
Distributed Computing



DSBA 6190-U90 | Colby T. Ford, Ph.D.

Overview

Terms

History of Parallel Computing

Containerization and Kubernetes

Intro to Apache Spark

Terms

High Performance Computing

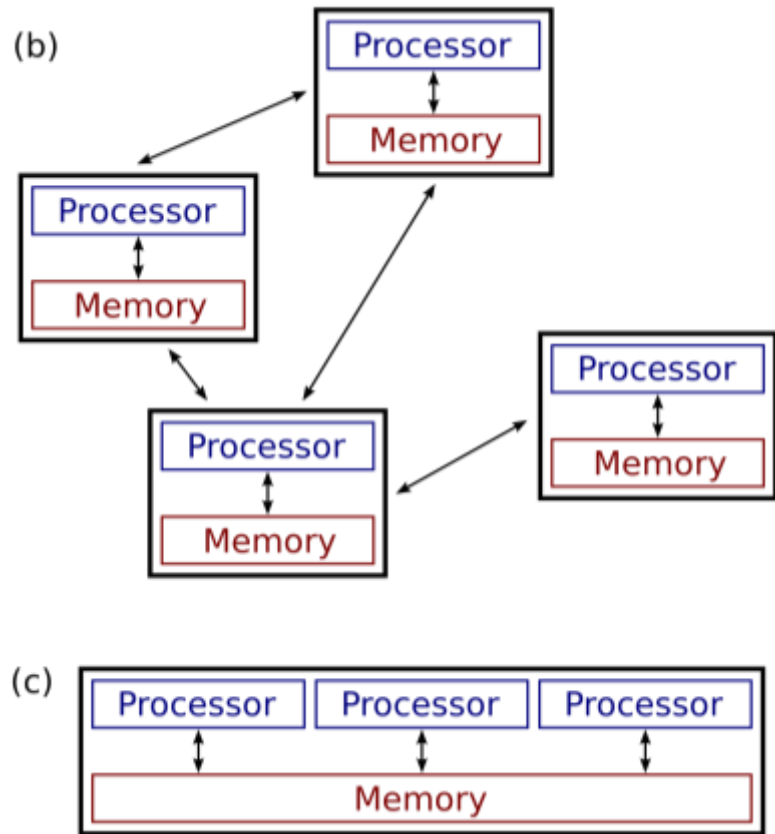
- Using more powerful machines to do computations faster.
- Measured in FLOPS

Parallel Computing (c)

- Using multiple cores to do multiple things at once.
- Measured in degree of parallelism

Distributed Computing (b)

- Using multiple machines to do multiple things at once.
- Measured in degree of distribution and parallelism



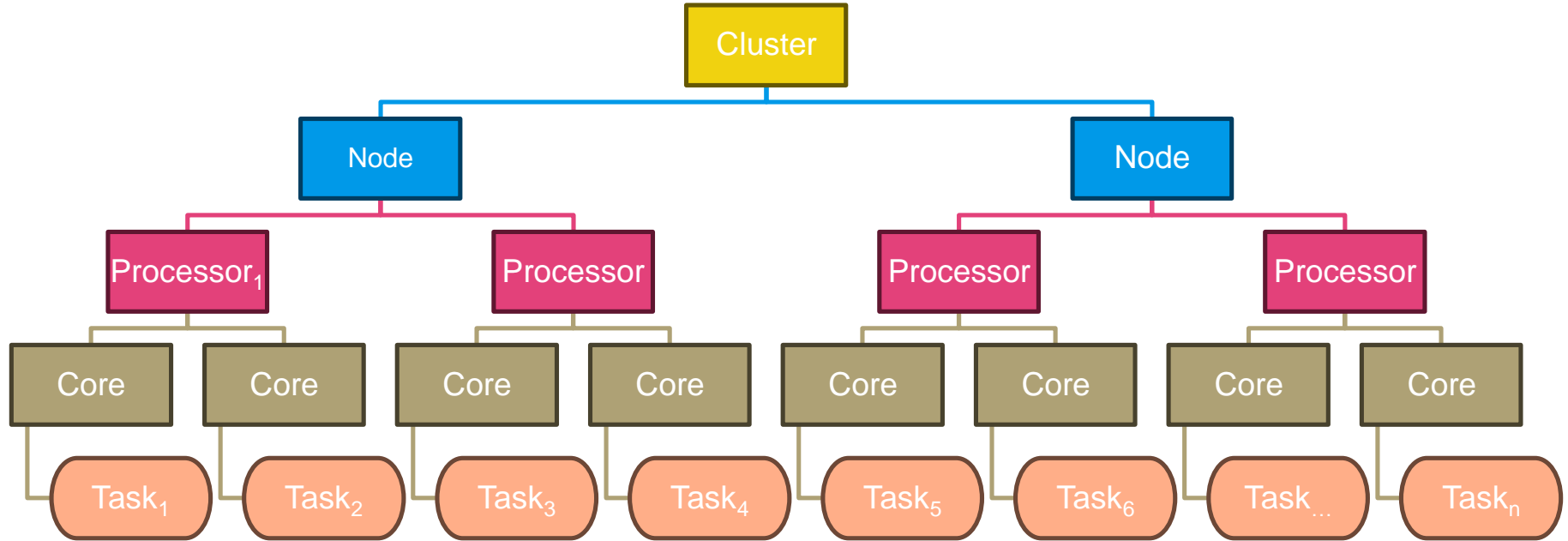
Technology Pipeline

Multicore
MPI
SNOW



kubernetes

Cluster Architecture



Containerization

Popular Container Frameworks



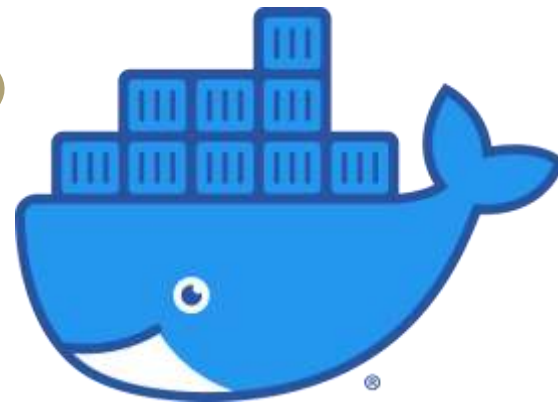
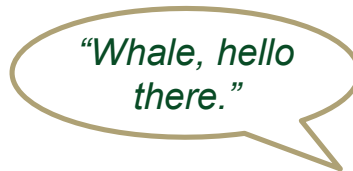
Most popular, Full-featured
Requires root
Windows, macOS, and Linux



Less popular, More limited
Daemonless and rootless
Windows, macOS, and Linux

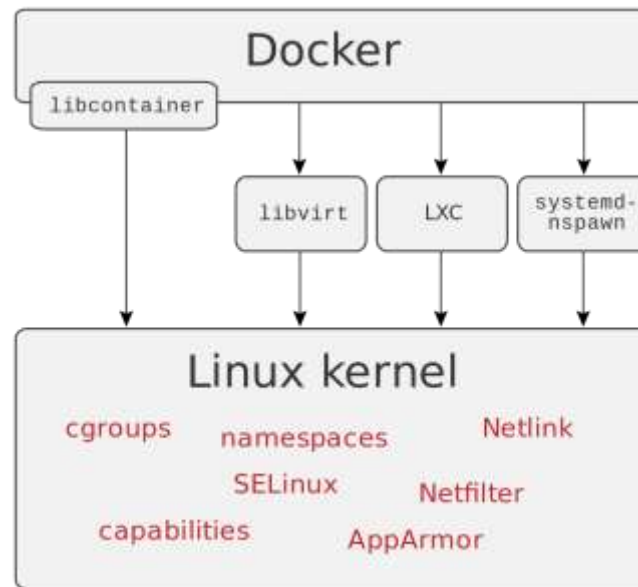


Popular on HPC Clusters
Only for Linux
(formerly "Singularity")



Docker

- Support for Linux and Windows Server containers (and now WebAssembly).
- Flexibility to support microservices and traditional app workloads.
- Integrated graphical user interface-based management and operation.
- Granular role-based access control, LDAP, and Azure Active Directory integration.
- Connection to custom networking and volumes (data storage)
- Think of Docker is a trimmed down VM image with sets of packages and settings that are configured for reuse.



Dockerfile

- Contains layers of instructions for configuring the system and installing libraries and files.
- Specifies a base layer, which is useful for “picking up where someone left off”

38 lines (13 sloc) | 546 Bytes

```
1 FROM rocker/r-ver:4.1.2
2
3 LABEL org.opencontainers.image.licenses="GPL-2.0-or-later" \
4      org.opencontainers.image.source="https://github.com/rocker-org/rocker-versioned2" \
5      org.opencontainers.image.vendor="Rocker Project" \
6      org.opencontainers.image.authors="Carl Boettiger <cboettig@ropensci.org>"
7
8 ENV S6_VERSION=v2.1.0.2
9 ENV RSTUDIO_VERSION=2021.09.1+382
10 ENV DEFAULT_USER=rstudio
11 ENV PATH=/usr/lib/rstudio-server/bin:$PATH
12
13 RUN /rocker_scripts/install_rstudio.sh
14 RUN /rocker_scripts/install_pandoc.sh
15
16 EXPOSE 8787
17
18 CMD ["init"]
```



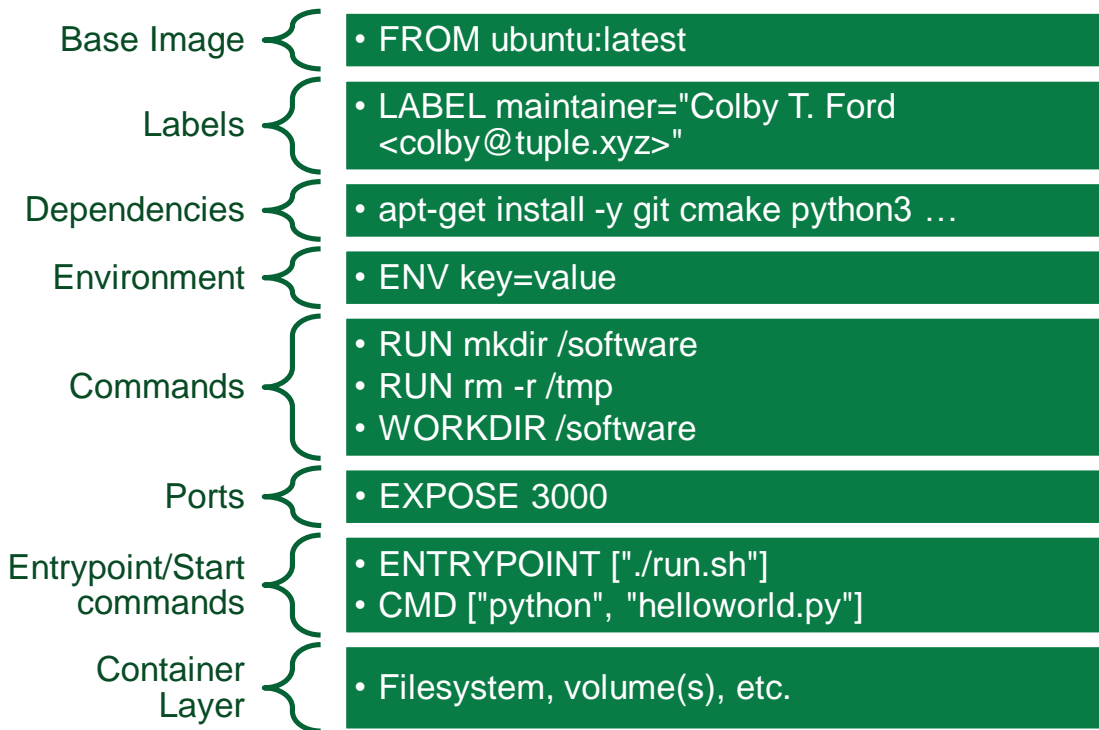
40 lines (34 sloc) | 1.00 KB

```
1 FROM nvidia/cuda:11.3.1-base-ubuntu20.04
2
3 # Install some basic utilities
4 RUN apt-get update && apt-get install -y \
5     curl \
6     ca-certificates \
7     sudo \
8     git \
9     bzip2 \
10    libx11-6 \
11    && rm -rf /var/lib/apt/lists/*
12
13 # Create a working directory
14 RUN mkdir /app
15 WORKDIR /app
16
17 # Create a non-root user and switch to it
18 RUN adduser --disabled-password --gecos '' --shell /bin/bash user \
19    && chown -R user:user /app
20 RUN echo "user ALL=(ALL) NOPASSWD:ALL" > /etc/sudoers.d/90-user-
21 USER user
22
23 # All users can use /home/user as their home directory
24 ENV HOME=/home/user
25 RUN chmod 777 /home/user
26
27 # Set up the Conda environment
28 ENV CONDA_AUTO_UPDATE_CONDA=false \
29     PATH=/home/user/miniconda/bin:$PATH
30 COPY environment.yml /app/environment.yml
31 RUN curl -sLO ~/miniconda.sh https://repo.continuum.io/miniconda/Miniconda3-py39_4.10.3-Linux-x86_64.sh \
32    && chmod +x ~/miniconda.sh \
33    && ~/miniconda.sh -b -p ~/miniconda \
34    && rm ~/miniconda.sh \
35    && conda env update -n base -f /app/environment.yml \
36    && rm /app/environment.yml \
37    && conda clean -ya
38
39 # Set the default command to python3
40 CMD ["python3"]
```



Dockerfile Layers

- Docker uses layers to save on image space and on build complexity.
- Each layer contains the information that changes the layer before it.
- When you rebuild, Docker will cache the unchanged layers.



Docker Commands

[Get started](#) | [Docker Docs](#)

Pull an Image from a Container Registry

- `docker pull <url>/<user>/<image>:<tag>`
- `docker pull nvidia/cuda:12.6.1-cudnn-devel-ubuntu22.04`
- `docker pull nvcr.io/nvidia/pytorch:22.04-py3`

Build an Image Locally

- `docker build -t <image tag> .`
- `docker build -t reactwebapp .`

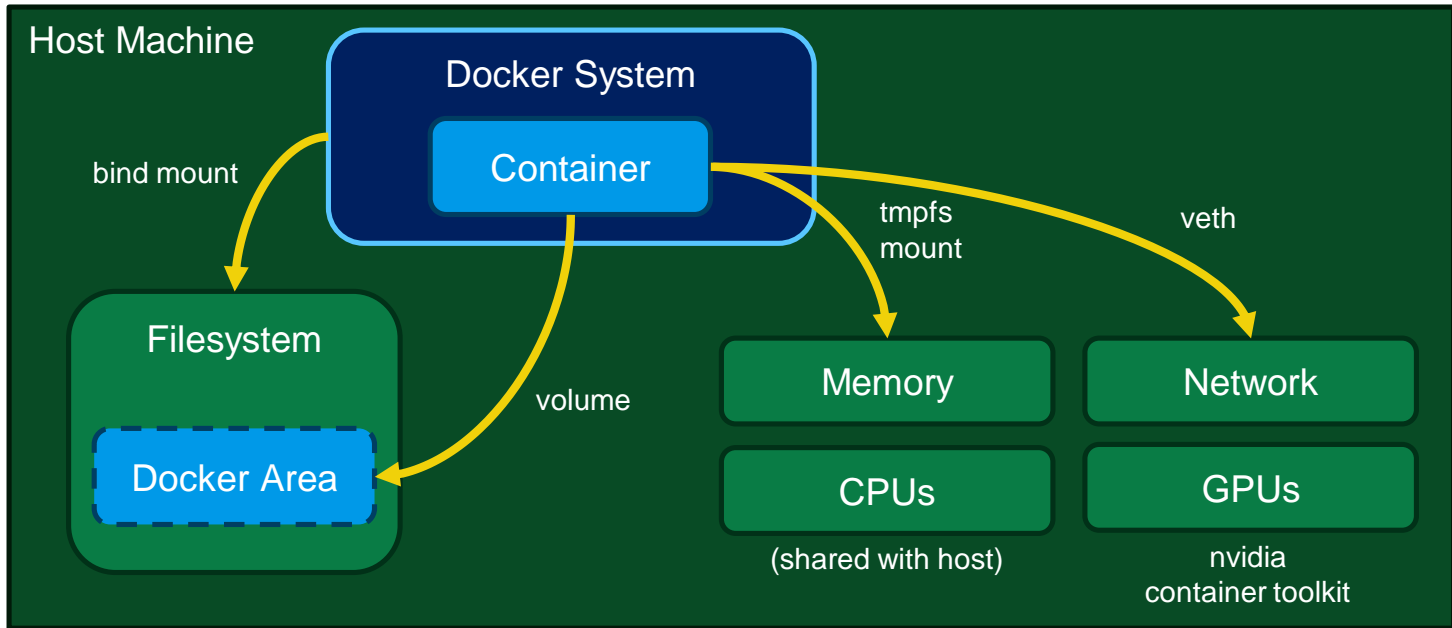
Run a Container Locally

- `docker run <image name> <other options>`
- `docker run -p 8888:8888 -v /data/local:/home/jovyan/work jupyter/base-notebook`
- `docker run --name cuda_gpu_1 --gpus all -t nvidia/cuda`

Execute Command in a Container

- `docker run/exec <image name> <commands>`
- `docker run --name mycontainer1 -it mycontainer /bin/bash`
- `docker exec -it mycontainer /bin/bash`

Volumes (and sharing other resources)



Docker Compose

- Docker Compose allows you to specify multiple images that can communicate with one another.
 - Networking (vNets)
 - Volumes (databases)

- `docker compose up -d`
- `docker compose down`

Less used in cloud native environments where PaaS options are more common.

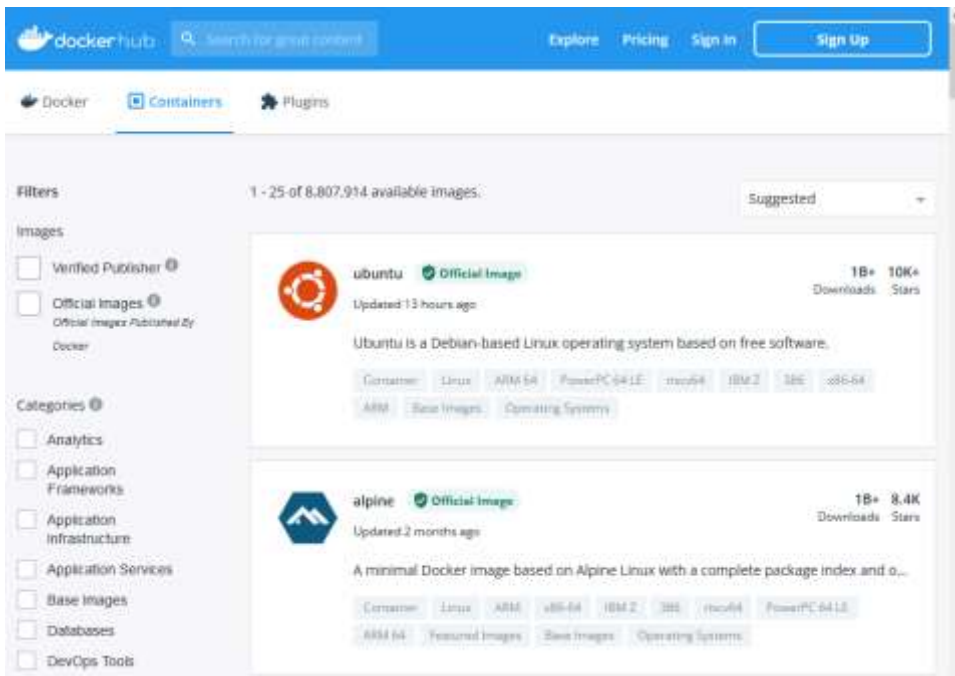
```
.
├── backend
│   └── Dockerfile
│   ...
├── db
│   └── password.txt
├── compose.yaml
├── frontend
│   └── ...
│       └── Dockerfile
└── README.md
```

Directory Structure

compose.yaml File

```
services:
  backend:
    build: backend
    ports:
      - 80:80
      - 9229:9229
      - 9230:9230
    ...
  db:
    image: mariadb:10.6.4-focal
    ...
  frontend:
    build: frontend
    ports:
      - 3000:3000
    ...
```

Container Storage

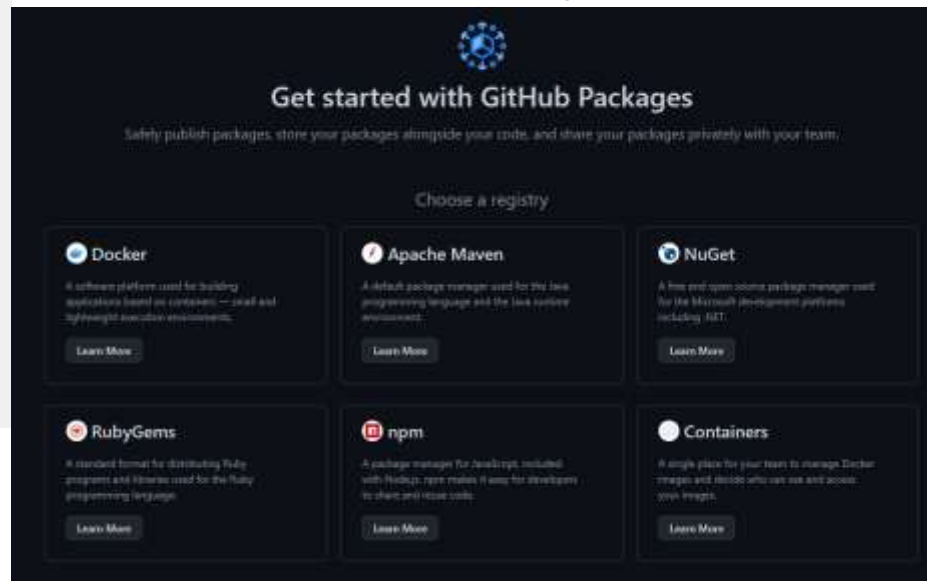


DockerHub



Azure Container Registry

GitHub Packages



Container Usage

- Deploy containerized applications to Azure for quick and scalable use.
- Pick the Runtime Stack (Node, .NET, Python, etc.) and OS (Linux or Windows)
- Pick the size of machine (RAM, CPUs, GPUs)



Azure Web Apps



Azure Container Instances



Azure Kubernetes Service

Container Registry Commands

Login to the Container Registry

- `az acr login --name <registry name>`
- `az acr login --name crdsba6190deveastus001`

Tag the Image with the URL, your name, and tag

- `docker tag <image name> <registry name>.azurecr.io/<image name>:<tag>`
- `docker tag instructor_sklearn crdsba6190deveastus001.azurecr.io/instructor_sklearn:latest`

Push the Image to the Container Registry

- `docker push <registry name>.azurecr.io/<image name>:<tag>`
- `docker push crdsba6190deveastus001.azurecr.io/instructor_sklearn:latest`

Container Orchestration

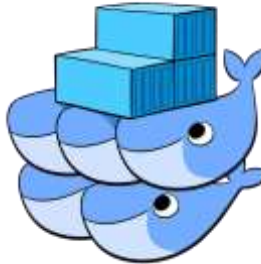
- Once containers are created, use container orchestration to organize, coordinate, and schedule their use.
- Useful for scaling containers and making use of distributed environments
- Other capabilities:
 - Security management
 - API Serving
 - Resource Monitoring
 - Load Balancing



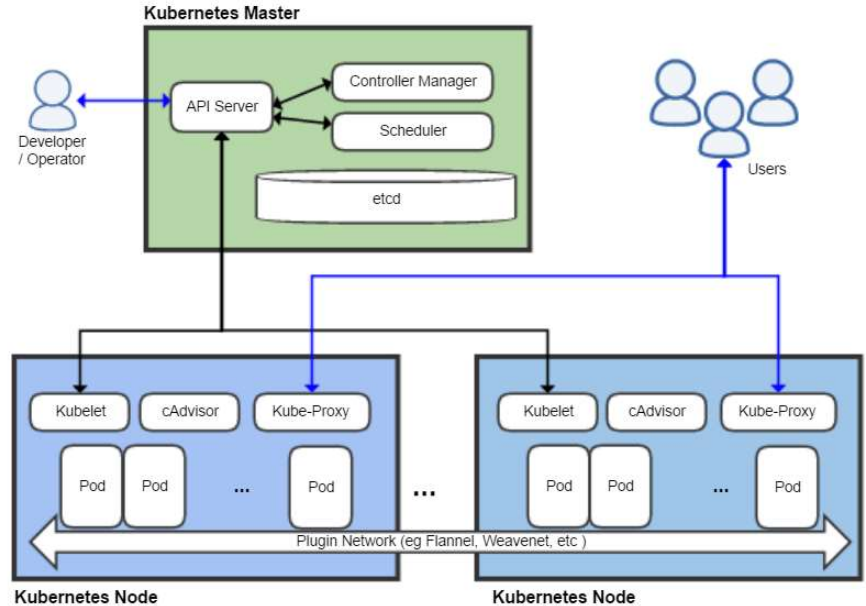
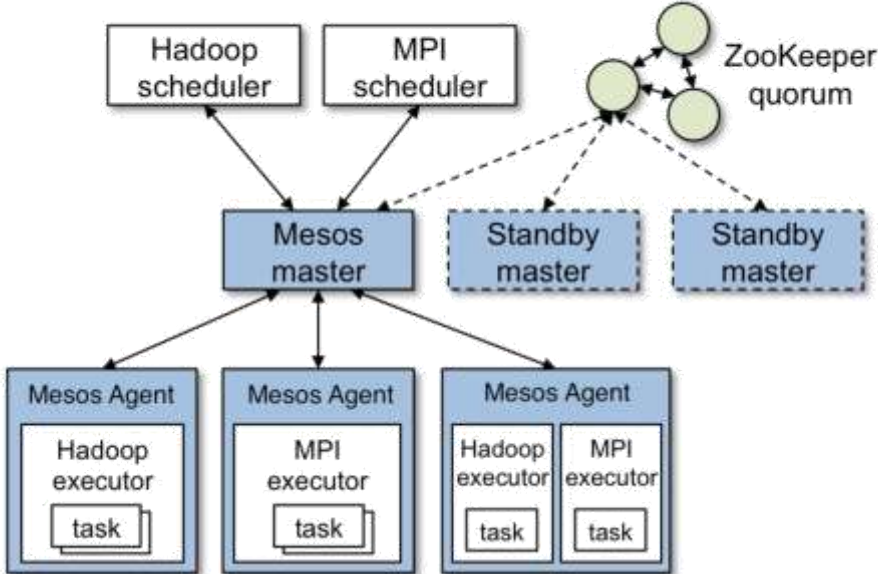
kubernetes



MESOS



Mesos Architecture



What is / Why Kubernetes?

What is Kubernetes?



Kubernetes (K8s) is an open-source container orchestration system.

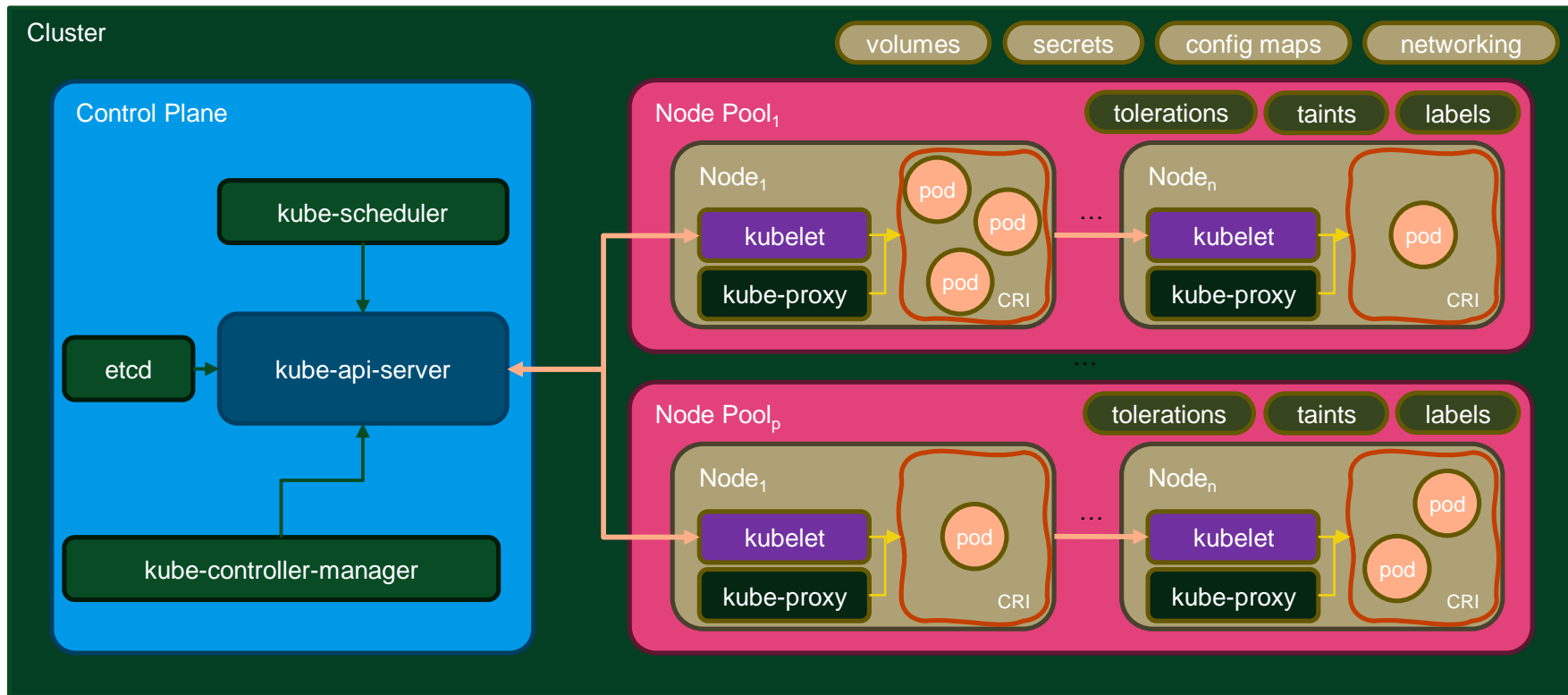
Use Docker containers and spin up compute nodes as needed.

Specify different resource needs by node pool.
Memory, CPUs, GPUs

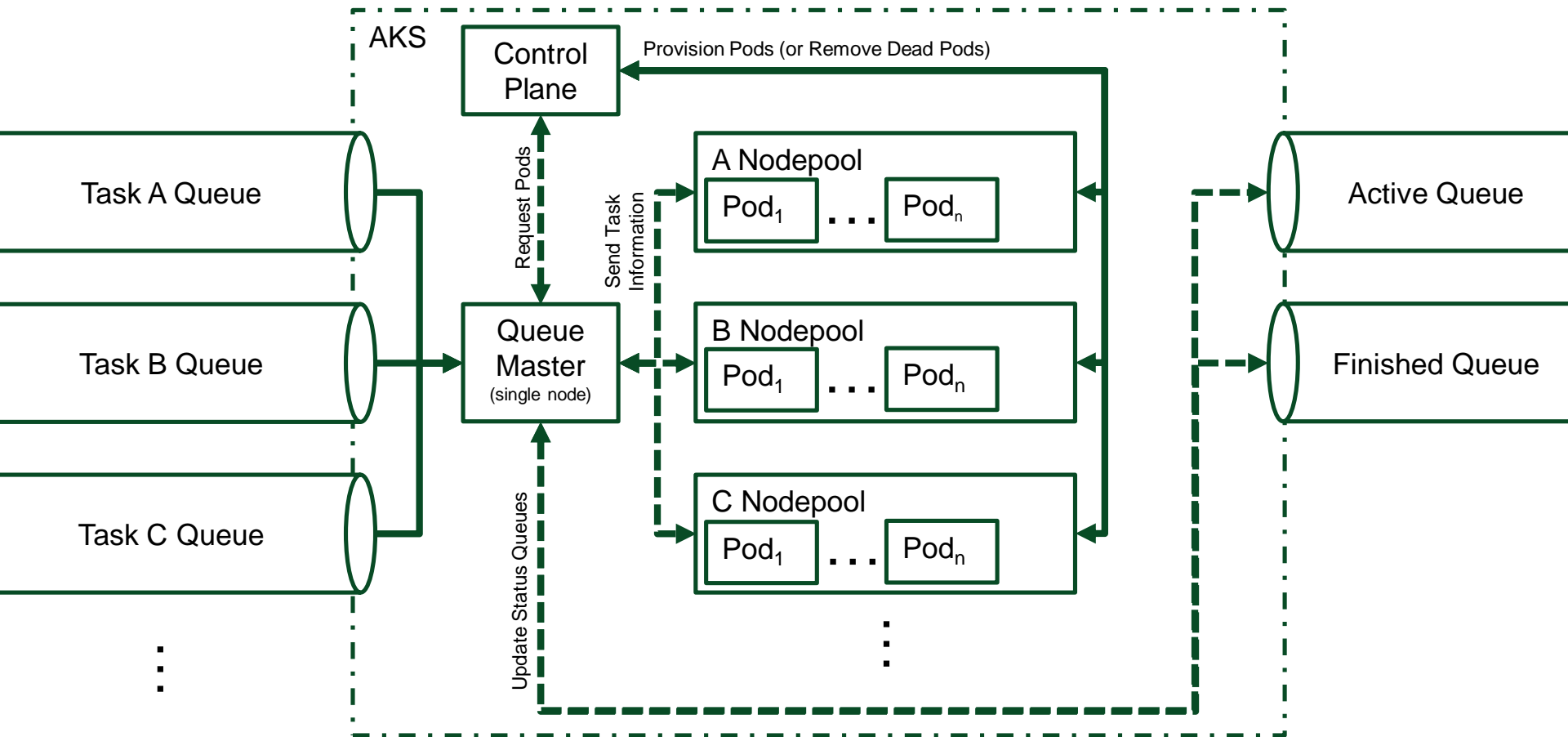
Why (Cloud) Kubernetes?

- K8s can be used for everything from microservices to long-running compute tasks
 - Not everything in bioinformatics is a pipeline. (Duct taped shell scripts, anyone?)
 - Not every pipeline has the same needs (GPUs, memory, CPUs, etc.)
- Cloud providers have platform services that are fully compatible with open-source K8s
 - Azure Kubernetes Service (AKS)
 - AWS Elastic Kubernetes Service (EKS)
 - Google Kubernetes Engine (GKE)
- Take advantage of the “limitless” compute capacity of the cloud + networking and security
- Mount your data lake as a Volume in the K8s cluster, making the data available to every container
 - No copying data around
 - Get references and input files from `/mnt`
 - Write results directly back to the data lake

Basic K8s Cluster Architecture



Kubernetes + Queue-Based Architecture



Example Pod Specification

- Specifies the name of the pod, its image, and other information.
 - Volumes
 - Commands
 - Secrets
 - Resource Requests
 - CPUs
 - GPUs
 - Memory
 - Tolerations

```
apiVersion: v1
kind: Pod
metadata:
  name: instructor-test-01
spec:
  restartPolicy: Never
  containers:
    - name: instructor-sklearn
      image: crdsba6190deveastus001.azurecr.io/instructor_sklearn:latest
      volumeMounts:
        - name: datalake
          mountPath: "/mnt/datalake/"
          readOnly: false
      # command: ["/bin/bash", "-c"]
      # args: [ "./run.py" ]
      command: [ "/bin/bash", "-c", "--" ]
      args: [ "while true; do sleep 30; done;" ]
      # resources:
      #   limits:
      #     memory: "2Gi"
      #     cpu: "200m"
      imagePullSecrets:
        - name: acr-secret
      volumes:
        - name: datalake
          persistentVolumeClaim:
            claimName: pvc-datalake-class-blob
```


kubectl Commands

Get Azure Kubernetes Service Credentials

- `az aks get-credentials --resource-group rg-ahab-dev-eastus-001 --name kub-ahab-dev-eastus-001 --overwrite-existing`

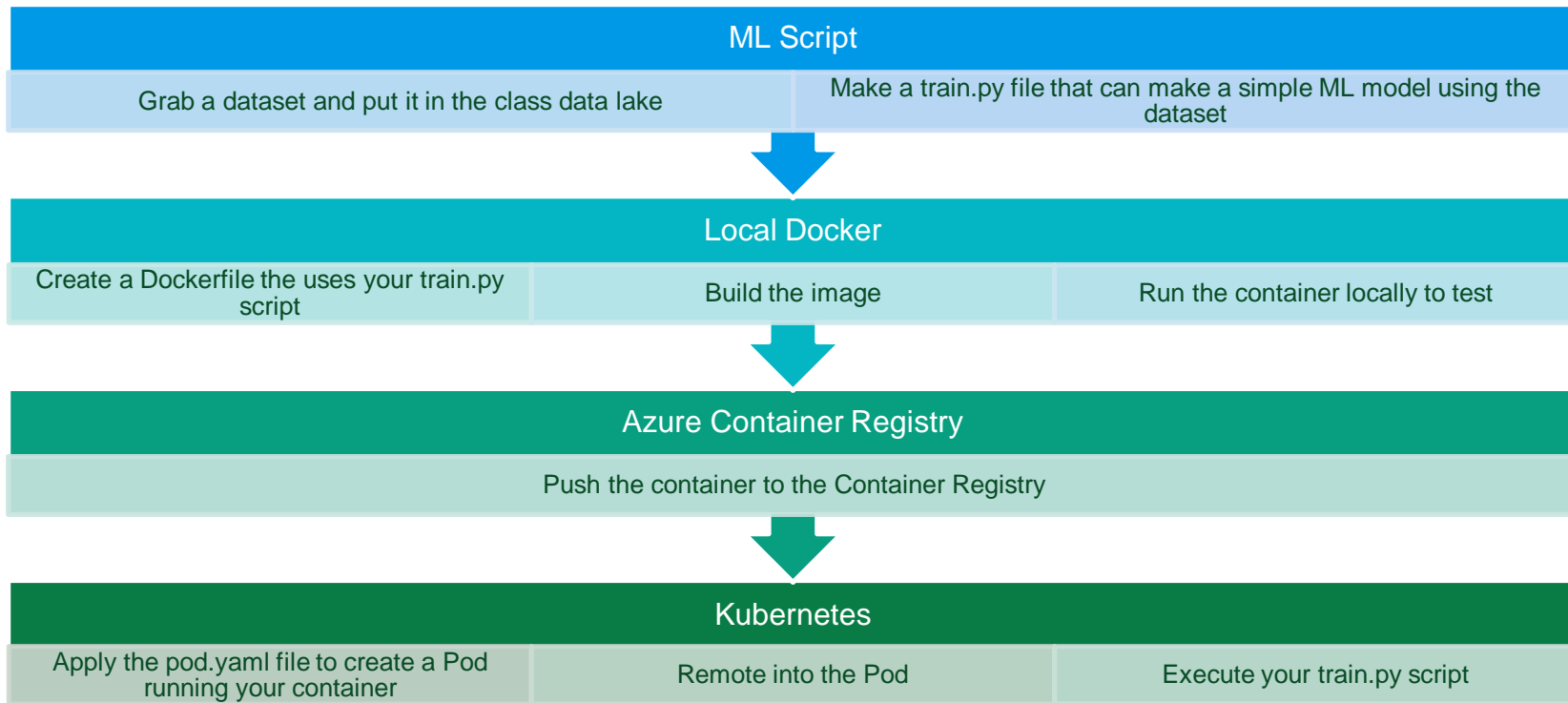
Apply Kubernetes Pod

- `kubectl apply -f <pod yaml file>`
- `kubectl apply -f example_pod.yml`

Execute Command in Pod

- `kubectl exec -it <pod_name> -- <command>`
- `kubectl exec -it instructor-test-01 -- /bin/bash train.py`

Lab Steps - Kubernetes



Apache Spark + Databricks



Why Spark?

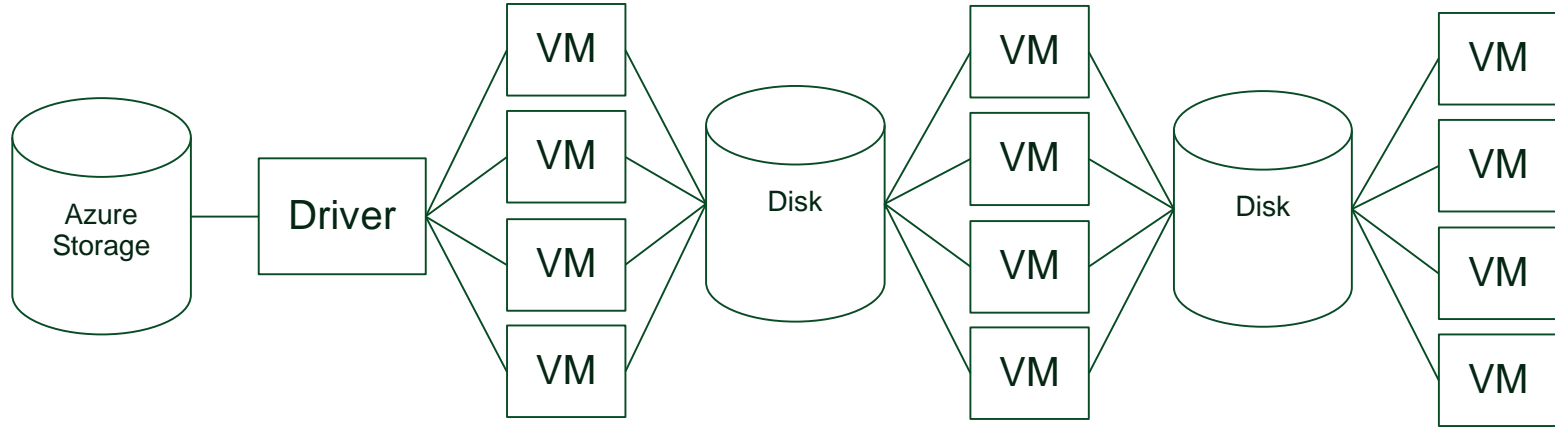
- Open-source data processing engine built around **speed, ease of use, and sophisticated analytics**
- In memory engine that is up to **100 times faster than Hadoop**
- **Largest open-source data project** with 1000+ contributors
- **Highly extensible** with support for Scala, Java and Python alongside Spark SQL, GraphX, Streaming and Machine Learning Library (MLlib)

Why Databricks?

- Databricks is the premium version of Spark available in the market
- Spark founders created Databricks
- Spark is the dominant workload in Hadoop
- **Databricks commits 75% of the code to Open-Source Spark**

Hadoop MapReduce

MapReduce in Hadoop



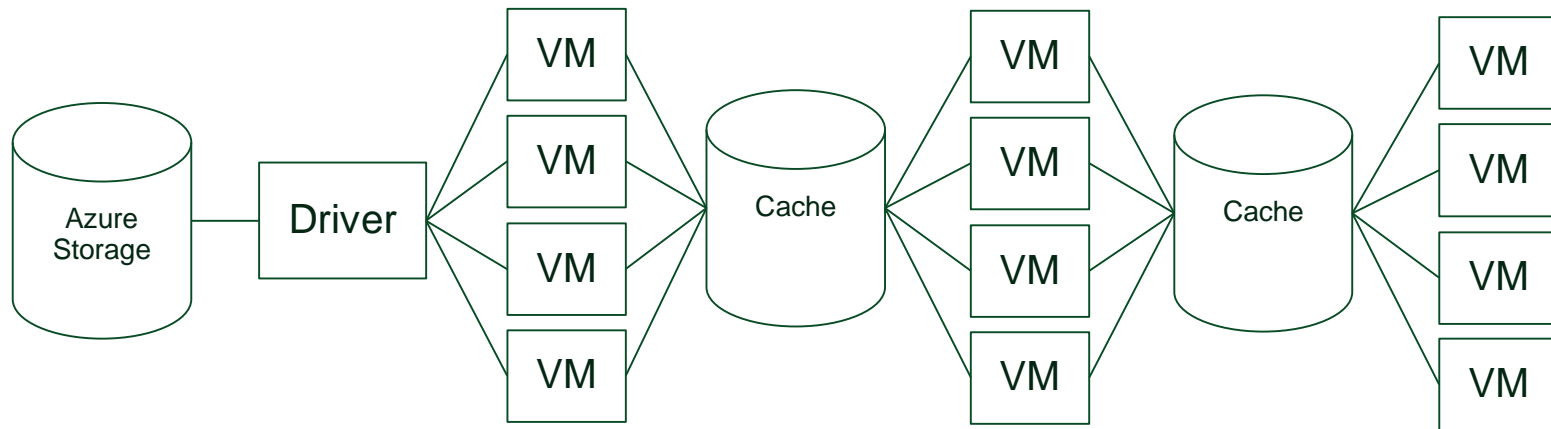
Azure Storage > Driver > VM/Parallelization > write to Disk > VM/Parallelization > write to disk > repeat...



Writing to disk takes time... every time you run this process in MapReduce

What is Azure Databricks?

Apache® Spark™ is FASTER and EASIER than MapReduce in Hadoop



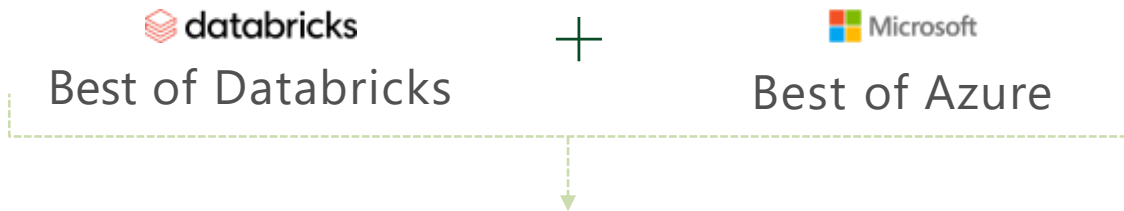
Faster – In Spark data stays in cache this give Spark the speed over MapReduce (writing to disk)



Easier – You can use the language you are most comfortable with in Spark (Python, Scala, R, SQL)

What is Azure Databricks?

A fast, easy and collaborative Apache® Spark™ based analytics platform optimized for Azure



Designed in collaboration with the founders of Apache Spark



One-click set up; streamlined workflows



Interactive workspace that enables collaboration between data scientists, data engineers, and business analysts.

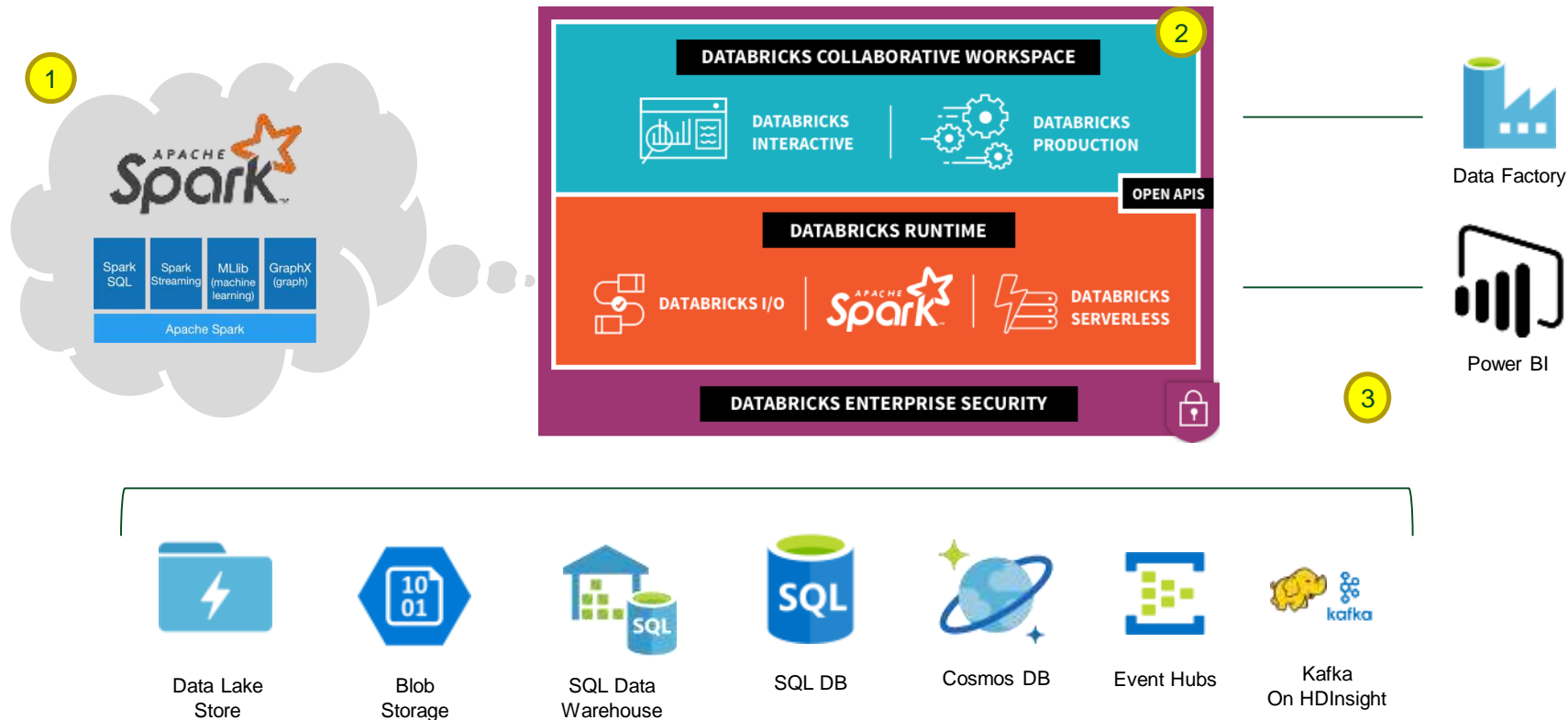


Native integration with Azure services (Power BI, SQL DW, Cosmos DB, Blob Storage)

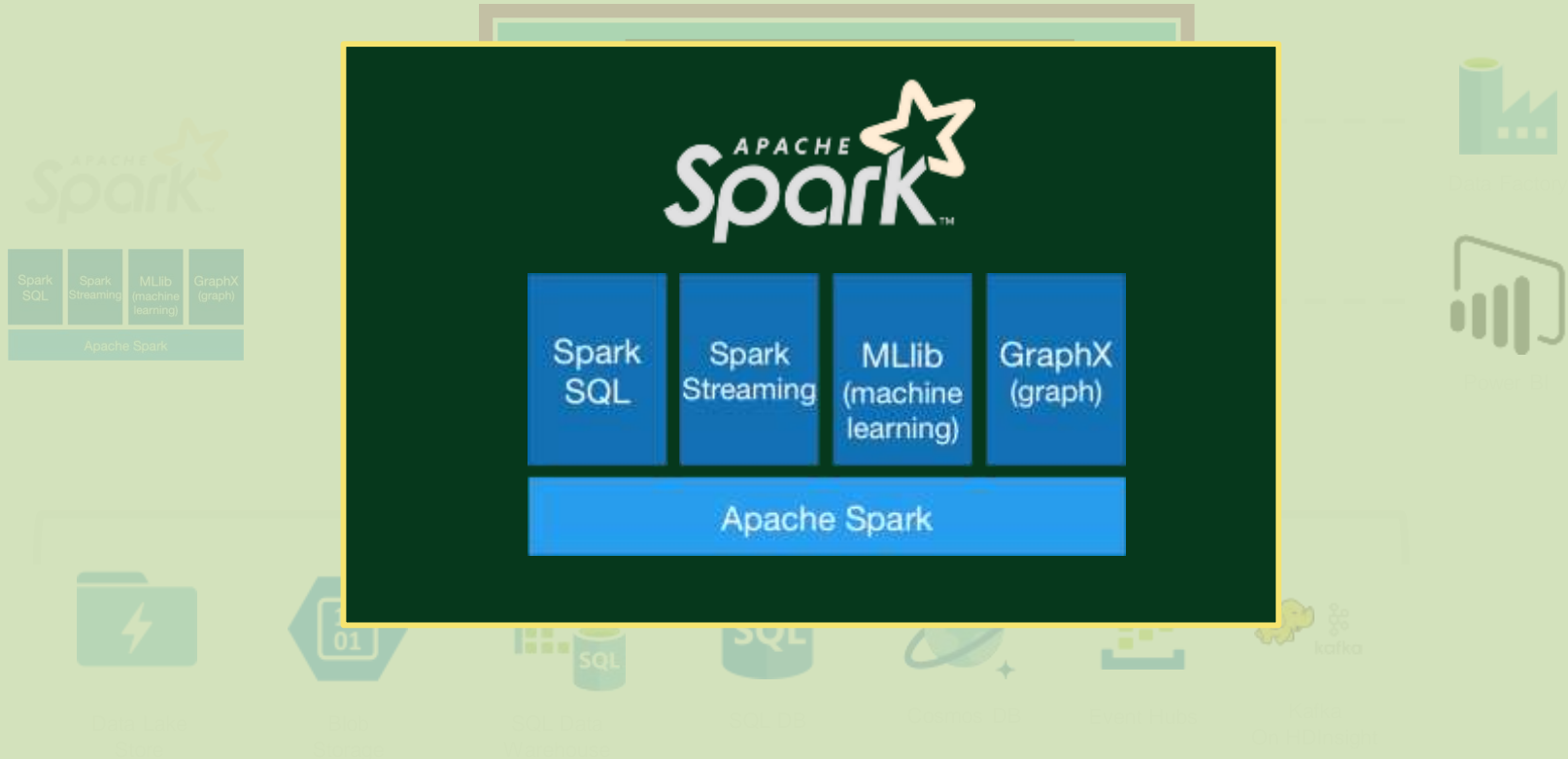


Enterprise-grade Azure security (Active Directory integration, compliance, enterprise-grade SLAs)

What's Under the Hood?



What's Under the Hood?



Azure Databricks key audiences & benefits



Data scientist

Integrated workspace

Easy data exploration

Collaborative experience

Interactive dashboards

Faster insights

- Best Spark & serverless
- Databricks managed Spark



Data engineer

Improved ETL performance

- Zero management clusters, serverless

Easy to schedule jobs

Automated workflows

Enhanced monitoring & troubleshooting

- Automated alerts & easy access to logs

Zero Management Spark

Cluster democratization (High-concurrency)



CDO, VP of analytics

Fast, collaborative analytics platform accelerating time to market

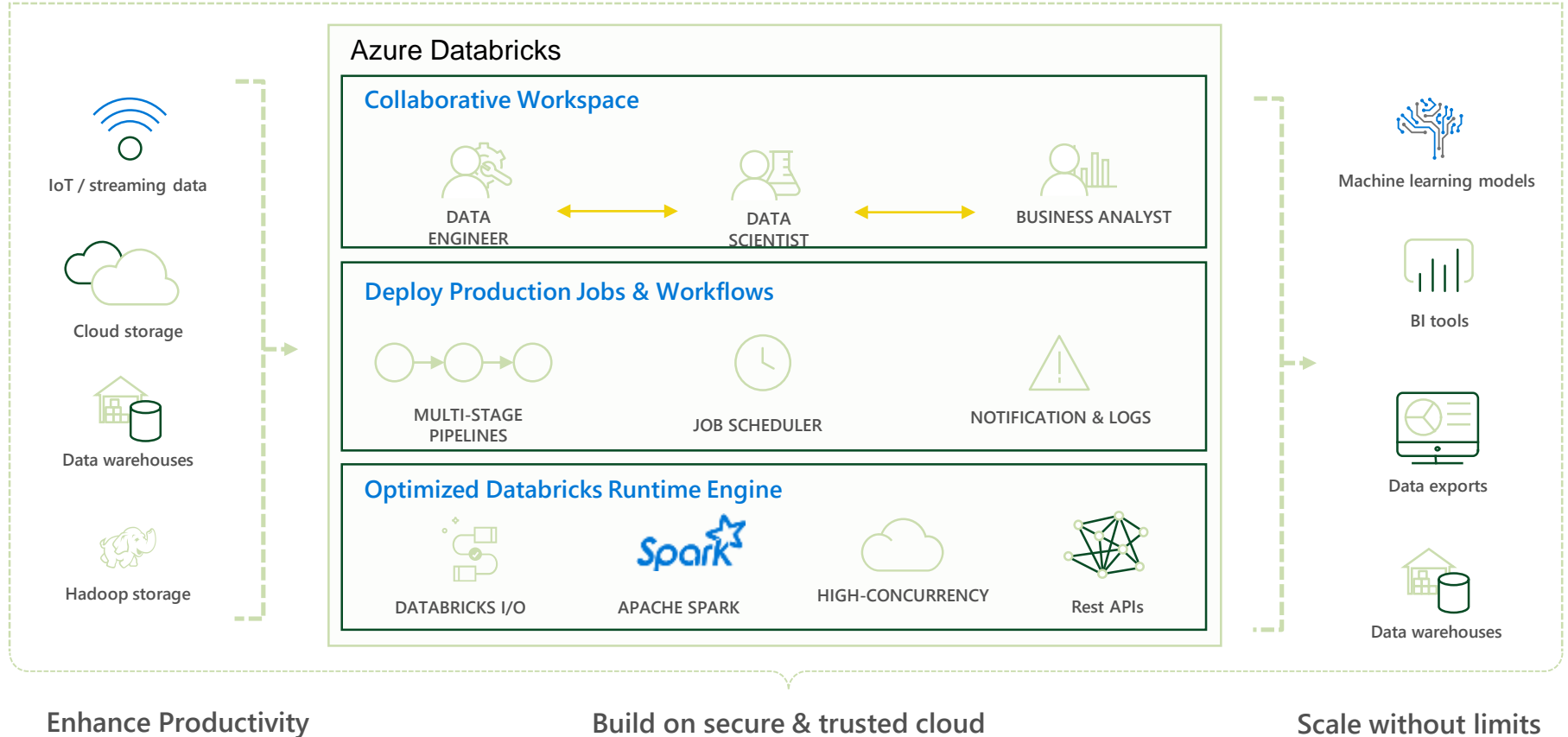
No dev-ops required

Enterprise grade security

- Encryption
- End-to-end auditing
- Role-based control
- Compliance

← Unified analytics platform →

Azure Databricks



Spark is Lazy, but in a Smart Way

When you execute a command, Spark doesn't actually do anything unless it's required to show you something or it has to write something out.

This is the difference between a **Transformation** and an **Action**.

“Lazy Evaluation”

Transformations

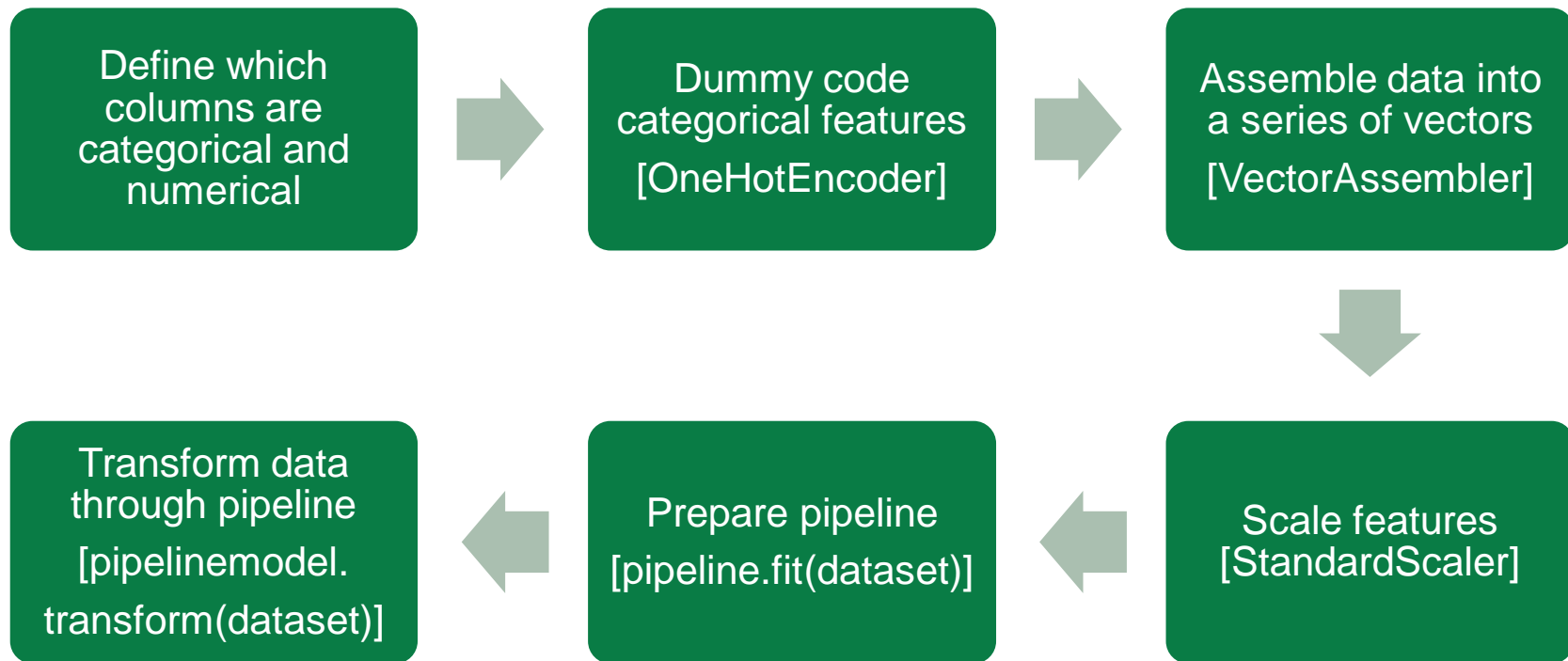
- Lazily evaluated.
- Spark operation that returns a DataFrame (usually).
- Multiple transformations add to the steps of a Spark job, but no data gets processed.
- Examples: select, filter, groupBy, map

Actions

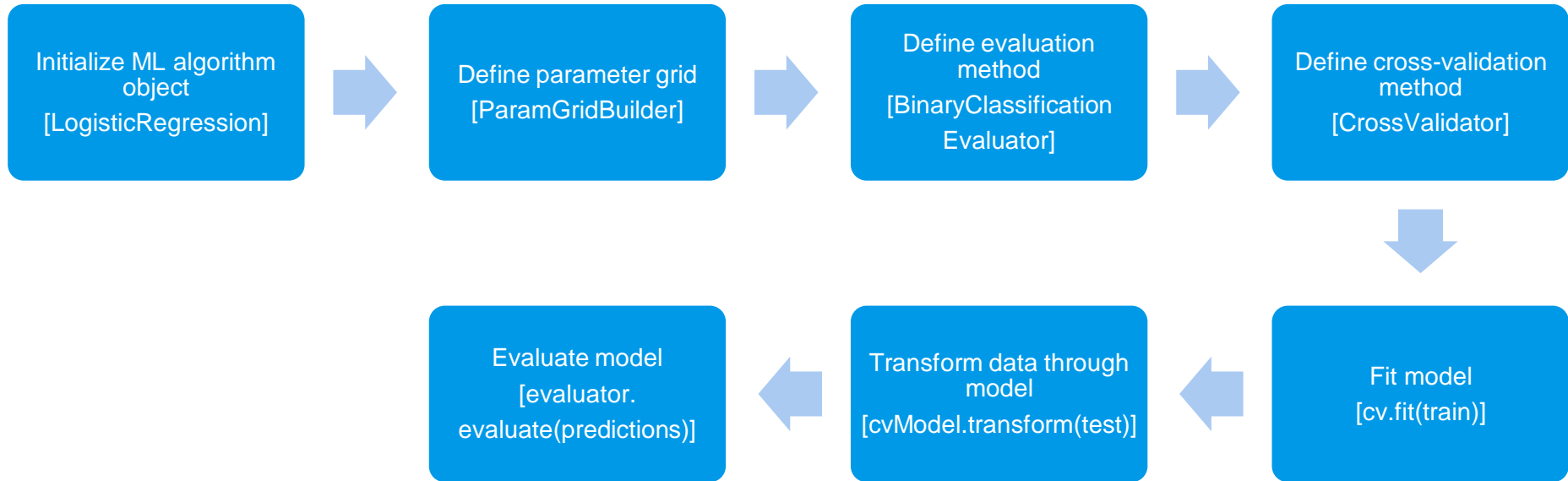
- Not lazily evaluated.
- Returns counts of elements or lists of them (usually)
- Enacts a series of transformations
- Examples: show/display, count, collect, take, write

Spark Pipelines

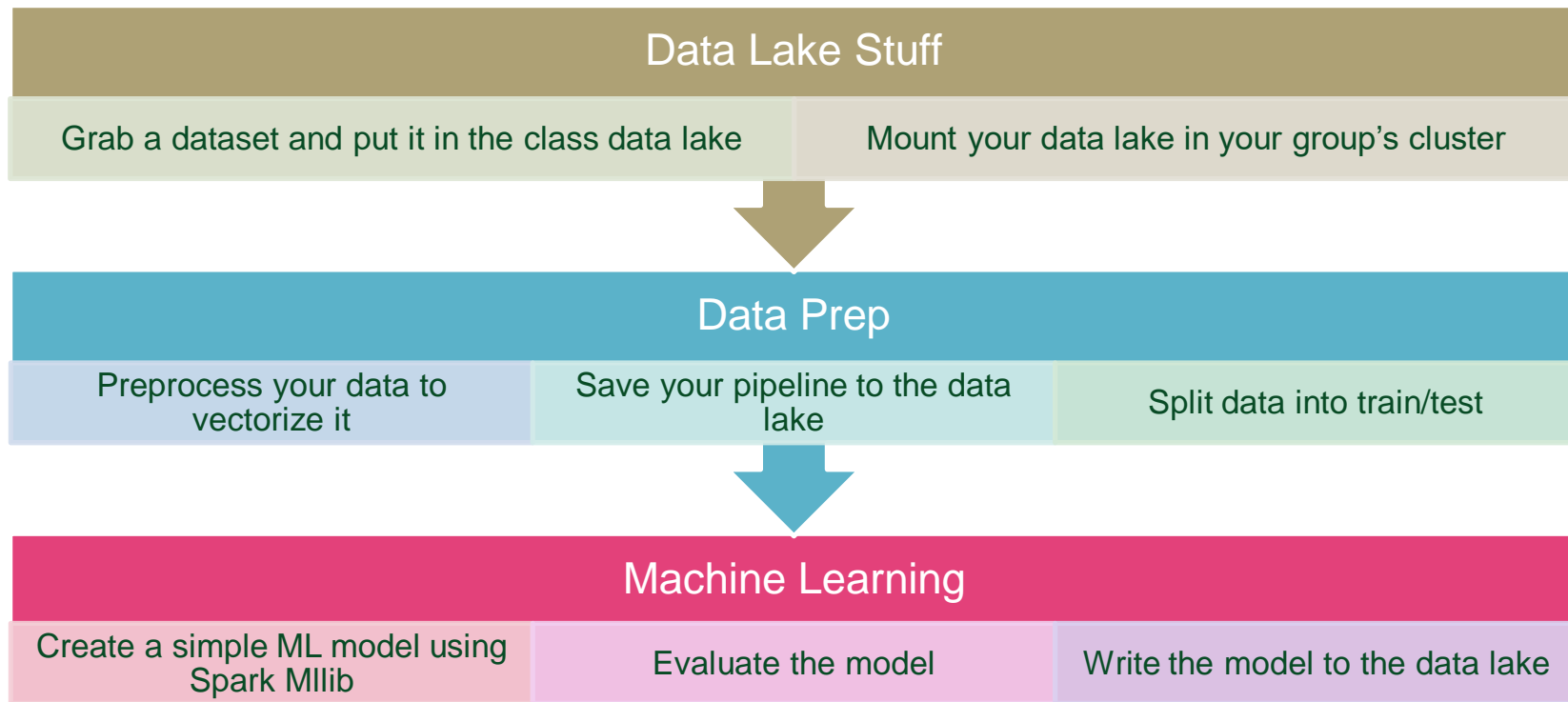
- Can be saved similar to ML models
- Provides stability into the model scoring process



MLlib Models

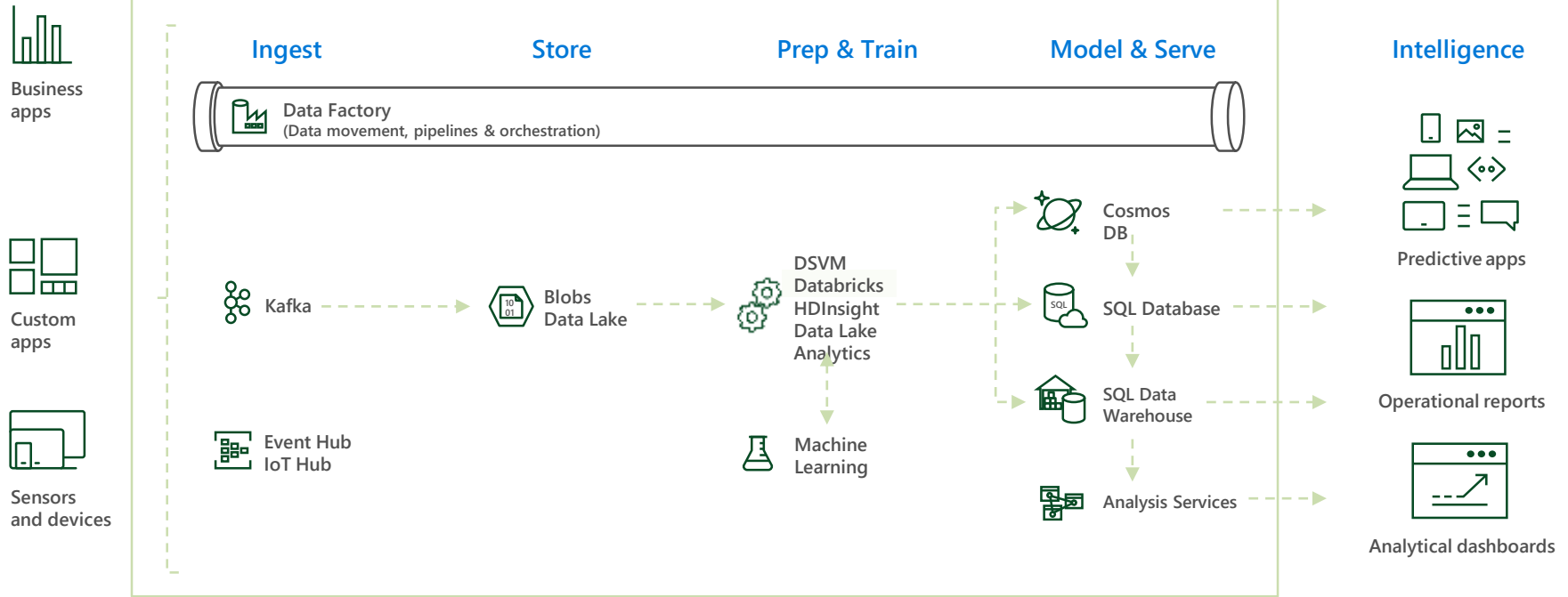


Lab Steps - Databricks

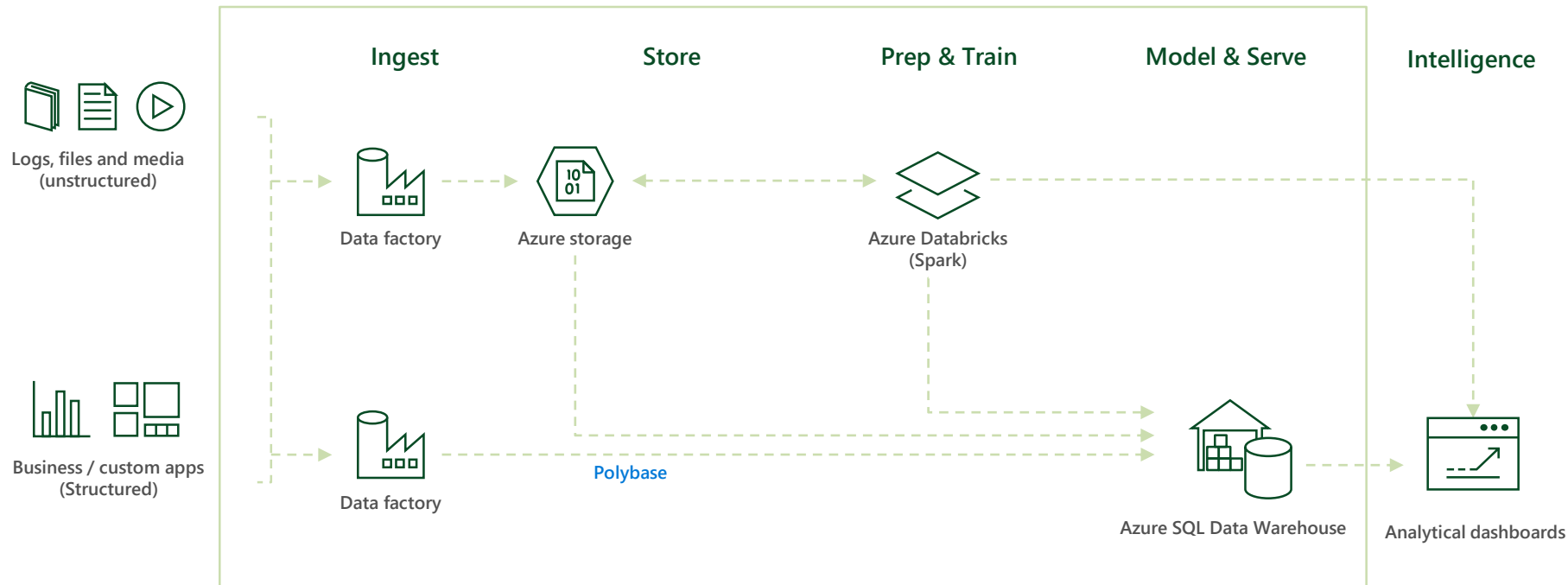


Sample Architectures

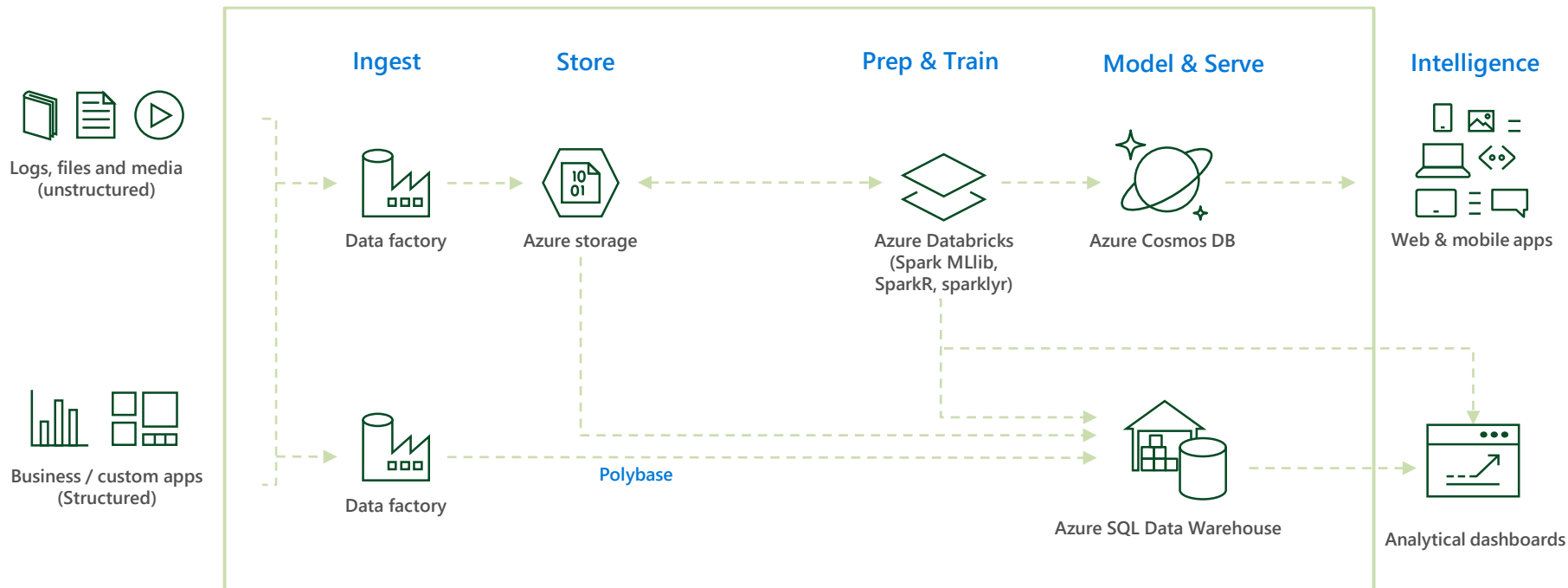
BIG DATA & ADVANCED ANALYTICS AT A GLANCE



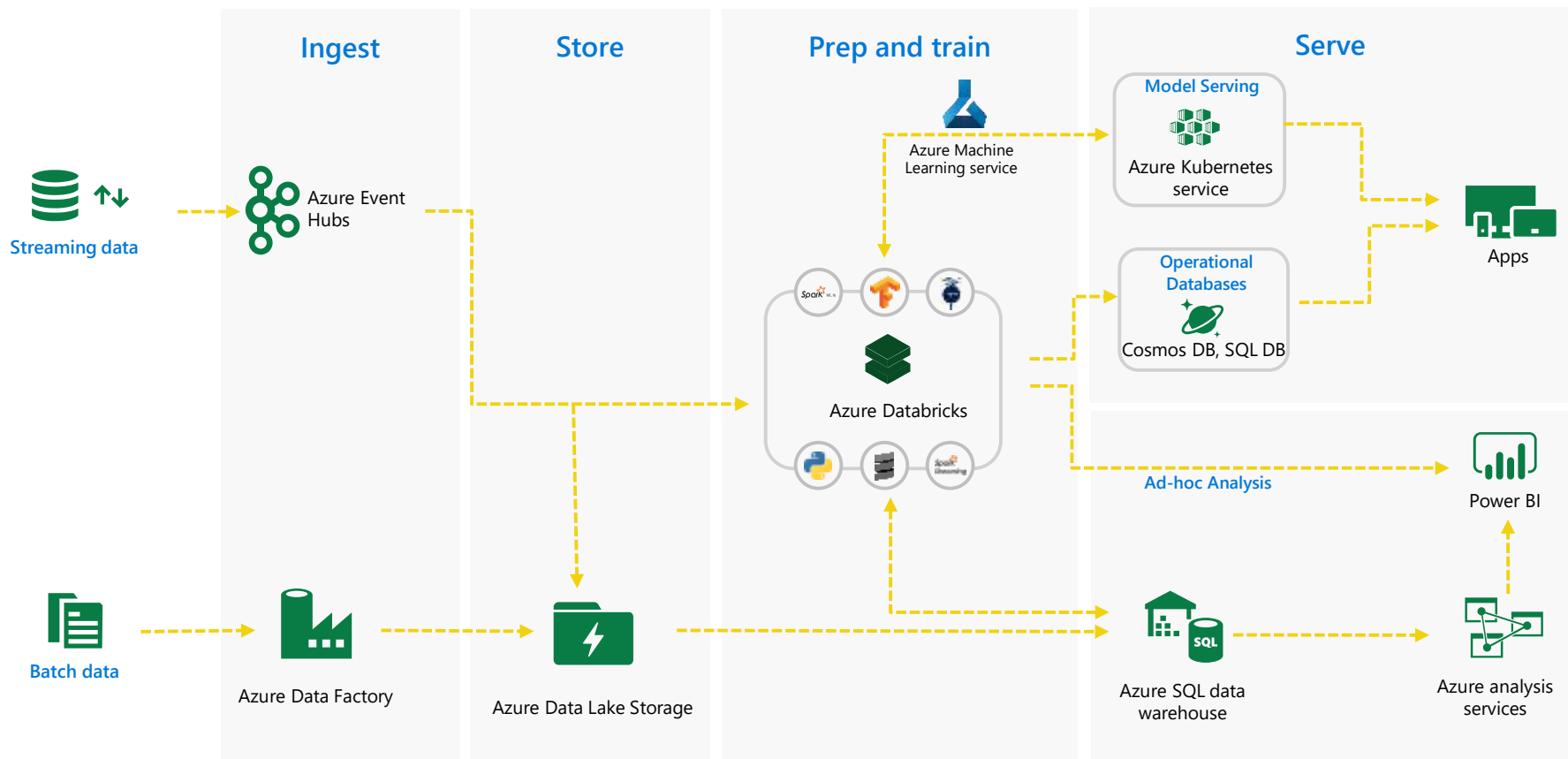
Modern Big Data Warehouse



Advanced Analytics on Big Data



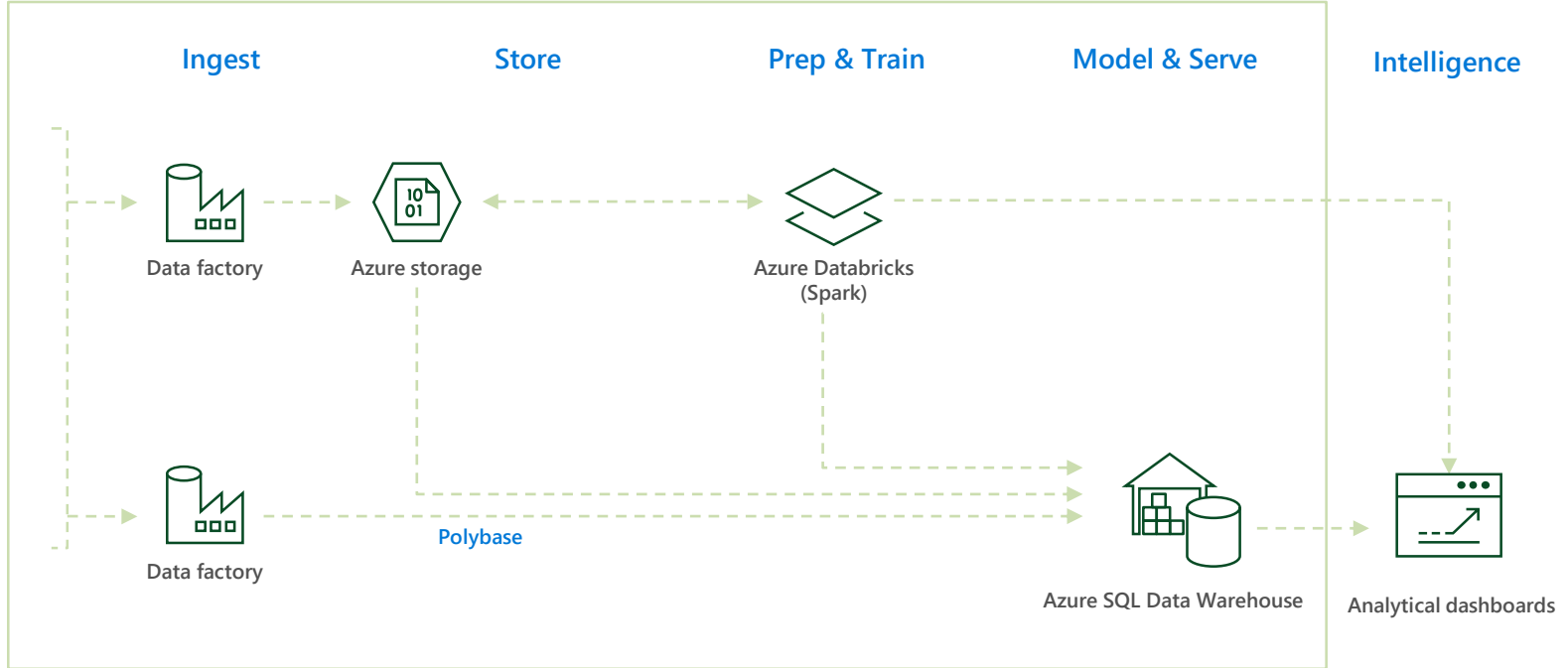
Modern Data Platform with Azure Databricks



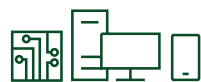
Modern Big Data Warehouse


Logs, files and media
(unstructured)

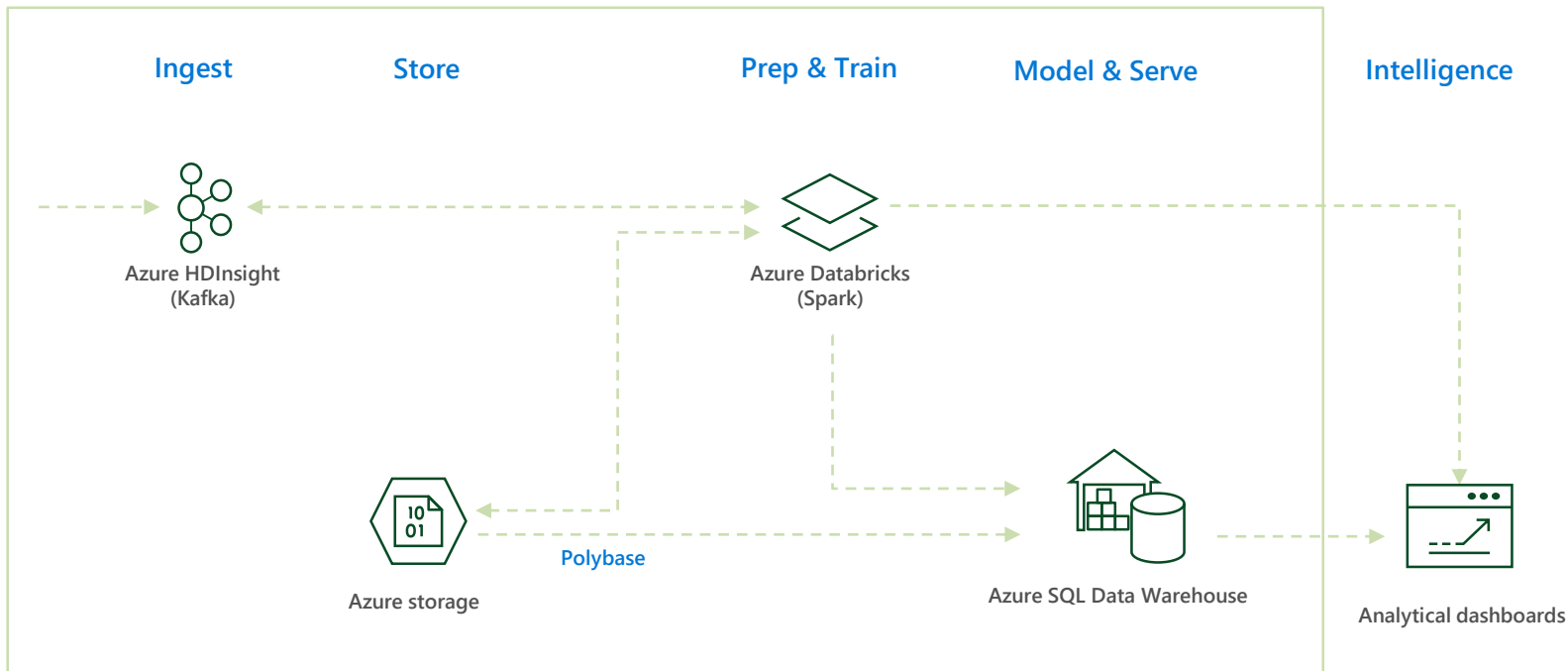

Business / custom apps
(Structured)



Real-time analytics on Big Data



Unstructured data



KNOWING THE VARIOUS BIG DATA SOLUTIONS

