NIH: Data Science with Docker

Containers Workflow Interest Group (NIH/NCI)

June-2024

A word from Docker's CEO

Scott Johnston
CEO

How to participate in today’s session

For online viewers:

- Please place questions into chat. We will either address as we go or discuss at the end, time pending

BUT wait… we only have a limited time today, right?

- This session is the first we intend to hold as well as more collaborative workshops. You will receive a survey which you can select topics you would like to review next session.

Your Docker account team

Dylan Tegarden
Account Executive

Jessica Squizzato
Customer Success

Todd Densmore
Solution Architect

Agenda

1. *Who is Docker?*
2. *What is a Container?*
3. *Modern Data Science*
4. *Data Science in the Cloud*
5. *Data Science Workflows*
6. *Kubeflow Pipelines*
7. *Closing Remarks*

Who is Docker?

Docker

Docker created software “containers”

- **dotCloud Inc.** was founded in 2011
  - a platform as a service (PaaS) for developers
  - run their applications in the cloud
- renamed to **Docker Inc.** in 2013
- Docker containers debuted to the public in Santa Clara at PyCon in 2013

The container specifications

- Docker formed the Open Container Initiative in June 2015
- The purpose is to govern standard image formats and runtimes
- Currently run by the Linux Foundation
  - **image-spec** - build images (BuildKit, podman, buildah)
  - **runtime-spec** - run containers (containerd, CRI-O, Windows Containers)
  - **distribution-spec** - share images (Docker Hub, Artifactory, ECR, GCR, ACR)

At Docker We Believe It Is Critical To Empower ~~Developers~~ Data Scientists

Docker Products and Services

A suite of developer tools that empower innovation

docker hub

Largest marketplace of OSS container images

docker desktop

Fast, secure, hybrid application development from code to run

docker scout

SBOM, Image analysis, policy evaluation, and remediation recommendations

docker build cloud

Fast local and CI builds, multi architecture natively

docker Testcontainers

Test dependencies with real services wrapped in Docker Containers

Security and Management at Scale

What is a Container?

containers

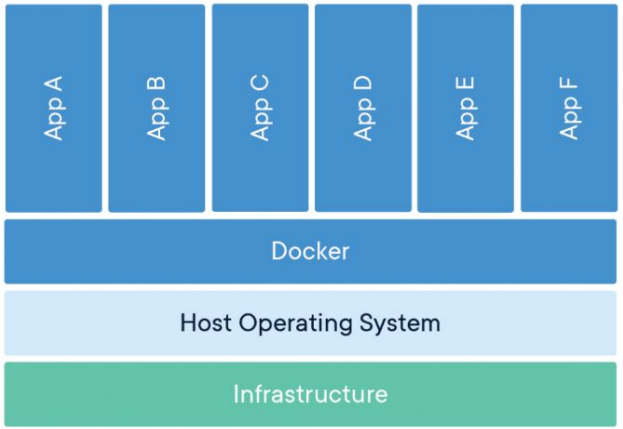
A collection of Linux technologies to package and run **lightweight** application processes **securely**. Software “containers” are similar to shipping containers

Containers run reliably across systems (MacOS, Windows, Linux) and platforms (GCP, AWS, Azure)

Container technology

Features of the **Linux** kernel to allow multiple processes to run in isolation

- **Union filesystem** - single view of multiple file layers
- **Chroot** - change the root directory for a process family (jails)
- **Namespaces** - provides isolation for kernel resources
  - User, PID, network, mount, IPC, UTS, etc.
- **Cgroups** - resource limits for (namespaced) processes
  - CPU, memory, disk I/O, network, etc.



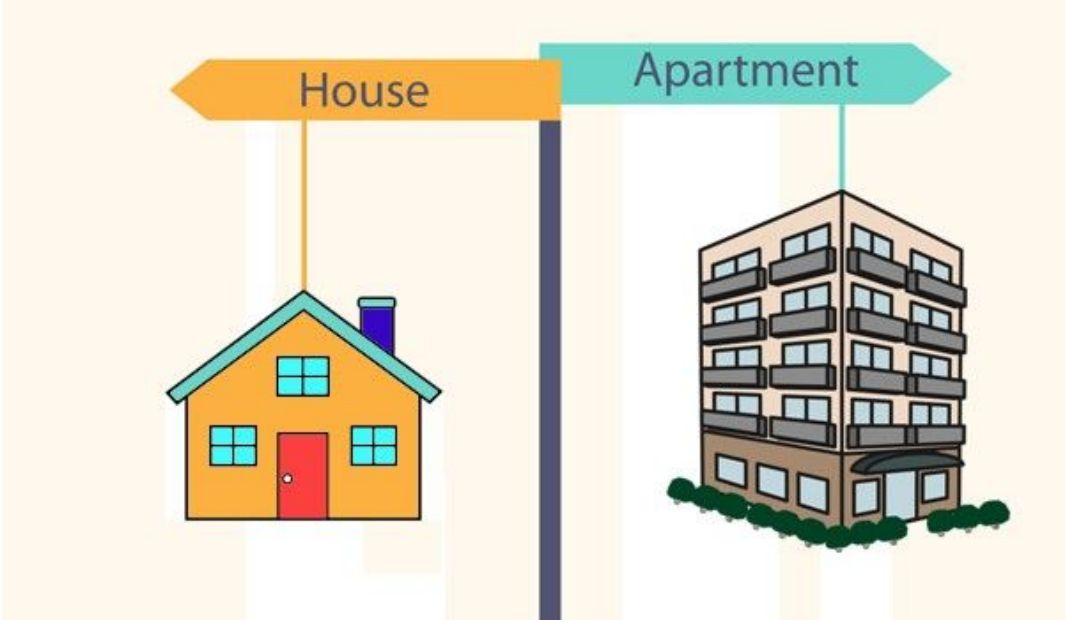
Houses vs Apartments

House has its own

- **Water**
- **Heat**
- **Electricity**

Apartment shares

- **Water**
- **Heat**
- **Electricity**



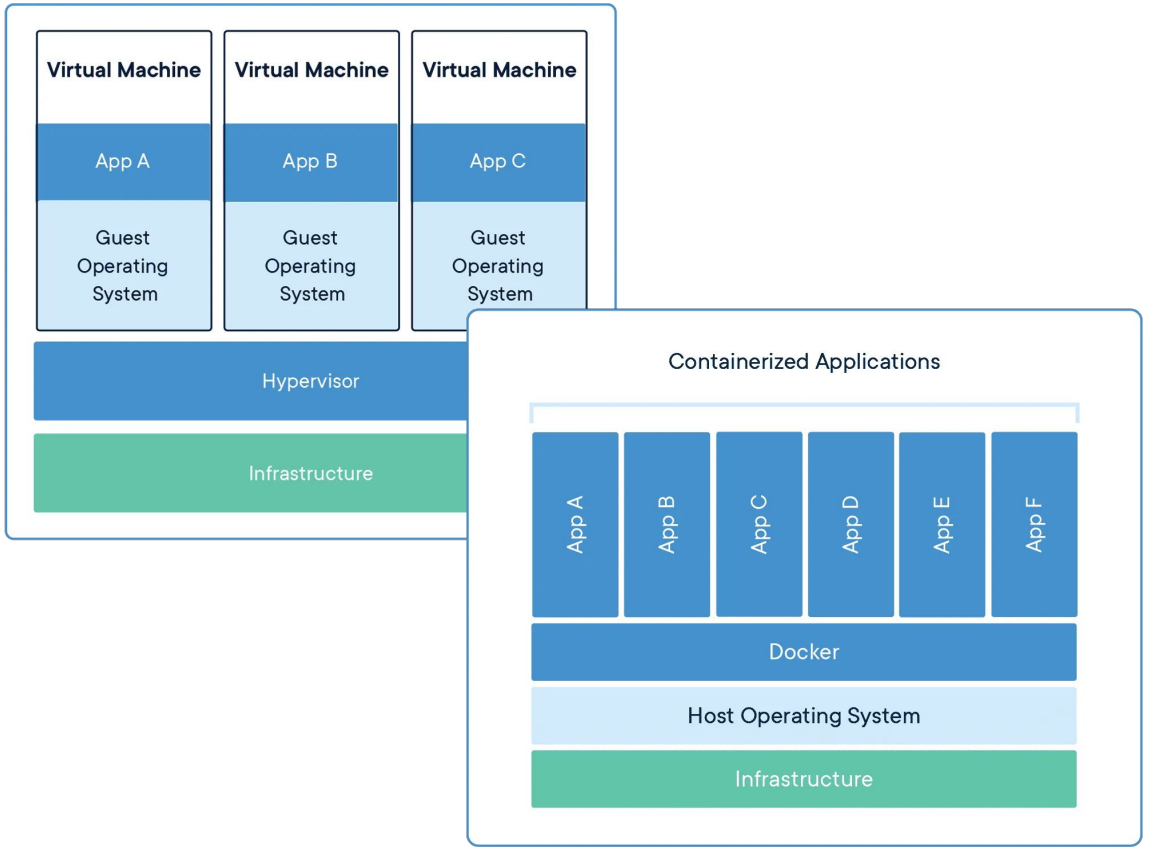
VMs vs Containers

VM virtualizes its own

- **CPU**
- **Memory**
- **Disk**

Container shares

- **CPU**
- **Memory**
- **Disk**



Benefits of Software Containers

- **Efficient**

  Containers require less system resources (they don't include operating system things)
- **Portable**

  Can be run on multiple different operating systems / hardware platforms / clouds
- **Consistent**

  applications in containers will run the same (they include dependencies)
- **Scalable**

  applications can be rapidly deployed, patched, or scaled
- **Application development**

  Containers support accelerate development, test, and production cycles

Result: Containers Power *every* Cloud

Google Cloud

Modern Data Science

Golden Age of Data Science

Everything is connected to the Internet

Massive amounts of data being generated (zettabytes)

- Mobile devices
- Cars
- Medical data

Improvements in cloud computing

- Massive compute power available on-demand

Advances in Artificial intelligence

Some challenges

Problem: Managing Data

Data sets are becoming VERY large (GB-TB)

Providing data sets quickly (on-demand)

- Imaging Data Commons
- Genomic Data Commons

Storing (archiving) data sets

Multiple formats

Data Integrity (Accurate, complete, and valid)

Genomic Data

The original human genome project (Celera) is based on 5 anonymous donors, however this is being expanded.

The current reference human genome is 3.1 billion base pairs

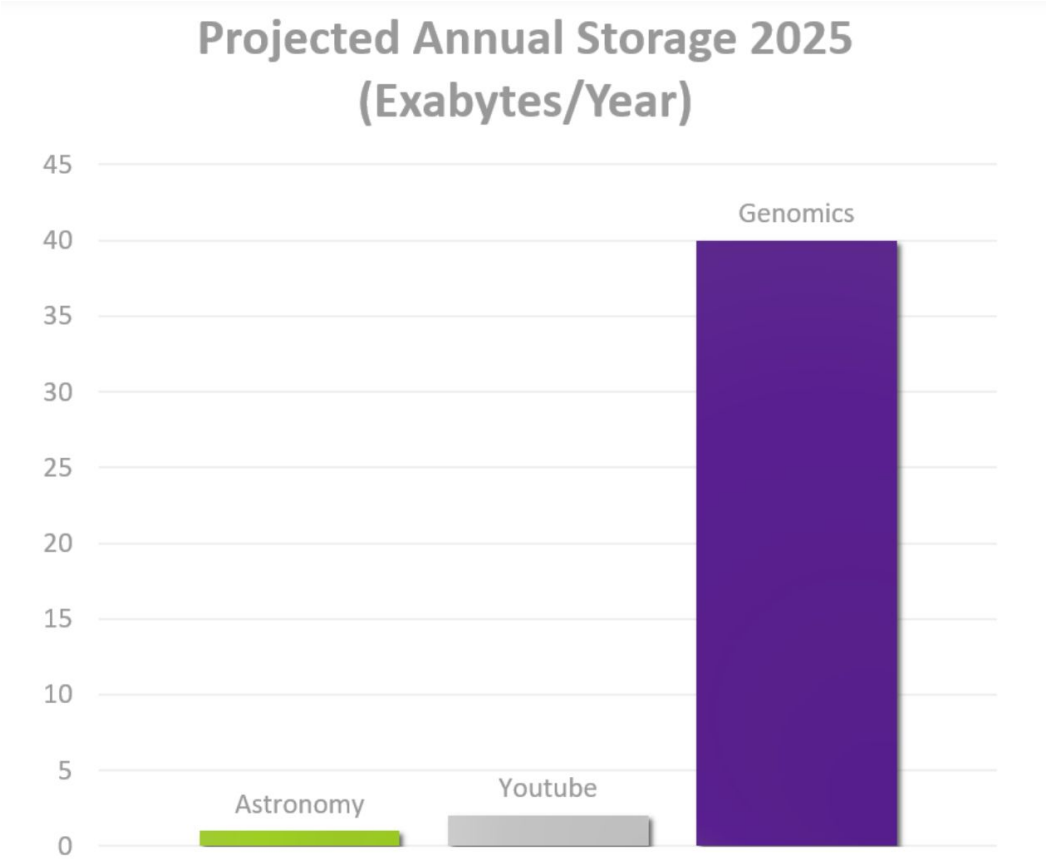
GRCh38.p14 (July 2023) is 3.1 Gb (2 Gb compressed)

But… the volume of genomics data is doubling every 7 months

By 2025, the predicted total genomics data capacity is 40 exabytes

(40,000 petabytes)

Genomic Data vs. YouTube



Data Science in the Cloud

Benefits: Cloud Platforms for Data Science (1)

Compute scalability

- Enables massive bursts of computation
- close to the data - easily attach to terabytes of data

Accessibility

- Share data easily (14 million medical images with annotations)
- Share data globally

On-demand specialized Tools

- Machine learning tools

Benefits: Cloud Platforms for Data Science (2)

Cost savings

- Pay by usage (hour / query / terabyte)

Reliability

- Availability - guaranteed uptime (99.999% == 5 mins 15 seconds)
- Fault tolerance / redundancy (data and services)

Security

- data security and isolation (regions are very important)

Example: Machine Learning

Specialized Hardware: GPUs

Example: a Google `a3-megagpu-8g` machine has:

- 8 - NVIDIA A100 GPU (208 virtual CPUs)
- 1,872 GB RAM
- 6.4 TB storage

Specialized Frameworks:

- Google Cloud AI Platform
- Tensorflow

NIH Cloud Platforms: STRIDES

STRIDES - Science and Technology Research Infrastructure for Discovery, Experimentation and Sustainability

https://cloud.nih.gov/resources/cloudlab/

- Amazon Web Services
- Google Cloud
- Microsoft Azure

Data Science Workflows

What is Kubernetes (K8S)?

The standard way to define and manage containerized applications

- Open Source
- Works across all cloud providers (or on-prem)
- Originally created by **Google** (Borg)
- Greek word for “helmsman” or “(ship) pilot”
- Handles networking, secrets, scaling, data storage, failover, deployments, etc

What is Kubeflow?

A simple way to define and run ML workflows

Runs on Kubernetes (based on **containers**)

Standardizes ML Ops

Provides best-in-class tools for data scientists

- Notebooks (JupyterLab, RStudio, and Visual Studio Code)
- Data pipelines (ML workflows)
- Model training using PyTorch, TensorFlow, XGBoost
  - or BYOM (HuggingFace, DeepSpeed, or Megatron-LM)
- Model serving

Kubeflow Pipelines

What is a Workflow?

A formal series of (software) steps

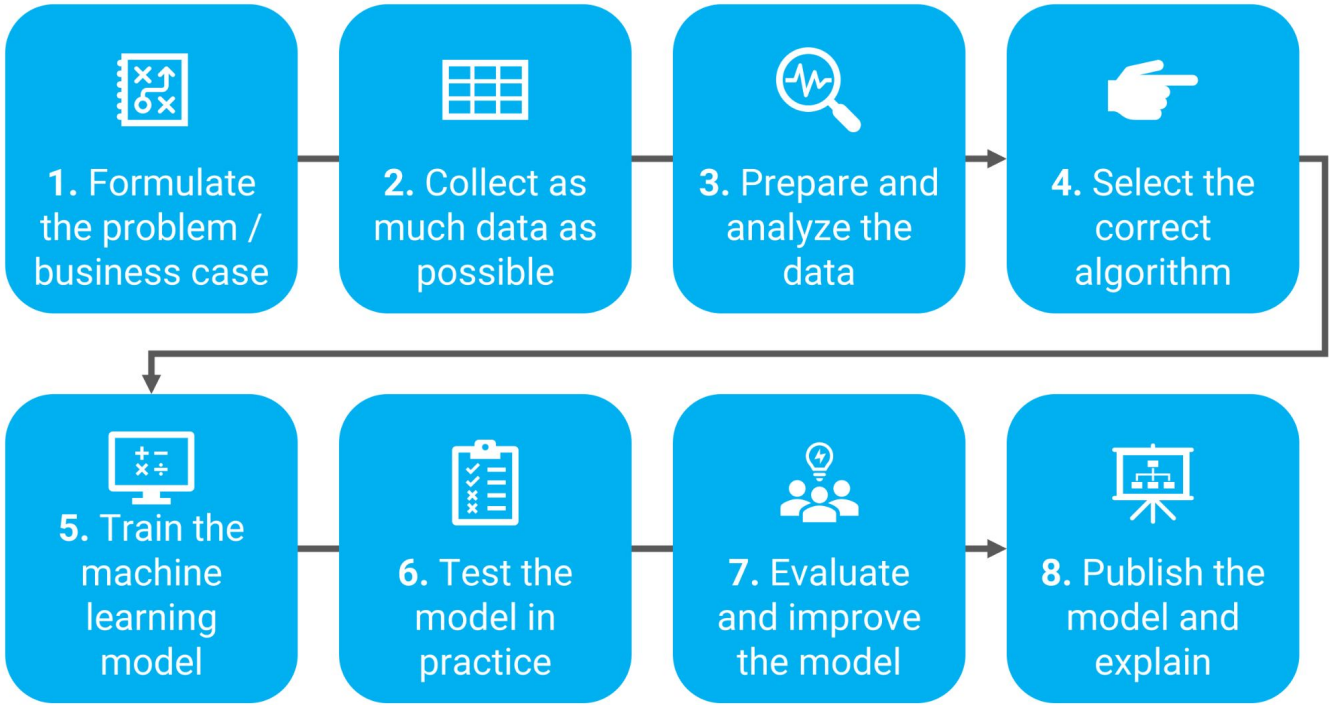
Each steps performs a specific task:

- Input massive amounts of data
- Process (transform) and analyze the data
- Outputs (including metadata and data provenance)

Guidelines

FAIR principles - findability, accessibility, interoperability, and reuse (of assets) for humans and computers

Example: Data Science Workflow



Example Kubeflow pipeline

Kubeflow Pipeline Code: Python

```python
import kfp
from kfp import dsl

def load_data_op():
    return dsl.ContainerOp(
        name="Load Data",
        image="python:3.7",
        command=["sh", "-c"],
        arguments=["echo 'Loading data' && sleep 5"],
    )
def preprocess_data_op():
    return dsl.ContainerOp(
        name="Preprocess Data",
        image="python:3.7",
        command=["sh", "-c"],
        arguments=["echo 'Preprocessing data' && sleep 5"],
    )
def train_model_op():
    return dsl.ContainerOp(
        name="Train Model",
        image="python:3.7",
        command=["sh", "-c"],
        arguments=["echo 'Training model' && sleep 5"],
    )
@dsl.pipeline(
    name="My First Pipeline",
    description="A simple pipeline that demonstrates loading, preprocessing, and training steps."
)
def my_first_pipeline():
    load_data = load_data_op()
    preprocess_data = preprocess_data_op().after(load_data)
    train_model = train_model_op().after(preprocess_data)
if __name__ == "__main__":
    kfp.compiler.Compiler().compile(my_first_pipeline, "my_first_pipeline.yaml")
```

Write a pipeline in familiar Python

Each python function is a step in the pipeline

Can have as many steps as needed to load, transform and analyze data

Kubeflow Pipeline Code: Python

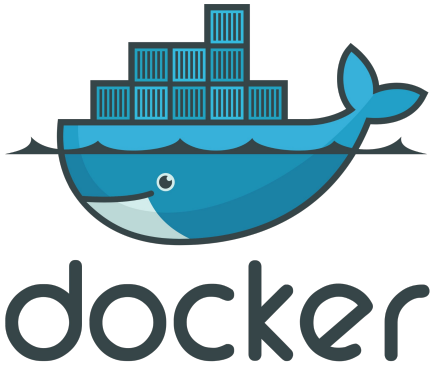
```python
import kfp
from kfp import dsl

def load_data_op():
    return dsl.ContainerOp(
        name="Load Data",
        image="python:3.7",
        command=["sh", "-c"],
        arguments=["echo 'Loading data' && sleep 5"],
    )
def preprocess_data_op():
    return dsl.ContainerOp(
        name="Preprocess Data",
        image="python:3.7",
        command=["sh", "-c"],
        arguments=["echo 'Preprocessing data' && sleep 5"],
    )
def train_model_op():
    return dsl.ContainerOp(
        name="Train Model",
        image="python:3.7",
        command=["sh", "-c"],
        arguments=["echo 'Training model' && sleep 5"]
    )
@dsl.pipeline(
    name="My First Pipeline",
    description="A simple pipeline that demonstrates loading, preprocessing, and training steps."
)
def my_first_pipeline():
    load_data = load_data_op()
    preprocess_data = preprocess_data_op().after(load_data)
    train_model = train_model_op().after(preprocess_data)
if __name__ == "__main__":
    kfp.compiler.Compiler().compile(my_first_pipeline, "my_first_pipeline.yaml")
```

The KFP library converts each python function into a versioned container

The whole pipeline is transformed into protobuf YAML

Can import directly into Kubeflow or build pipelines in a Jupyter Notebook.

Docker benefits for Workflows



Portable

- Facilitates research collaboration
- Regulatory questions answered / data retention

Reproducible

- Generate the **same result** from the same data, software (packages, versions), compute environment
- Workflow results can be independently verified

Scalable

- Containers consume less resources than VMs
- Quickly scaled up or down to accommodate different data sets

Secure

- Containers are run in isolation

Closing Remarks

Docker Business Subscription

A suite of developer tools that empower innovation

docker desktop

docker hub

docker scout

docker build cloud

SSO, SCIM, and account hierarchy

Centralized configuration and audit logging

Hardened Rootless Docker Desktop

Policy-based registry and image access

Contact dylan.tegarden@docker.com for pricing

Thank You!