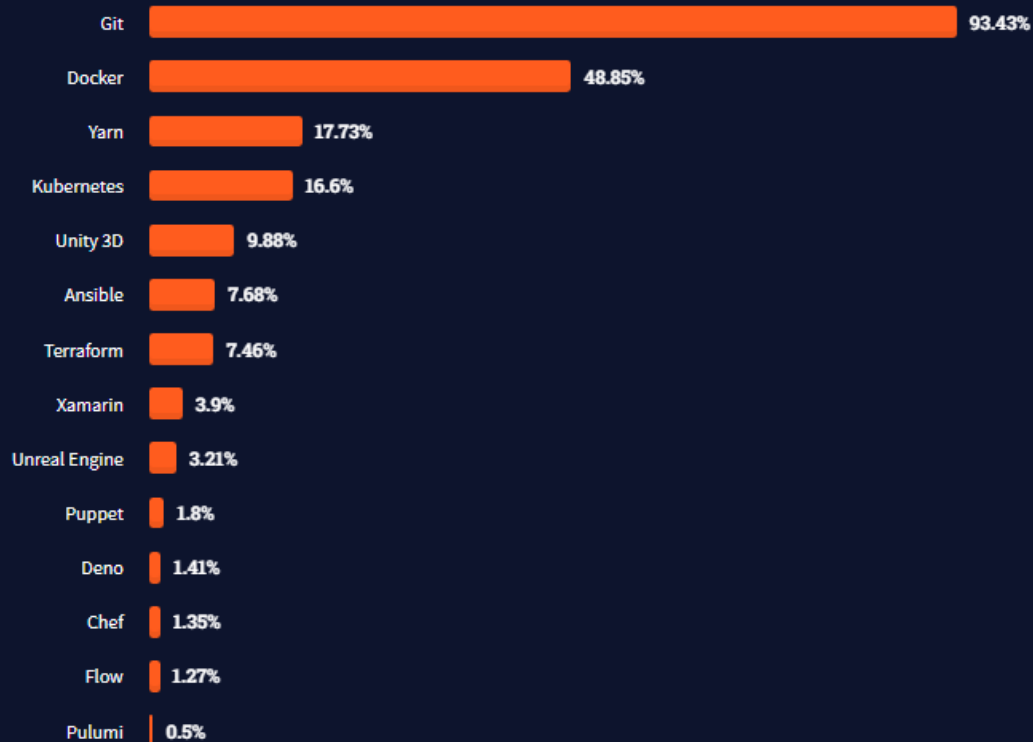


Figure 4: Comparism of container vs virtualization

Reference:  
<https://www.ibm.com/cloud/learn/containerization#toc-what-is-co-r25Smlqq>

# Introduction to Docker



## What is Docker?

According to docker documentation, “Docker is an open platform for developing, shipping, and running applications.”

Reference: <https://insights.stackoverflow.com/survey/2021/>  
<https://docs.docker.com/get-started/overview/>

Figure 5: 2021 Survey result on ‘Other’ tools used by developers [StackOverflow]

# Architecture & component of docker

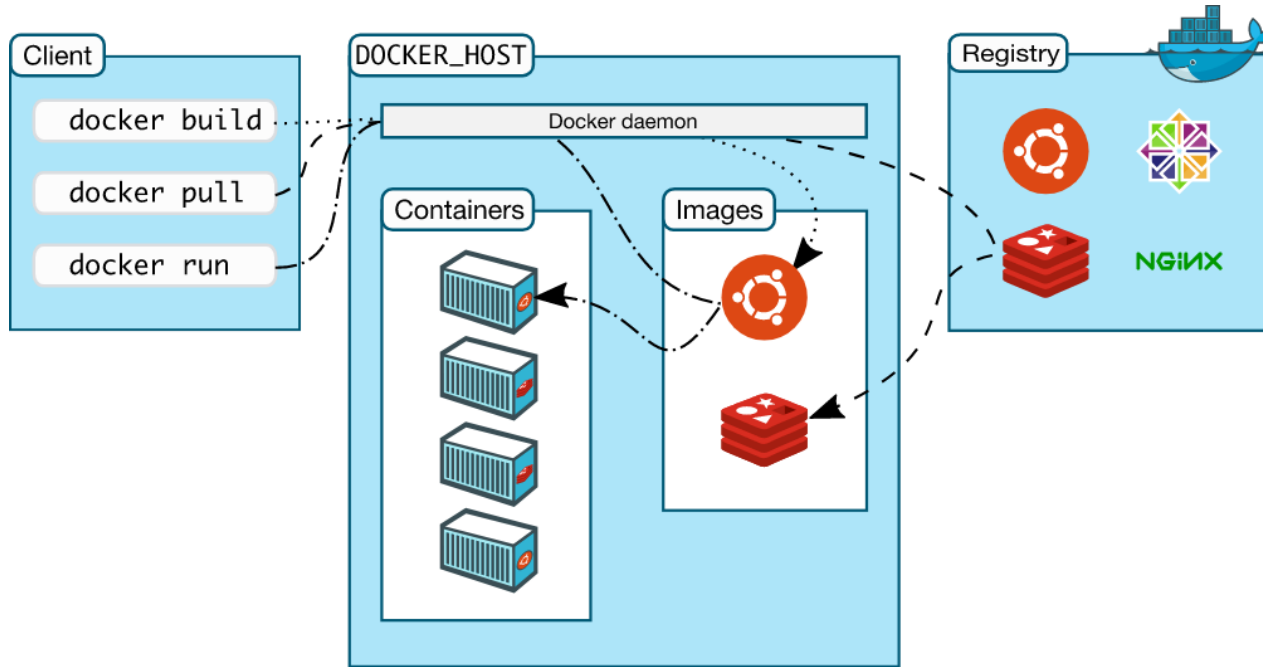
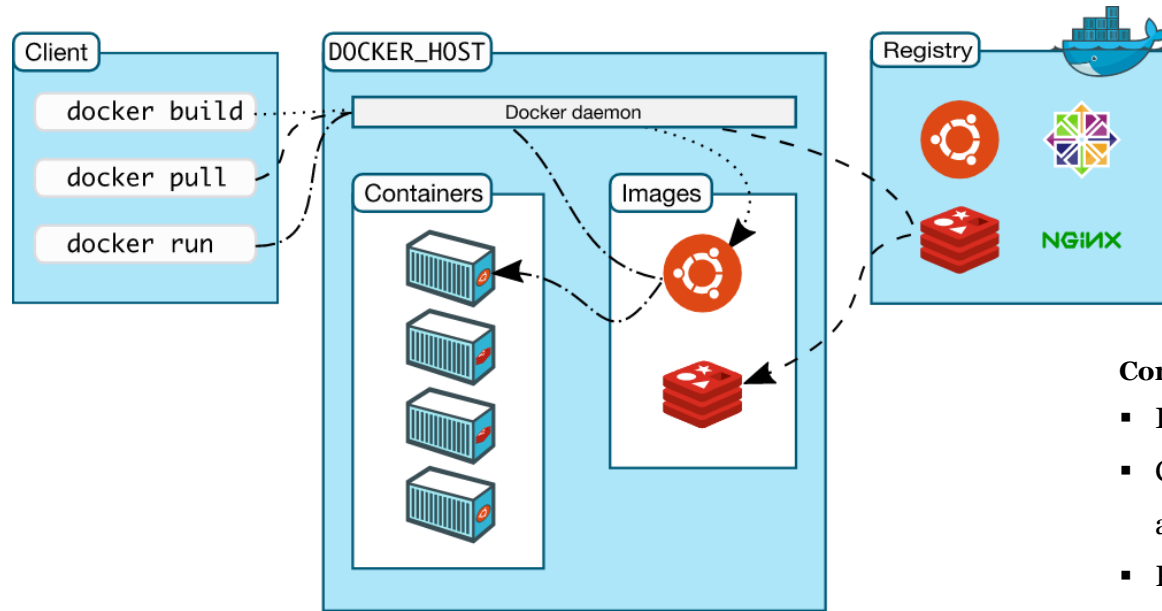


Figure 6: An architectural diagram of docker

# Architecture & component of docker



## Components of Docker

- Images:- the template for creating containers.
- Containers:- Live instance of images that allows app to run.
- Registries:- repositories for docker images.
- Docker Engine:- Daemon, Client, and REST API

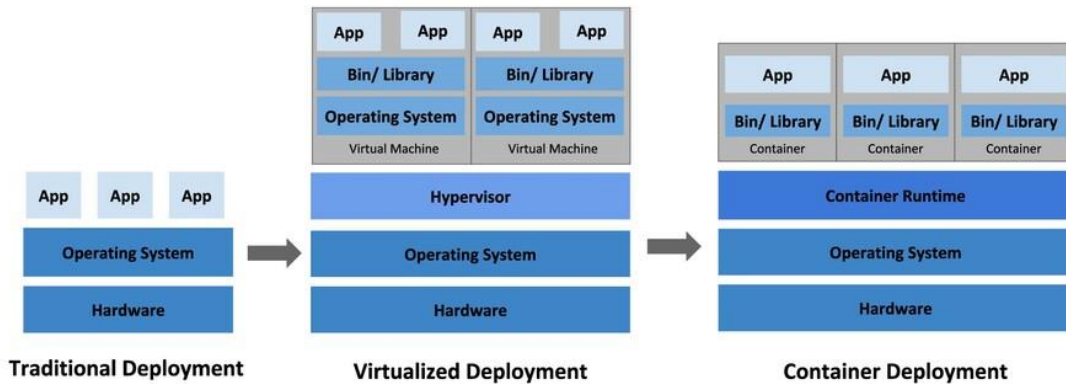
Figure 5: An architectural diagram of docker



# Architecture & component of docker



## WHY DO WE NEED CONTAINERS



<https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/>

Figure 6: Traditional deployment vs Virtualized Deployment vs Container Deployment

Reference: <https://speakerdeck.com/helayoty/kubernetes-101-for-java-developers>

# First Encounter with Docker



## Playing with Docker images:

- How to list/remove images
  - `docker image ls/ docker image rm --tag <image repo>:<image tag>`
- How build images
  - `docker image build -t <image repo>:<image tag>`
- How to tag images
  - `docker image rm --tag <image repo>:<image tag> .`
- How to understand image layers
  - `docker image history <image repo>:<image tag>`
- How to develop executable images
  - `?????`
- Sharing images online
  - `docker login`
- `docker --version`
- `docker-compose --version`

## Playing with Docker containers:

- Docker Installation
  - `pip install docker`
- How to remove containers
  - `docker container rm <identifier(id)>`
- How to list containers
  - `docker ls (--all)`
- How to stop/kill containers
  - `docker container stop/kill <identifier(id)>`
- How to re(start) containers
  - `docker container re(start) <identifier(id)>`
- How to create containers
  - `docker container create --publish 8080:80 <directory/script>`

Reference: <https://insights.stackoverflow.com/survey/2021/>  
<https://docs.docker.com/get-started/overview/>

# First Encounter with Docker



## How to write a docker file.

- Create a docker file: `$ touch Dockerfile`

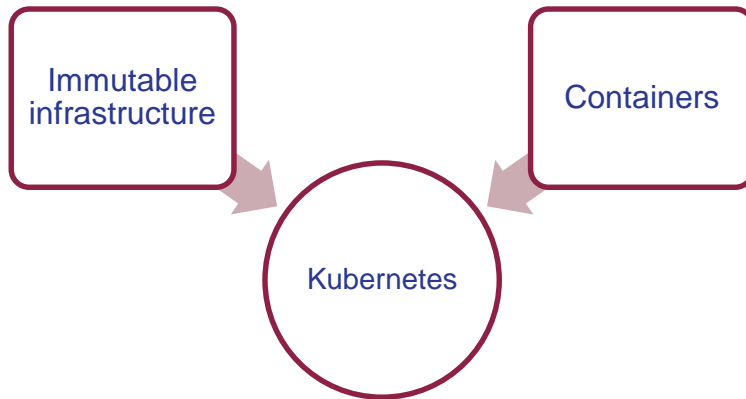
## Using docker compose to compose a project:

- Basics of docker-compose
- How to start services and list services
- How to stop running services
- Create a yaml file
- Docker-compose up
- Docker-compose stop

# Introduction to kubernetes



## WHAT IS KUBERNETES?



“Kubernetes as a container or microservice platform that orchestrates computing, networking, and storage infrastructure workloads. Because it doesn’t limit the types of apps you can deploy (any language works), Kubernetes extends how we scale containerized applications so that we can enjoy all the benefits of a truly immutable infrastructure.”

# Introduction to kubernetes



## ✓ K8s Architectural Components

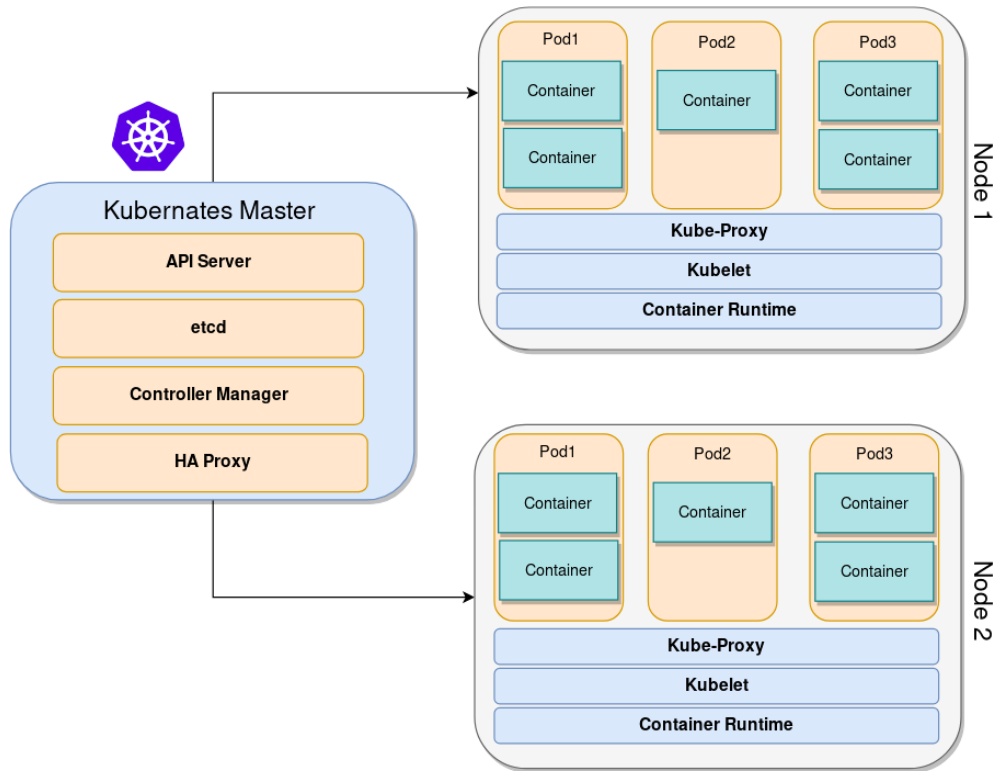


Figure 7: The Architecture of Kubernetes

# Introduction to kubernetes



## ✓ Advantages of Kubernetes

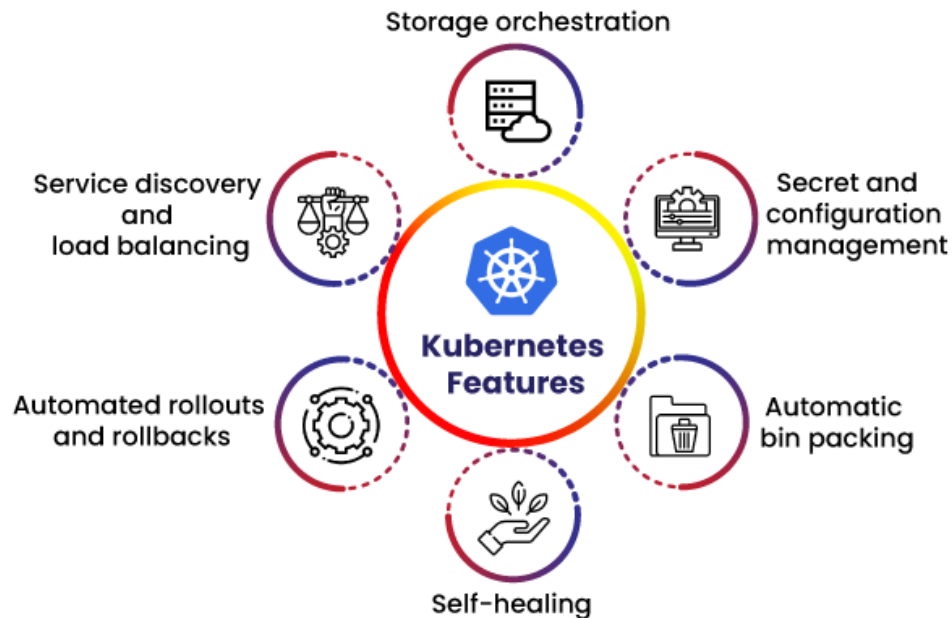


Figure 8: Merits of using kubernetes

Reference: <https://www.veritis.com/services/kubernetes/>

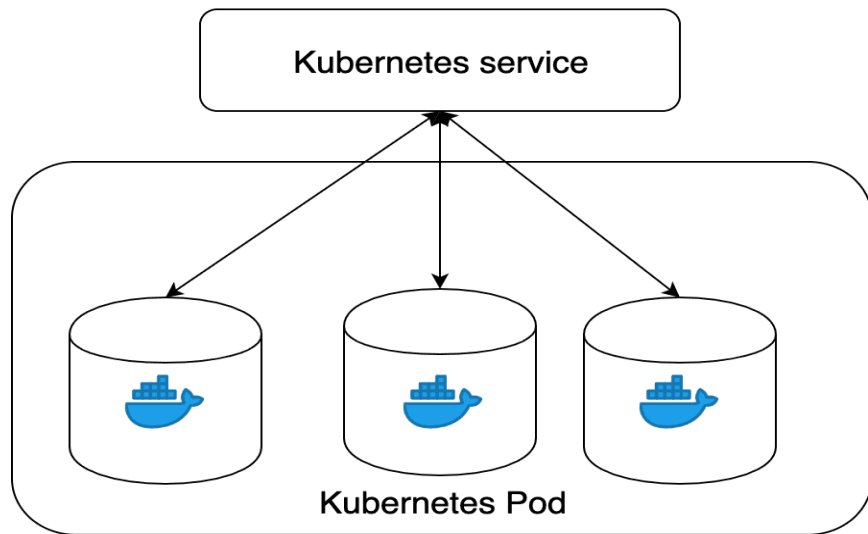
# Introduction to kubernetes



## ✓ Limitations of kubernetes

- *Complexity (manage kubernetes server and container server)*
- *Running costs(hosting, staffing cost, etc)*
- *Just a tool, not a solution(its not a full solution like heroku)*

# Components of kubernetes

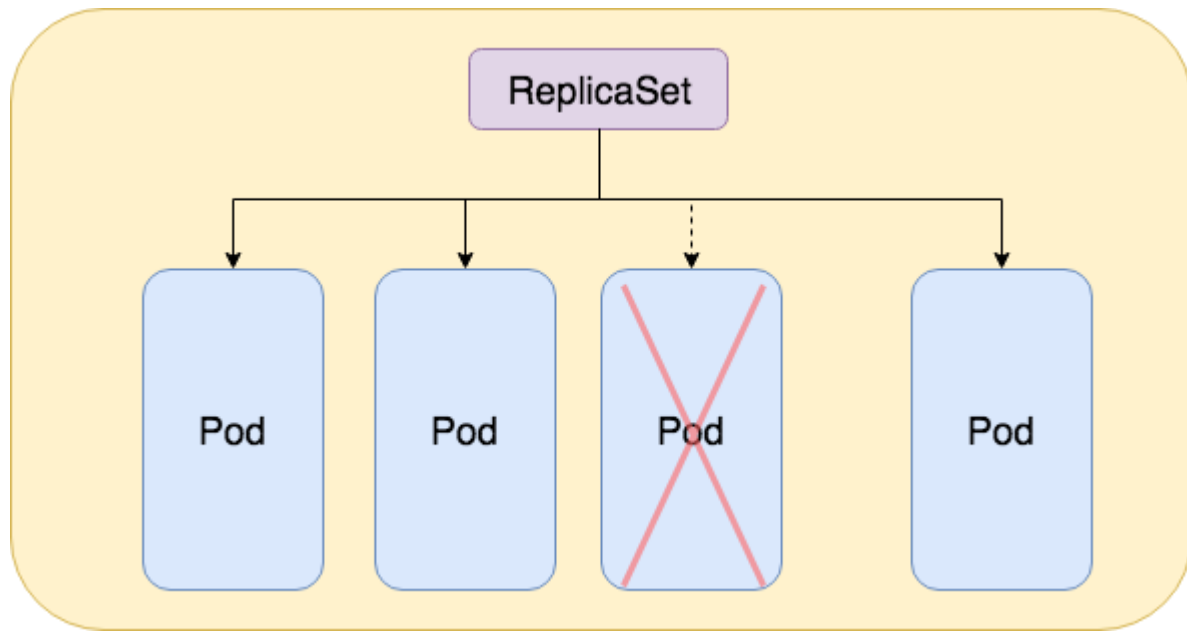


A group of running containers is referred to as **Pod**

Figure 9: A kubernetes pod



# Components of kubernetes

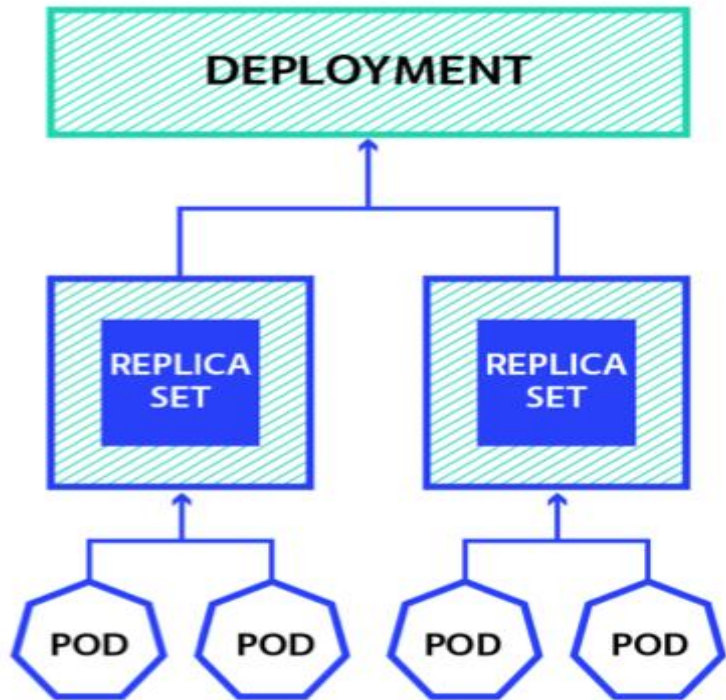


The ability of given number of pods to run at the same is enabled by **Replica Set**.

Figure 10: A kubernetes Replicaset

Reference: [https://medium.com/@jiamin\\_ning/build-your-first-kubernetes-service-with-replicaset-7c37d9be689c](https://medium.com/@jiamin_ning/build-your-first-kubernetes-service-with-replicaset-7c37d9be689c)

# Components of kubernetes

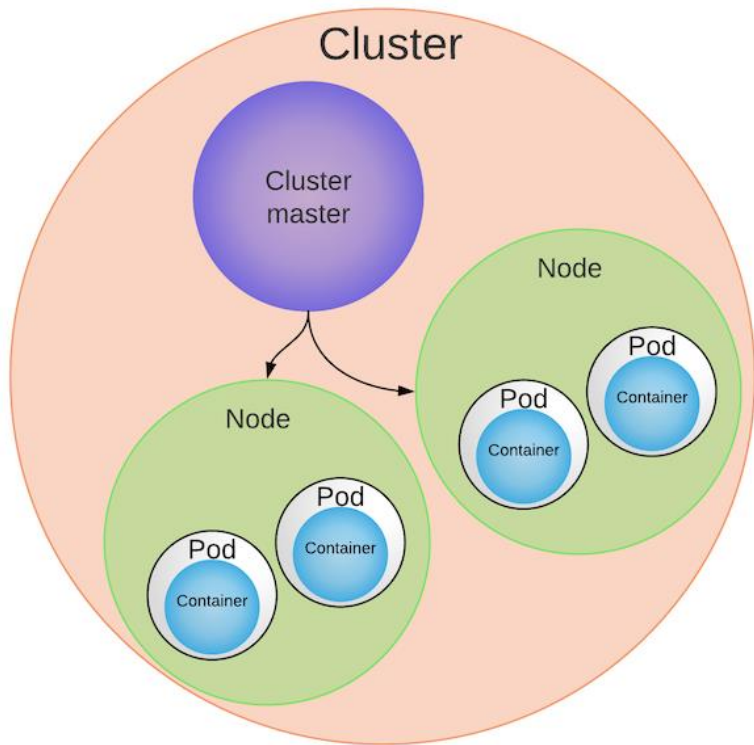


The rollback and rollout of pods is referred to as **Deployments**.

Figure 11: A kubernetes deployment

Reference: <https://thenewstack.io/deletion-garbage-collection-kubernetes-objects/>

# Components of kubernetes

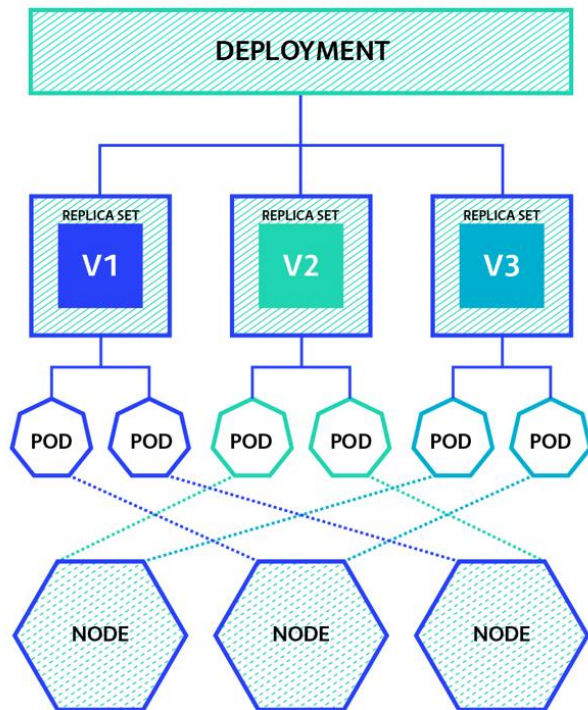


The entire system or all collective running machine is referred to as a **Cluster**.

So a group of nodes forms a cluster.

Figure 12: A kubernetes cluster

# Components of kubernetes



The machines in a cluster are referred to as **Nodes**.

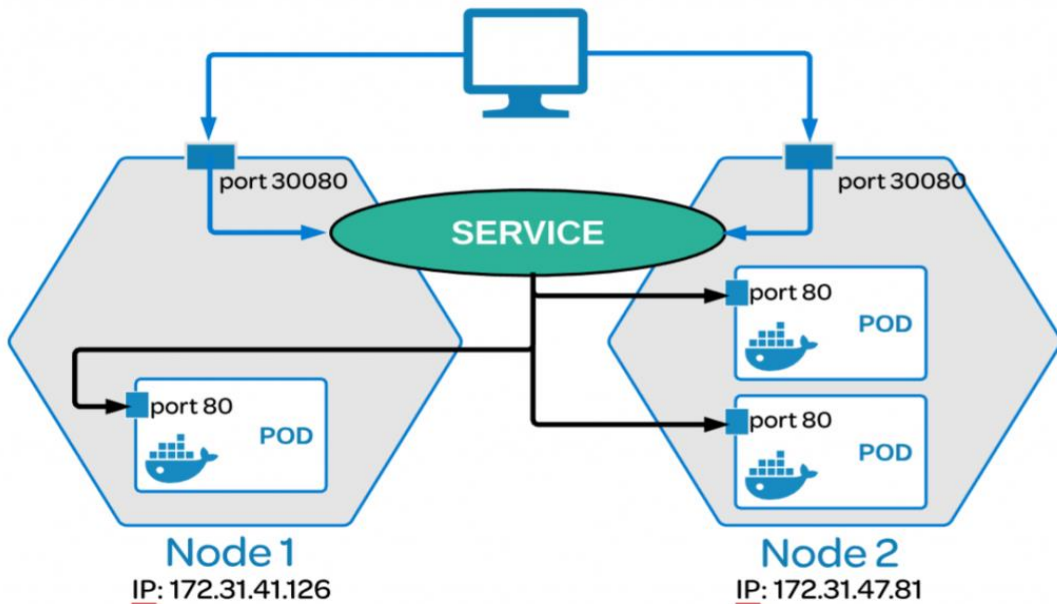
Nodes can be a **worker** or a **master**. The control panel is run by the master node. While the worker node consist of software to run containers and its managed by the control plane.

Figure 13: Kubernetes Nodes

# Components of kubernetes

## Kubernetes Service

A service allows you to dynamically access a group of replica pods.



How do we access our running pods?

Figure 14: The life cycle of a typical deep learning project

Reference: <https://medium.com/avmconsulting-blog/single-and-multi-port-service-in-kubernetes-k8s-8b08529d9ba6>

# Tea Break | Quiz

1) Which of the following allows one to access pods?

a) Kubernetes service, b) containers, c) pods

2) A group of nodes is refer to as:

a) Replicaset, b) Cluster, c) None of the above

3) Nodes can be categorize as:

a) Worker and walker node, b) Cluster and walker node, c) Worker and master node

# Introduction to kubernetes

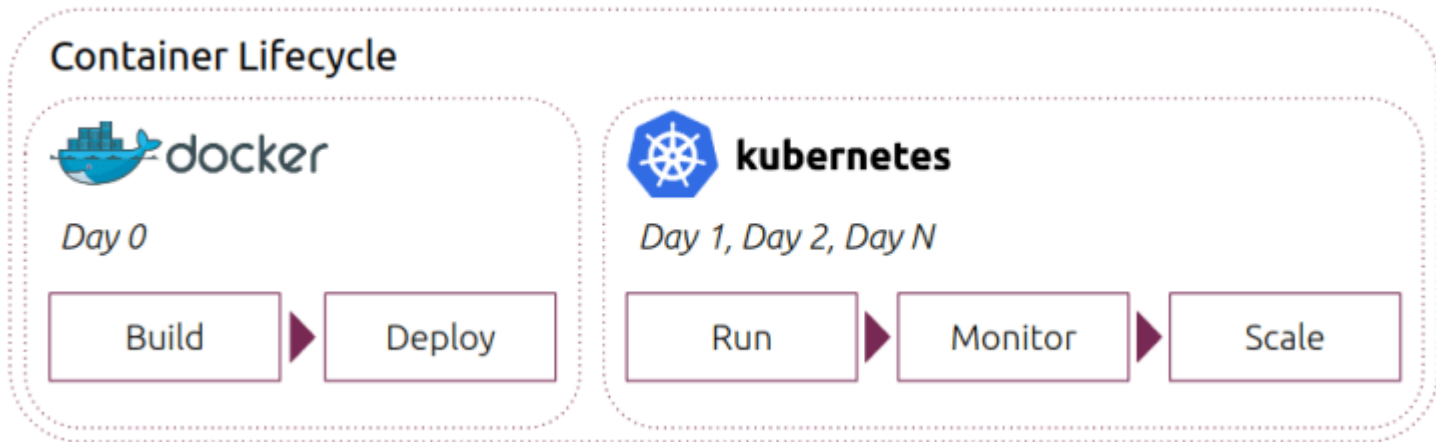
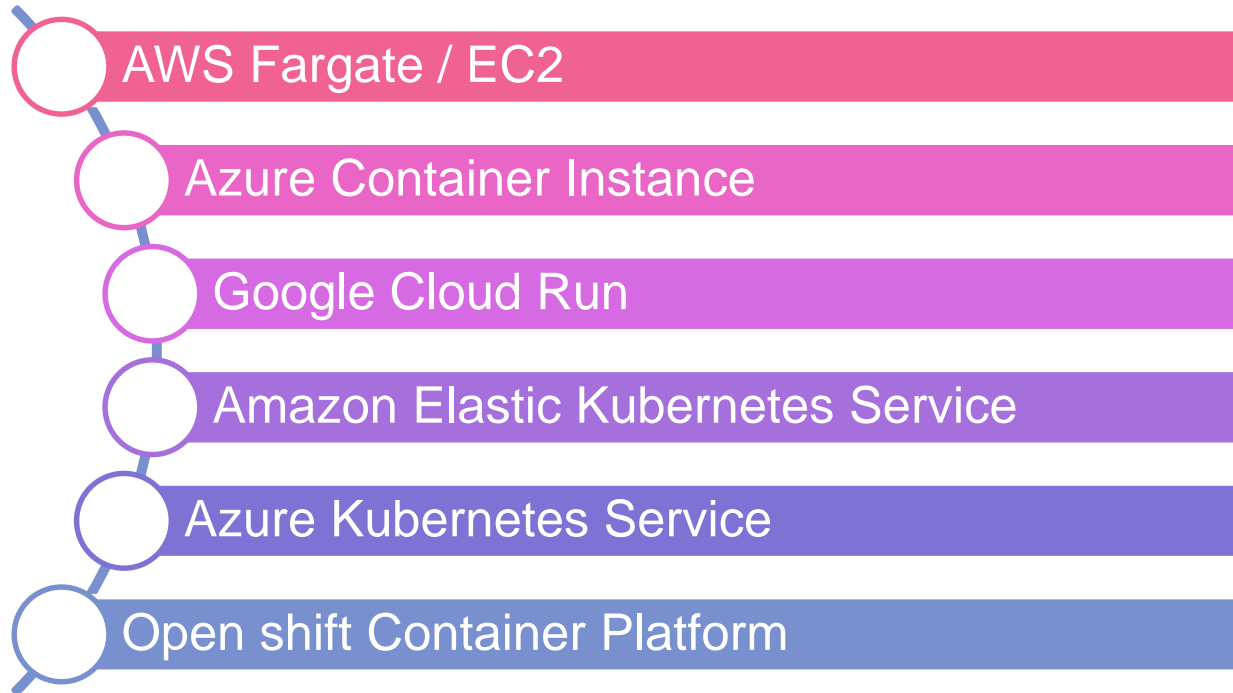


Figure 15: The life cycle of a container

# Introduction to kubernetes- Alternative options





# Deployment of container on kubernetes (EKS, GKE, etc)

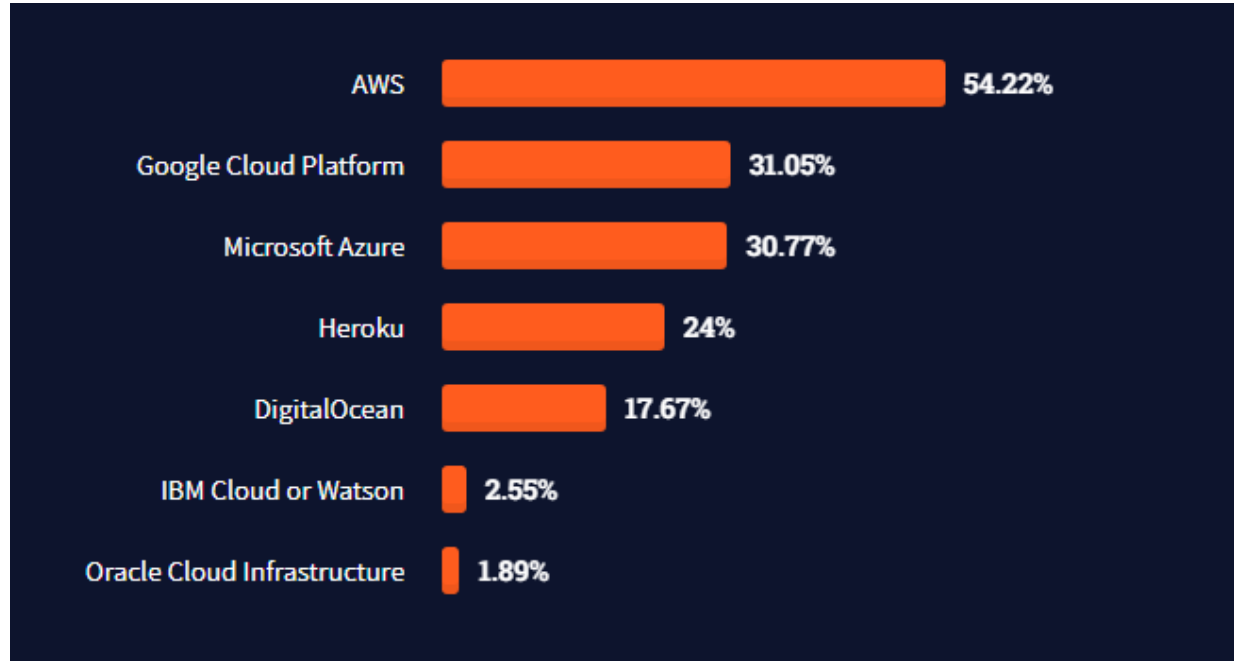


Figure 16: Stackoverflow survey on most used cloud services by developers

# An introduction to automating ML deployment workflow

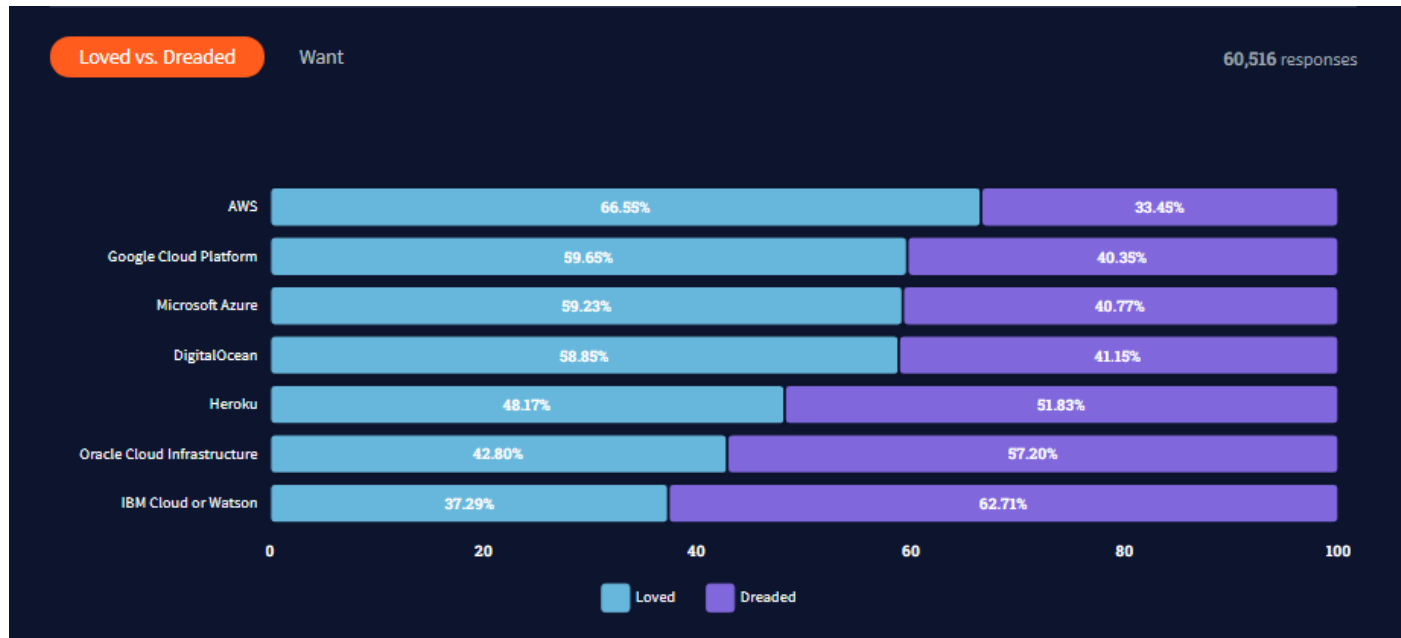


Figure 17: Stackoverflow survey on most loved vs most dreaded

# Containerize a python web app using docker



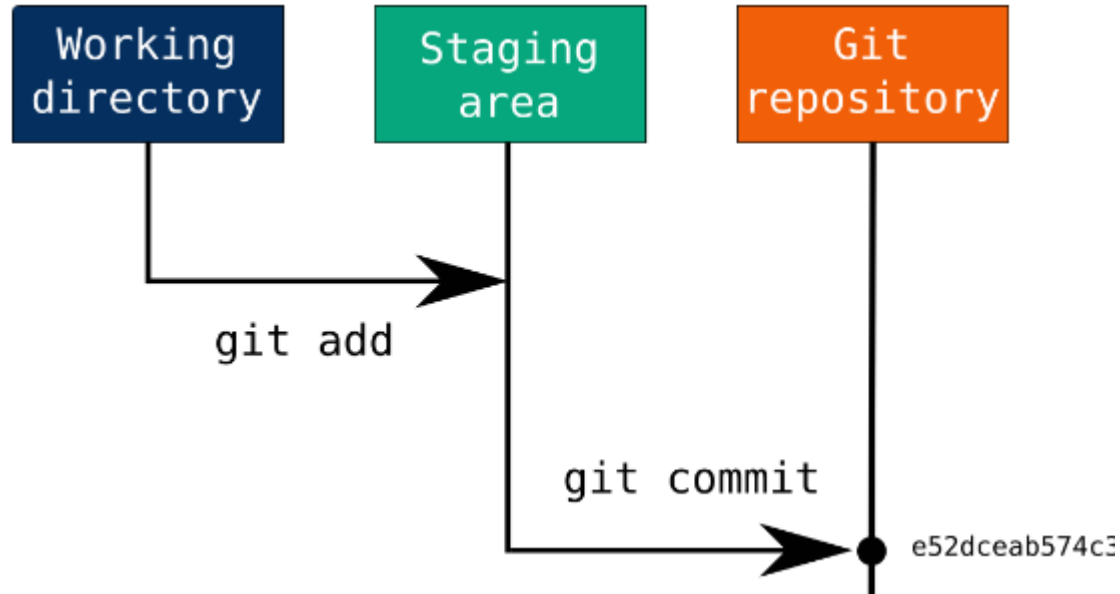
- 1) Create a project folder
- 2) Clone repo
- 3) Create a virtual environment
- 4) Install Flask/Fast
- 5) Develop a module(or script to solve a problem import it into the Flask/Fast app.
- 6) Test it using postman
- 7) Create a dockerfile at the root level of the project folder
- 8) Create a yaml file at the root level of the project folder
- 9) Containerize the project folder, try to push to docker hub.
- 10) Apply docker-compose to run application
- 11) Push to github.

Reference:

[https://miro.medium.com/max/700/1\\*rCyvV8hAhAhqNjkL7Wi7g.png](https://miro.medium.com/max/700/1*rCyvV8hAhAhqNjkL7Wi7g.png)



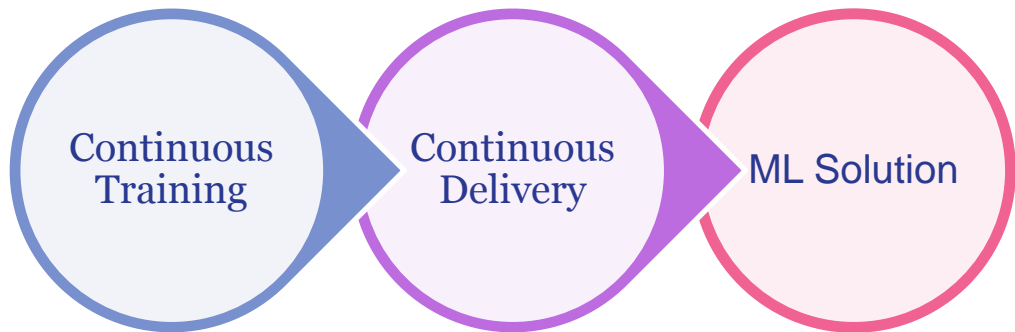
# Containerize a python web app using docker



Reference:

[https://miro.medium.com/max/700/1\\*rCyvV8hAhAhqNjkl7Wi7g.png](https://miro.medium.com/max/700/1*rCyvV8hAhAhqNjkl7Wi7g.png)

# An introduction to ML deployment workflow



# An introduction to ML deployment workflow

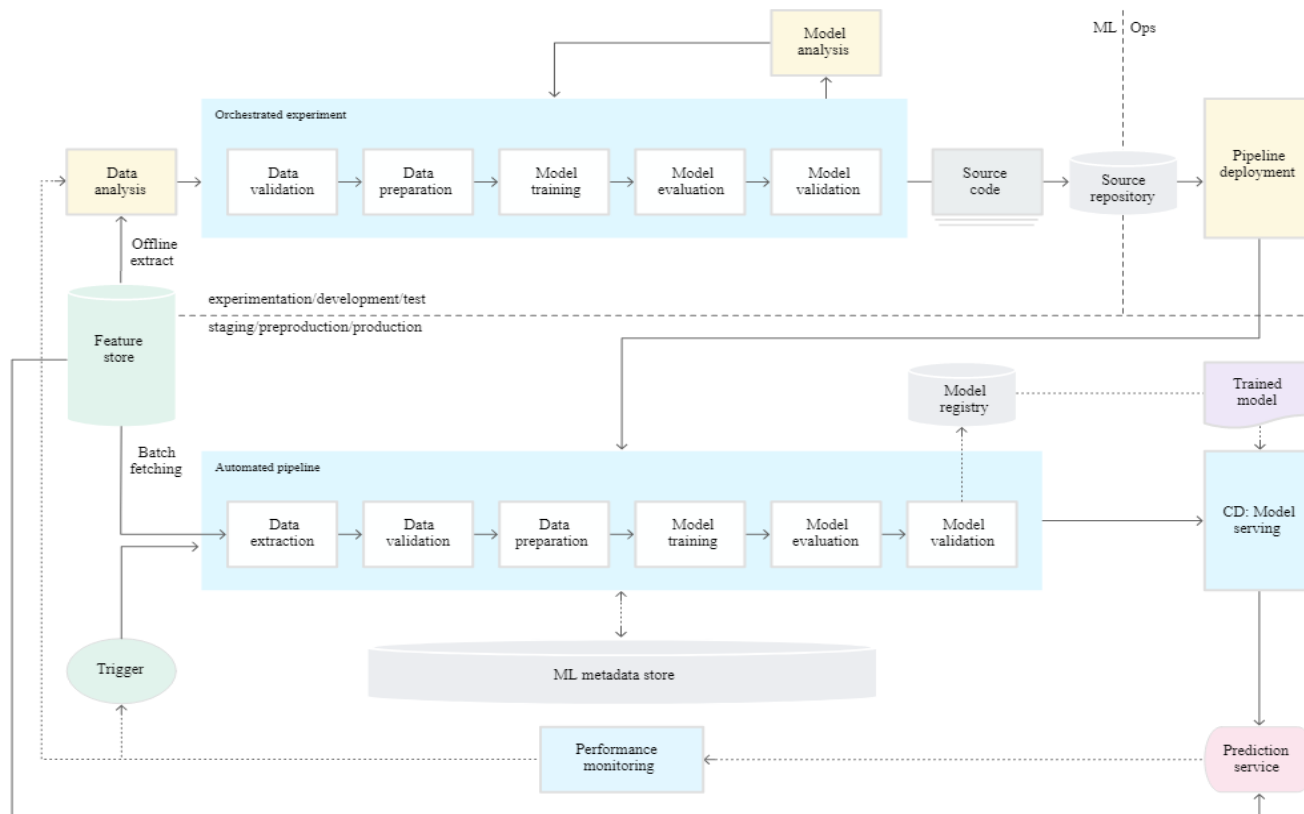


Figure 17: ML pipeline automation for CT

Reference: <https://bit.ly/3IsaoLi>

# An introduction to ML deployment workflow

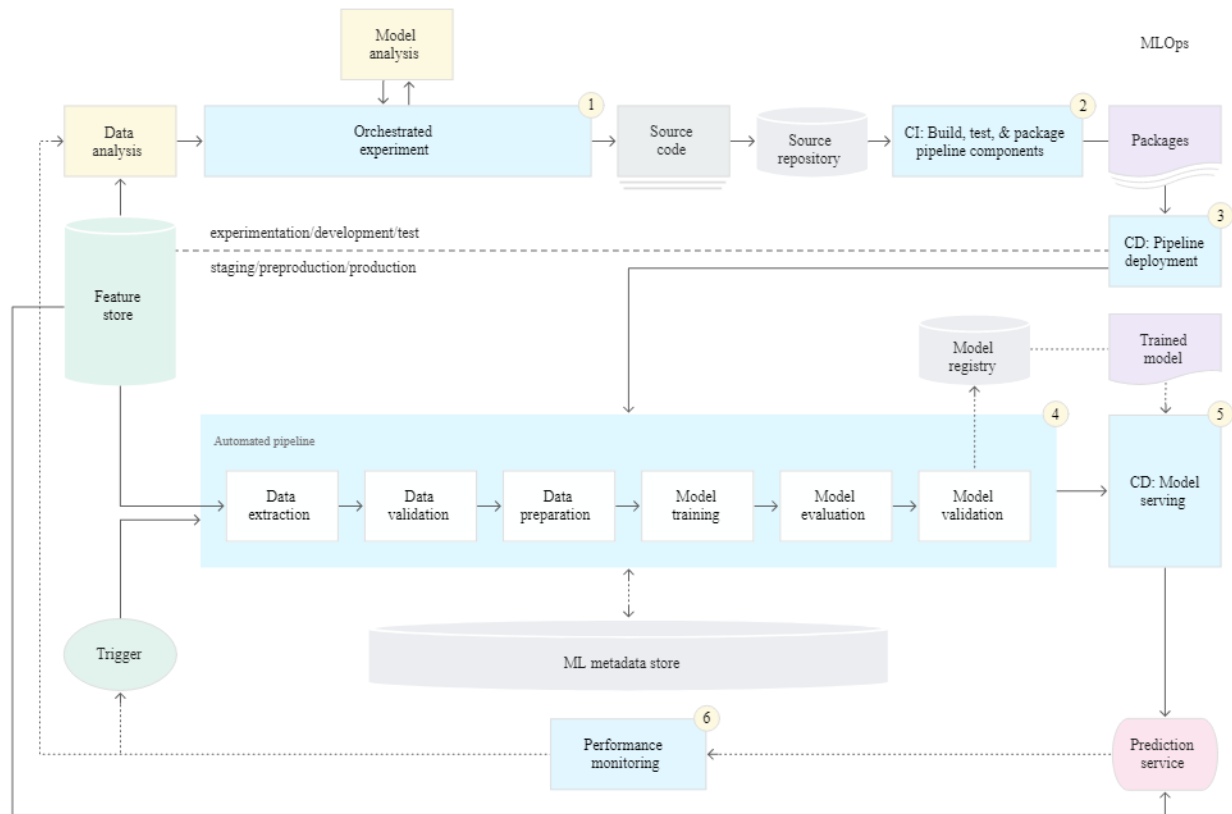


Figure 18: CI/CD and automated ML pipeline

Reference: <https://bit.ly/3IsaoLi>

# An introduction to ML deployment workflow

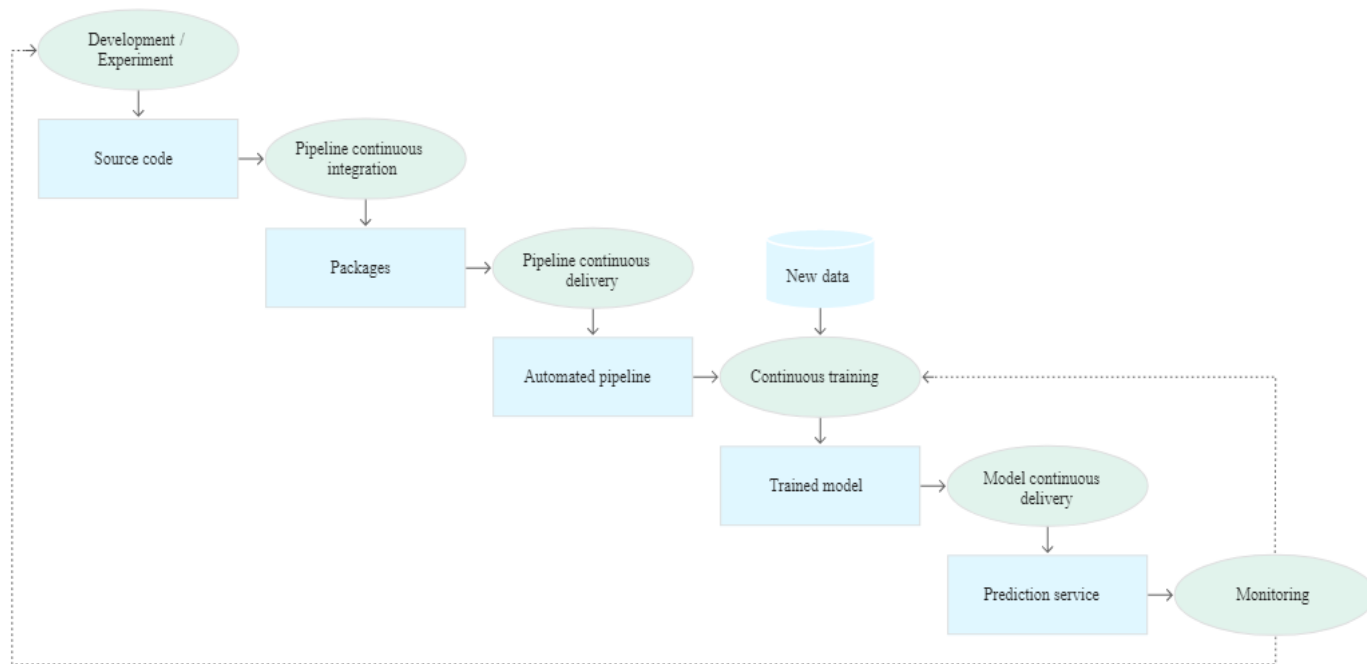


Figure 18: Stages of the CI/CD automated ML pipeline



# RECOMMENDED READING

- <https://stackify.com/docker-build-a-beginners-guide-to-building-docker-images/>
- <https://pythonbasics.org/flask-rest-api/>
- <https://www.digitalocean.com/community/cheatsheets/how-to-use-git-a-reference-guide>
- <https://docs.github.com/en/actions/learn-github-actions>
- <https://towardsdatascience.com/a-gentle-introduction-automating-machine-learning-pipelines-71f97192eabf>
- <https://cloudacademy.com/blog/how-to-deploy-an-app-from-github-with-aws-codedeploy/>
- [http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create\\_deploy\\_docker\\_image.h  
tml#create\\_deploy\\_docker\\_image\\_dockerrun](http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create_deploy_docker_image.html#create_deploy_docker_image_dockerrun)

# ASSIGNMENT

1) Create a program that reads the length and width of a farmer's field from the user in feet. Display the area of the field in acres.

**Hint:** There are 43,560 square feet in an acre. [source: The python workbook]

2) Containerize and push your app to github.

- Build your docker image
- Version the docker image to Docker Hub

# REFERENCE



- ❑ <https://harness.io/blog/how-to-do-continuous-delivery-for-machine-learning-systems/>
- ❑ [https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning#mlops\\_level\\_2\\_cicd\\_pipeline\\_automation](https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning#mlops_level_2_cicd_pipeline_automation)
- ❑ <https://www.slideshare.net/AmazonWebServices/managing-your-application-lifecycle-on-aws-continuous-integration-and-deployment>
- ❑ <https://www.bmc.com/blogs/docker-101-introduction/>
- ❑ <https://docker-curriculum.com/#getting-started>





Q & A