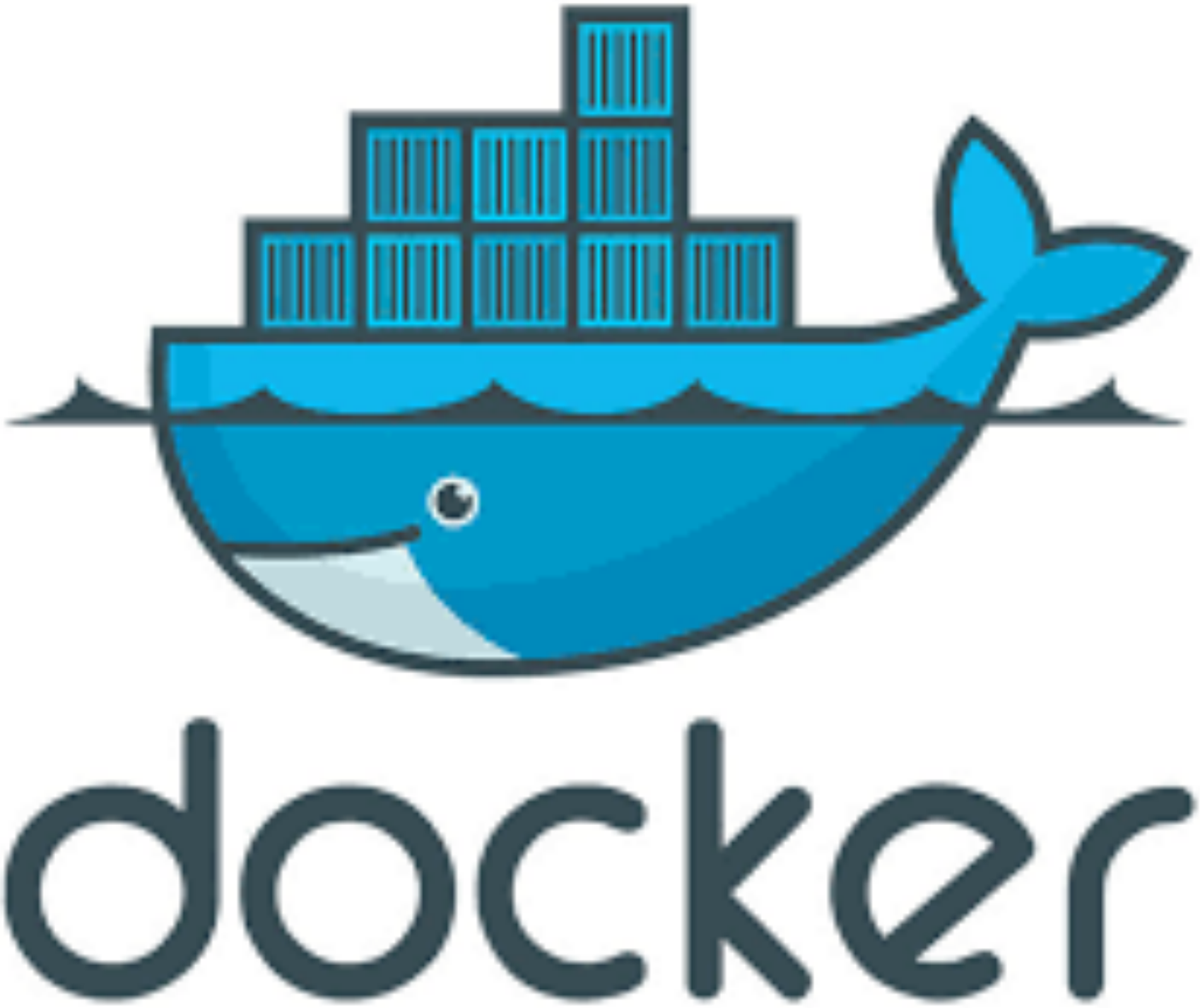
Docker & Containerization For Big Data


docker

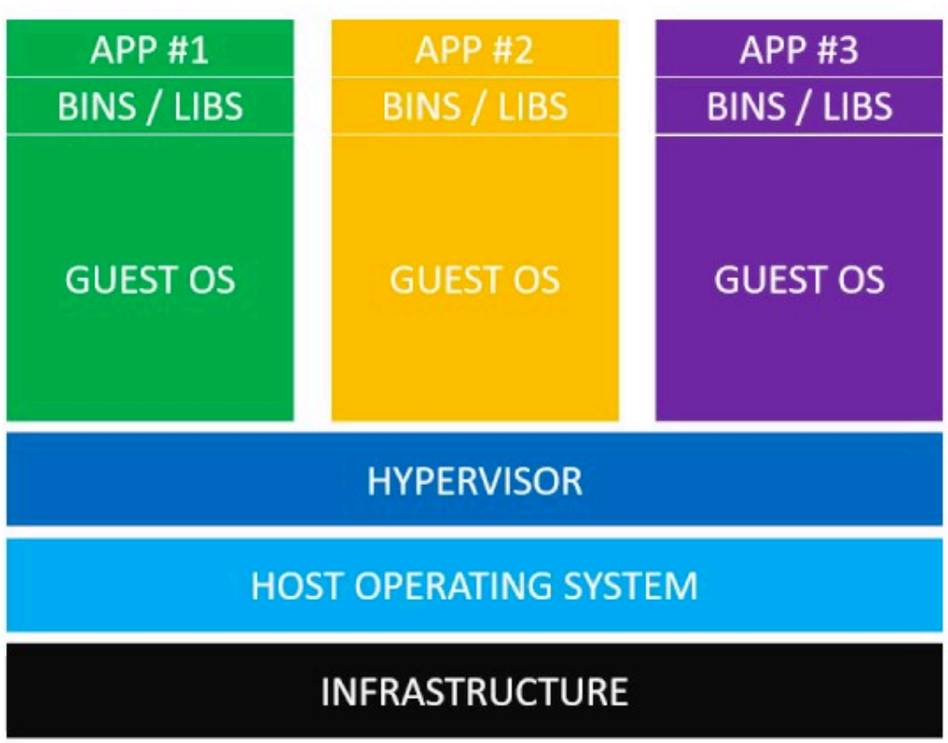
Containers / Containerization has been around for a long time, but really came into wide usage with Docker

A **Docker container** is a lightweight virtual computer.

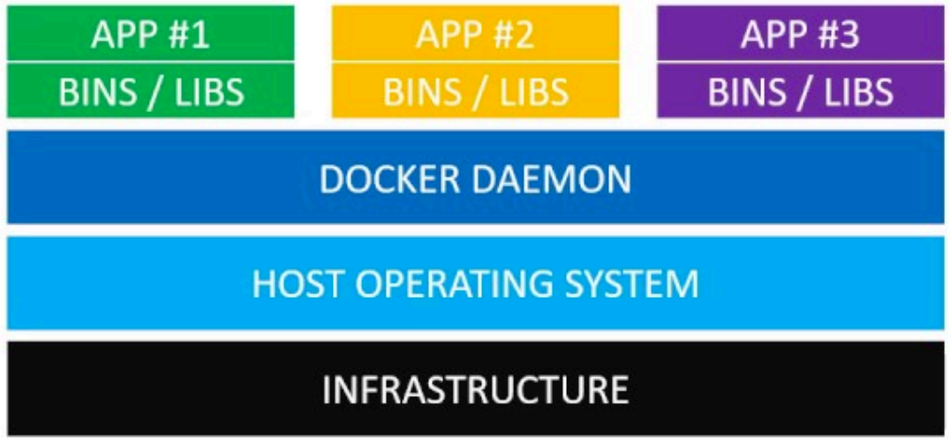
It shares the host operating system and kernel, rather than installing a new one. This is good (more efficient), but occasionally bad (security threat).

Docker Container

More efficient (lighter), with faster startup than Virtual Machines (VM)



Virtual Machines



Docker Containers

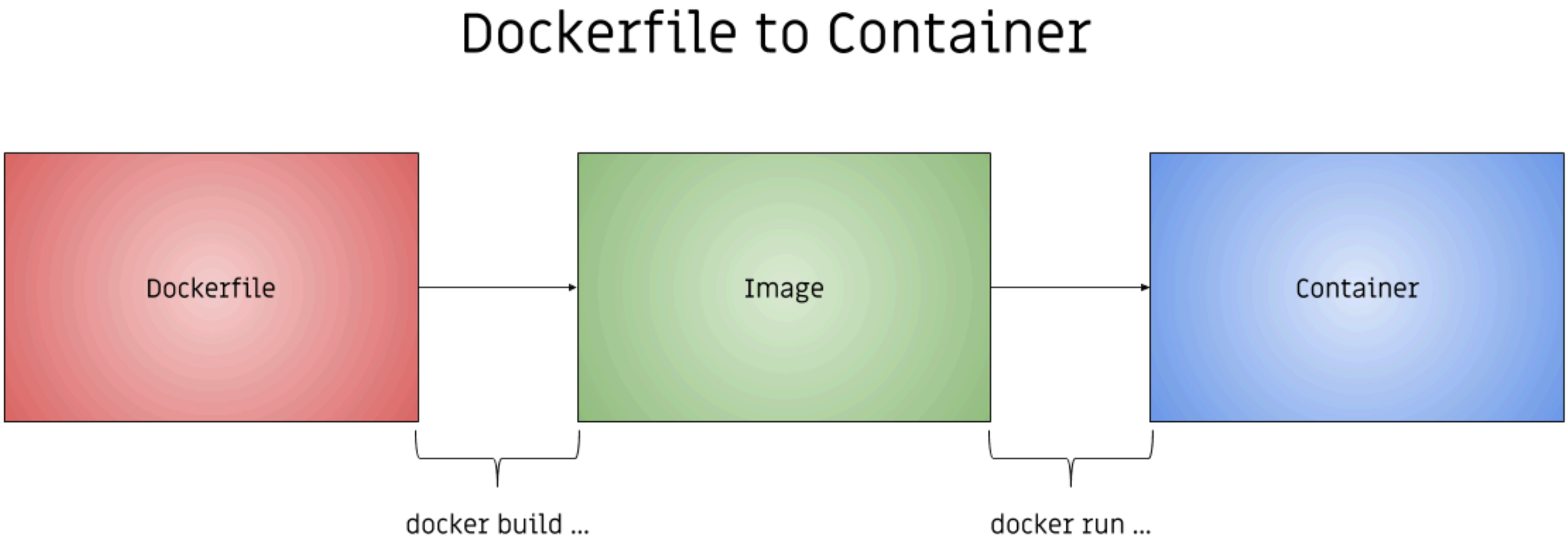
Docker Words

- **Image -** an executable package that includes everything you need to run an application. This includes the code, libraries, environmental variables, configuration files, even the operating system.

- **Container -** a run-time instance of an image. That is, what an image becomes when executed in memory.

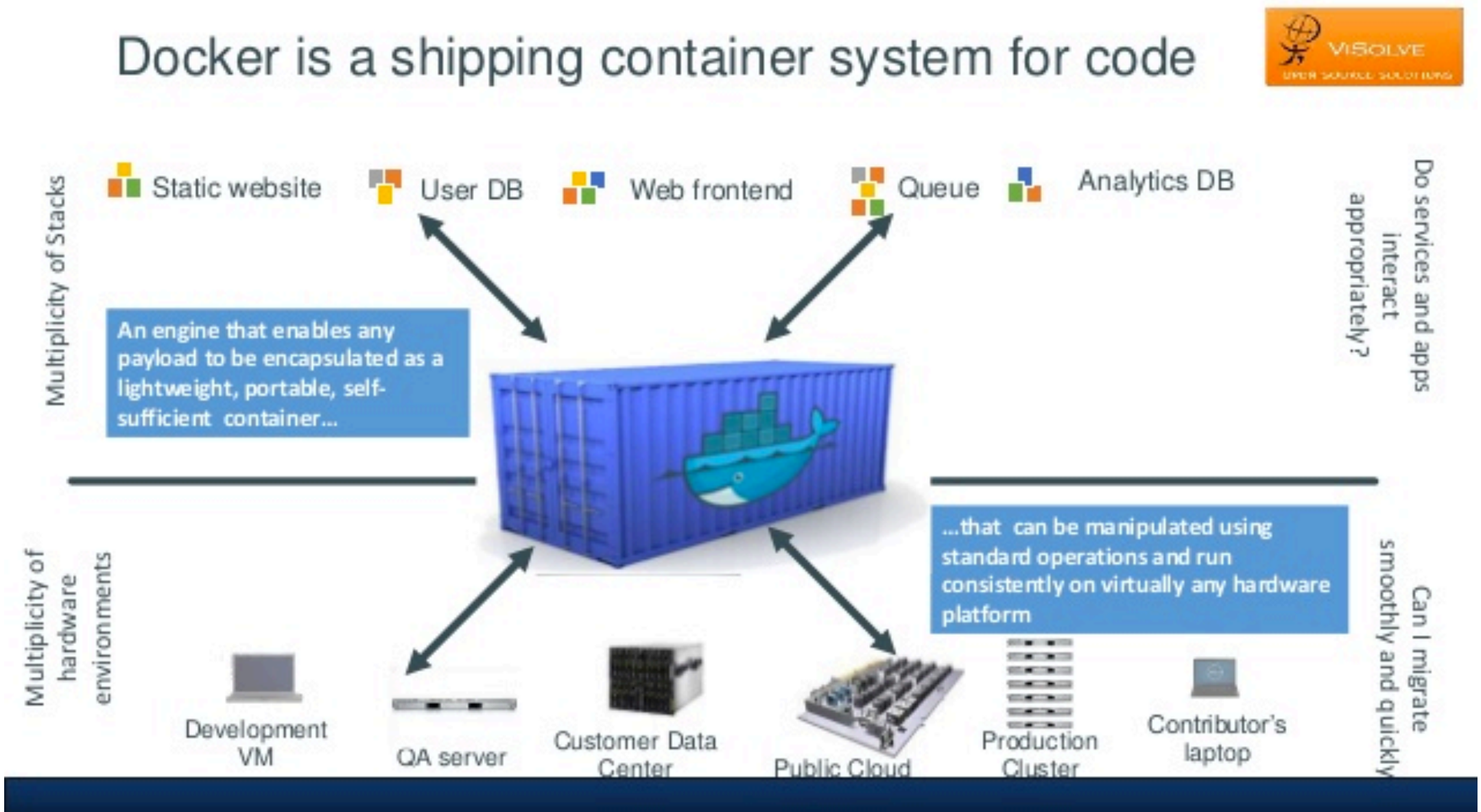
- **Service -** (less critical) how a container runs in production.

Docker Words



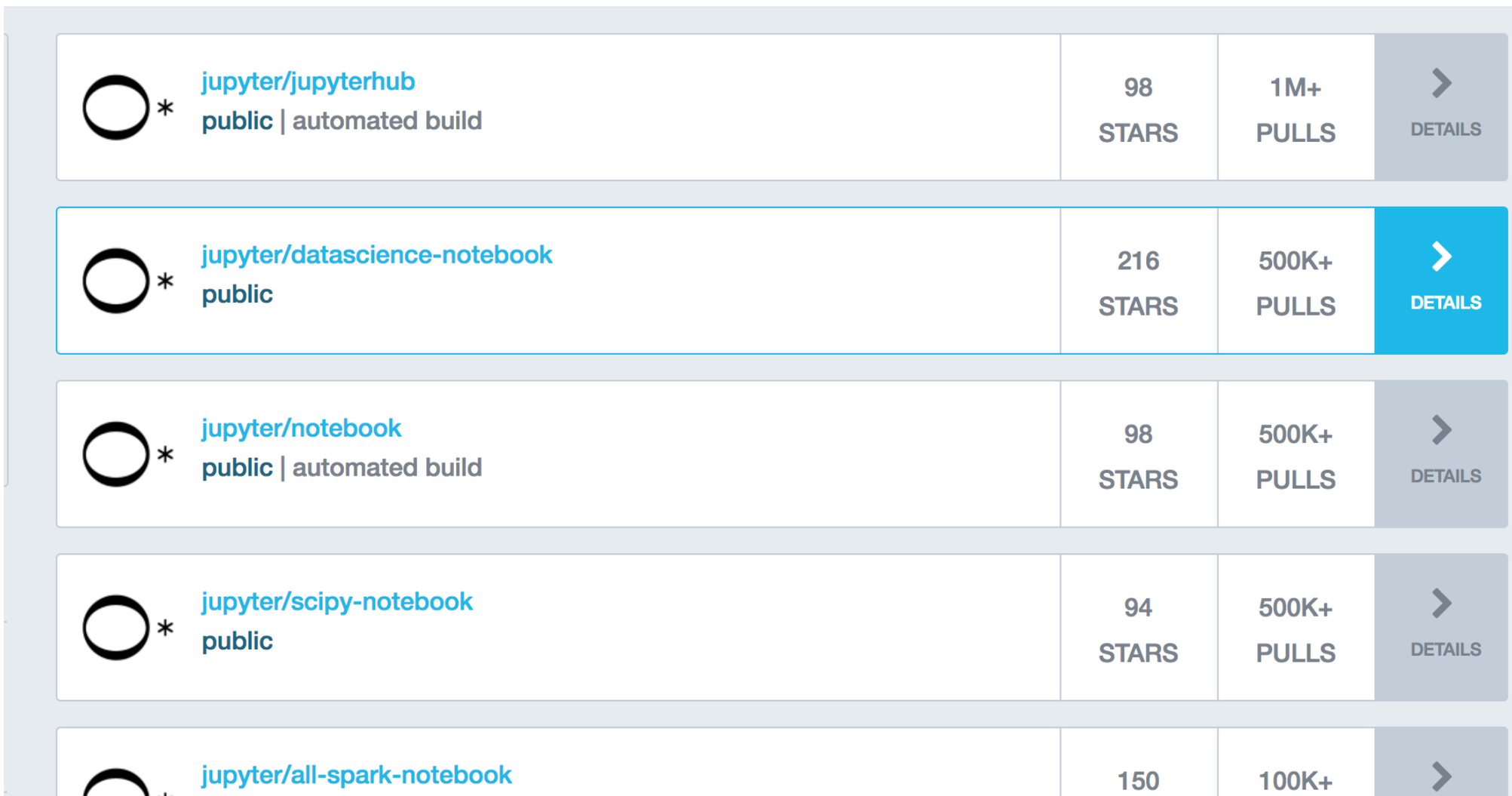
Docker for Big Data

Agnostic to underlying systems, which means it is easy to migrate between cloud providers and different environments (your laptop vs. a co-worker's vs production server).



Docker is Stackable and Portable

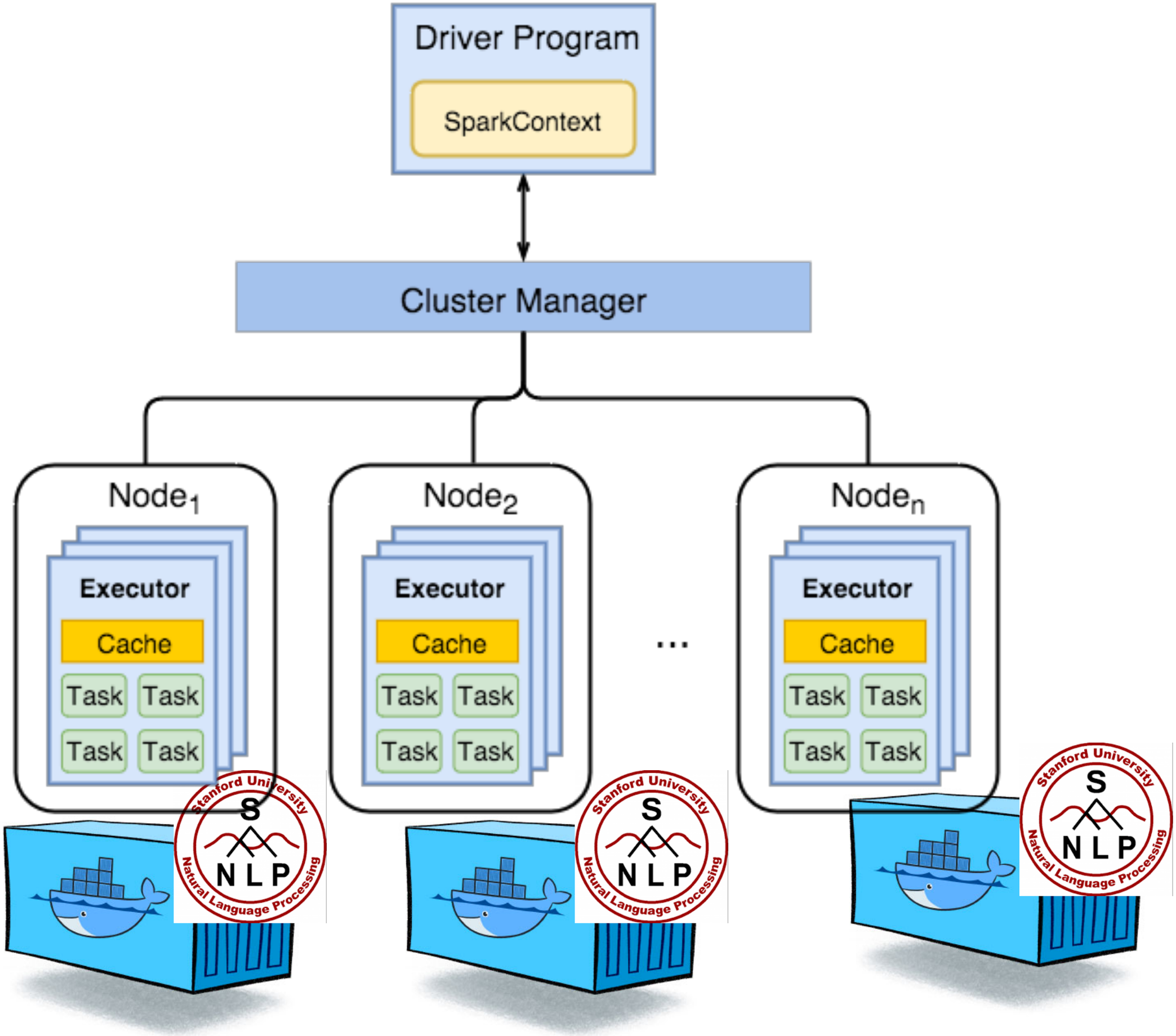
Solutions developed in other environments will work in yours, and you can build on top of them. Enter DockerHub. [Jupyter's DockerHub Repo]



Docker for Big Data

Spark + Open Source Routing Machine in Docker
https://github.com/UI-Research/spark-osrm
https://hub.docker.com/u/graham3333/

Stan + R in Docker: Reproducible Stan
https://gitlab.com/jzelner/reproducible-stan


Driver Program
SparkContext
Cluster Manager
Node1
Executor
Cache
Task
Task
Task
Task
Node2
Executor
Cache
Task
Task
Task
Task
...
Noden
Executor
Cache
Task
Task
Task
Task
Stanford University
S
NLP
Natural Language Processing

```python
from pyspark.sql.functions import udf

def map_sentiment(dataline):
  import requests
  import json
  url = 'http://172.17.0.2:9000/?properties=%7B%22annotators%22:%22tokenize,ssplit,sentiment%22,%22outputF
  datatext = dataline[0][2]
  data = requests.post(url, data={"q":datatext}, timeout=60).text
  json_data = json.loads(data)
  sents = [int(x['sentimentValue']) for x in json_data['sentences'] if x['sentimentValue'] is not None]
  sentVal = float(sum(sents))/len(sents)
  return sentVal

map_sentiment_udf = udf(map_sentiment)

df_new = df.withColumn("sentiment_score", map_sentiment_udf("tweet"))
```

Other Nice Docker Things

- Mistakes are less costly. Messing up within a docker container does not break anything else outside the docker container. This is nice for data scientists new to DevOps who want to experiment with environments.

- Non-competitive resource allocation. You can have five Docker containers with dedicated resources to each one (though isolation is not as strong as full virtual machine).

- Relatively easy to test. You can test anything locally and it will work the same as it will remotely on a server.

Docker for Reproducibility

What causes problems in reproducible research?

- Different data (versioning, change control log)
- Different code (version-control with Git/GitHub)
- Different version of analytical software, e.g. Python 2.7 vs. 3.6, different versions of R, and different packages (virtual environments)
- Different versions of underlying dependencies
- Different operating system, different versions of system libraries

Berkeley Initiative for Transparency in the Social Sciences included Docker as part of "Reproducible Research Computational Tools for the Next Generation of Social, Behavioral and Policy Scholars"

<u>LiftR: Persistent Reproducibility</u>

Docker & System Resources

By default, a Docker container has no limits on its memory and CPU usage.

Can limit CPU usage and memory usage with flags set with *docker run*

e.g.

docker run —rm -d -m="4g" —cpus="2.5" pyspark

Docker for Deployment

- Build complex applications as modular micro services rather than monolithic, traditional applications

- Isolation through separable systems.
  - e.g. Analysis using Anaconda 2.7 vs. Python 3.6 Django Web Application
  - Anything with node.js

- Enter **Docker Compose**

Docker Compose

- Define and run multi-container docker applications.
- You create ‘services’ (recall services are docker containers in production).
- Point docker compose yml towards either (1) a directory containing a dockerfile or (2) directly to an image
- Defining a Docker network

Docker Compose

```yaml
version: '3'

services:
  db:
    image: postgres
  web:
    build: .
    command: python3 manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/code
    ports:
      - "8000:8000"
    depends_on:
      - db
```

docker-compose.yml

```yaml
version: '3'

services:
  api:
    build:
      context: ./api-folder
      dockerfile: Dockerfile.api
    expose:
     - "80"
    ports:
     - "80:80"
    links:
     - web-interface:web-interface

  web-interface:
    build: ./web-folder
    volumes:
    depends_on: api
    expose:
     - "80"
    ports:
     - "8000:80"

```

Docker + Spark Deployment Assignment

- docker-compose YAML file
  - pointing to two dockerfiles

- ML Model & API container
  - mount a trained machine learning model pipeline object
  - build a simple API that calls the pipeline using data and returns a prediction/estimate

- Simple web app container
  - Has input cells for entering data for three features
  - Returns predicted value from ML model container.

- ReadMe - clear instructions on how to deploy to EC2