# Networking, Load Balancing, and Security

INTRODUCTION TO KUBERNETES

**Frank Heilmann**

Platform Architect and Freelance Instructor

datacamp

# More on Labels and Selectors

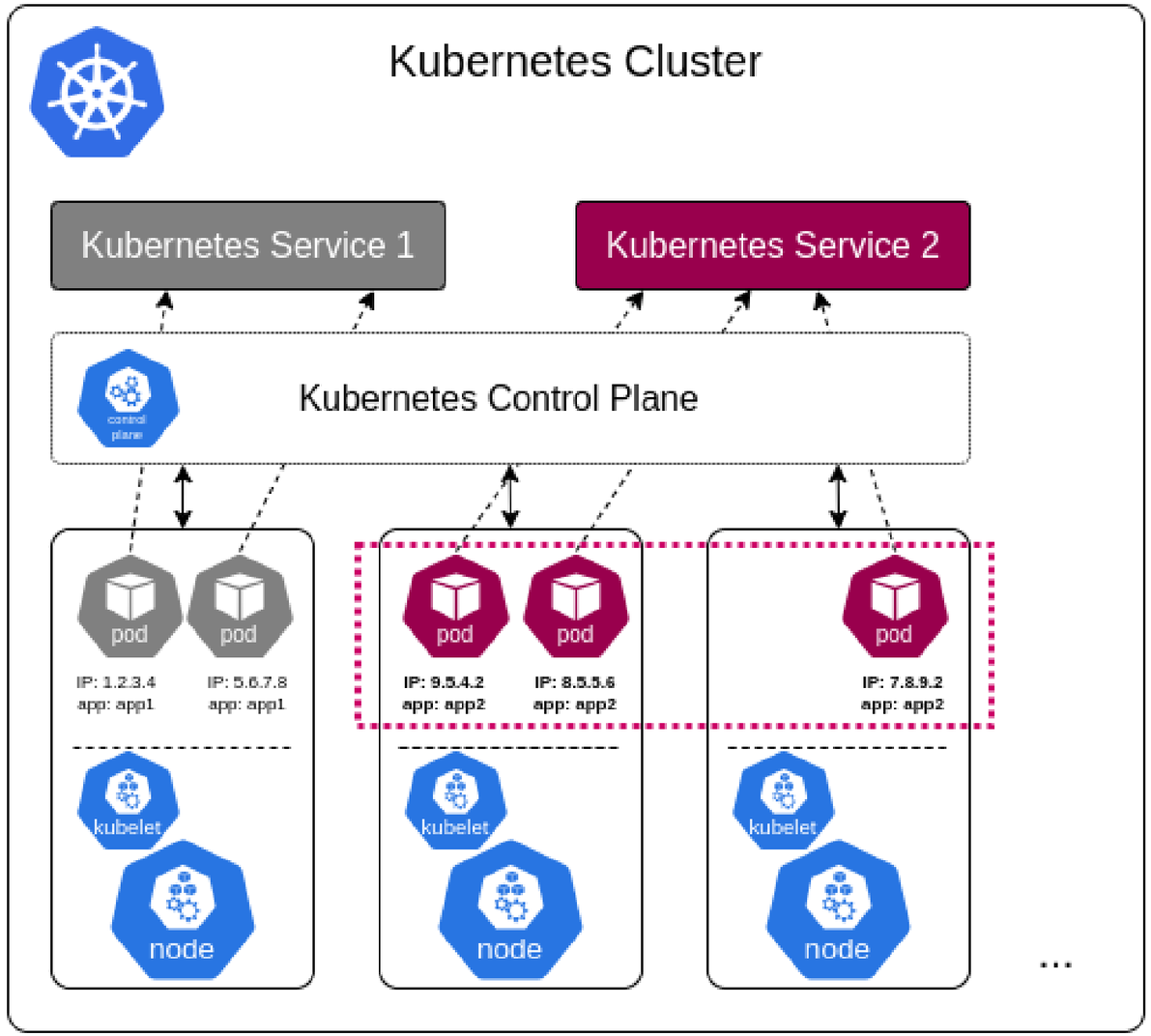
**Labels:**

- Key/Value pairs attached to Kubernetes objects like Pods or Nodes
- Can be used to organize subsets of objects
- Can be modified at any time
- Examples:
  - `environment: prod`
  - `app: my_cool_app`
  - `has_GPU: true`

**Selectors:**

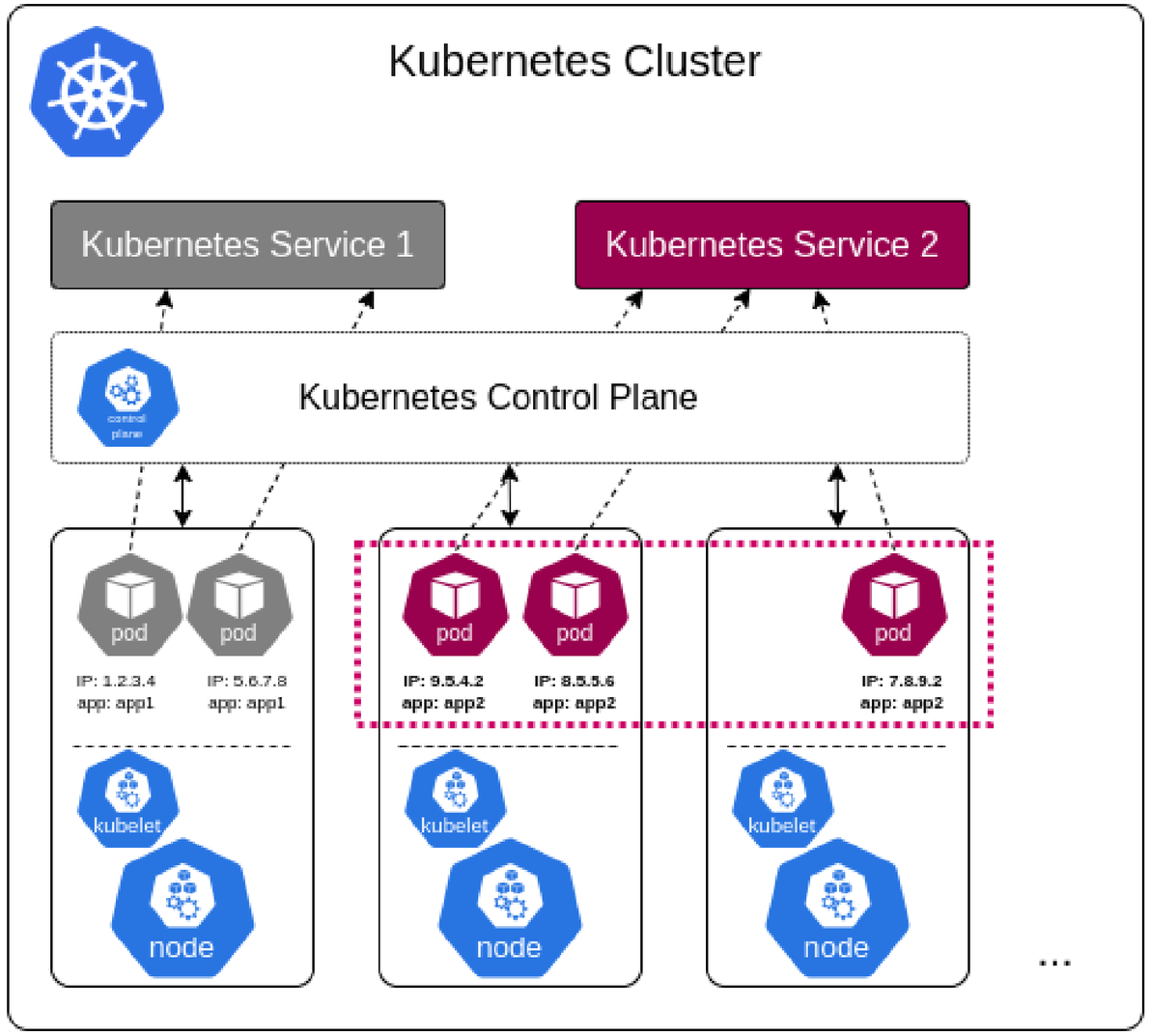
- Can be used to identify objects via labels
- Examples:
  - 
```
...
selector:
    environment: prod
    app: my_cool_app
...
```
  - 
```
...
nodeSelector:
  has_GPU: true
...
```

# Networking and Services



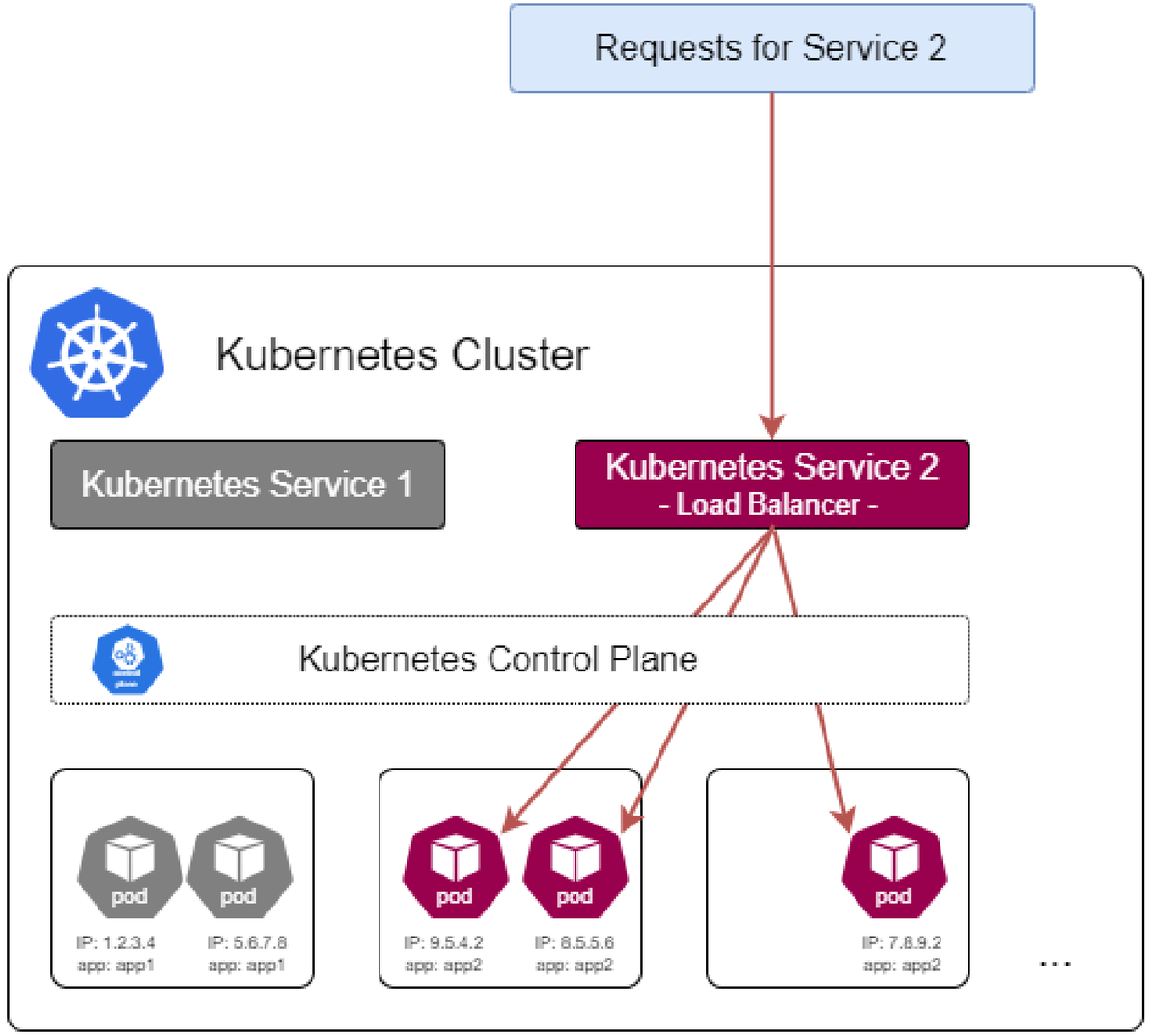

- Each Pod gets its own cluster-wide IP (internet address)
- Can be used for communication between Pods
- Not very useful, as Pods can restart at any time, and will get a new IP
- **Services** are used to attach Pods to, and offer stable connectivity

# Service Manifests




```
apiVersion: v1
kind: Service
metadata:
  name: Kubernetes_Service_2
spec:
  type: ...
  selector:
    app: app2
  ...
```

# Load Balancing



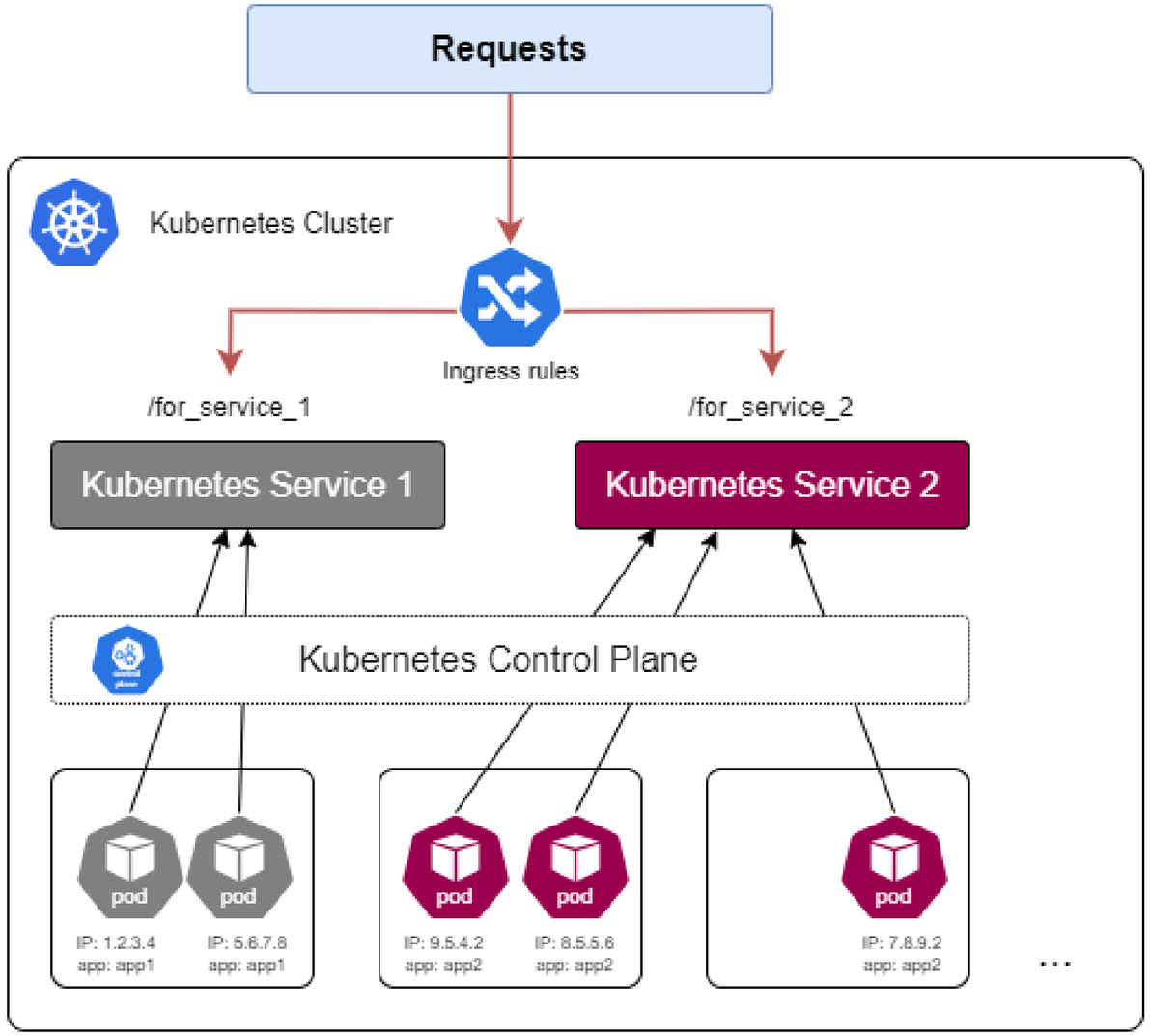

- A **load balancer** in Kubernetes distributes load over Pods
- Avoids uneven load on resources, increases efficiency and lowers response times
- Example:
  - providing a service from multiple Pods
  - load balancer will distribute load evenly to Pods

# Load Balancing in Kubernetes

```
apiVersion: v1
kind: Service
metadata:
  name: <service name>
spec:
  type: LoadBalancer
  selector:
    <key1>: <value1>
    <key2>: <value2>
  ...
```

- Load balancers are typically pre-configured by Kubernetes Provider (Cloud Provider)
- No need to declare additional manifests for a load balancer - will automatically be created and attached to the service

# Ingress




- **Ingress** objects are used to route HTTP and HTTPS requests (traffic) from outside the cluster to services in the cluster
- Ingress rules define which requests are served by which service
- Typically used in combination with load balancing

# Kubernetes Security

- Security in modern IT architectures is an extremely important, but complex field with many facets
- Kubernetes has all necessary components to secure applications running on it, e.g:
  - the "Secret" API for confidential objects like passwords, tokens, keys etc.
  - tools and APIs to enable encrypted network communication
  - methods for authentication of users
  - *role-based* and *attribute-based access control* ("RBAC" and "ABAC")

# Let's practice!

INTRODUCTION TO KUBERNETES

# Data Pipelines on Kubernetes

INTRODUCTION TO KUBERNETES

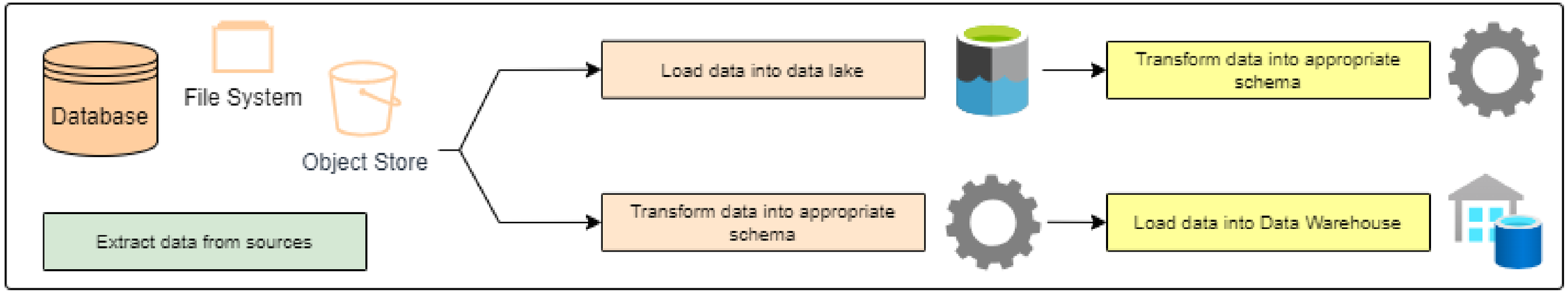
**Frank Heilmann**

Platform Architect and Freelance Instructor

datacamp

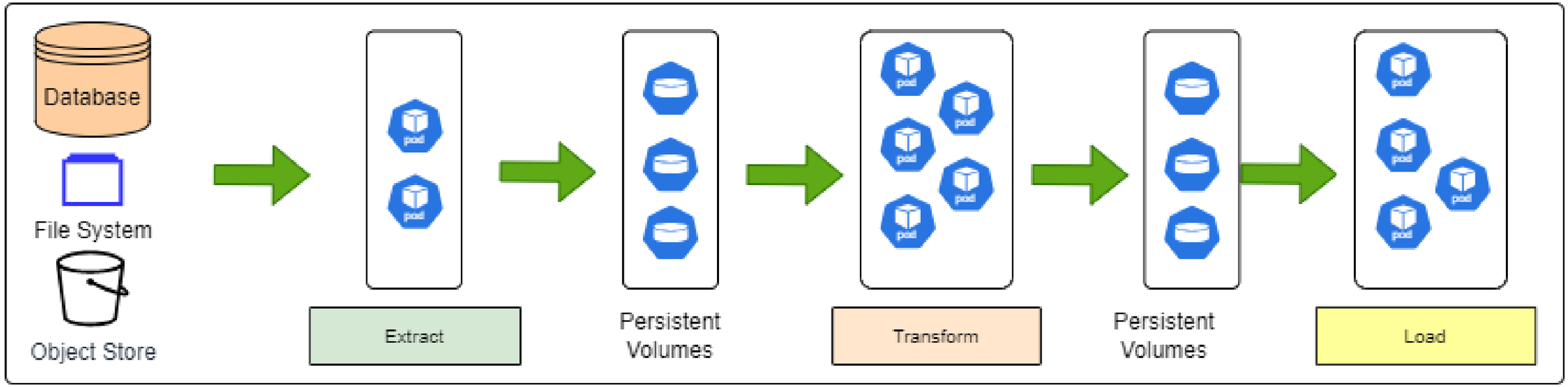
# What are Data Pipelines?

- Set of processes to move, transform, or analyze data
- Typical steps:
  - *ETL*: **Extract** data from various data sources, then **Transform** into a meaningful schema, finally **Load** into a target data sink (e.g., a data warehouse )
  - *ELT*: **Extract** data from various data sources, then **Load** into a target data sink (e.g., a data lake), finally **Transform** data into meaningful schema when needed



# Data Pipelines on Kubernetes

- The steps of a data pipeline map nicely to Kubernetes objects:
  - Extract, Transform, Load steps: Pods (Deployment or StatefulSet)
  - Extracted and Transformed Data: Persistent Volumes
- Kubernetes can scale out Deployments and Storage as required, hence increase throughput



# Open-Source Tools for Data Pipelines

- Many open-source software exists that is readily deployable on Kubernetes
- Some examples:
  - Extract: **Apache NiFi**, **Apache Kafka** with **Kafka Connect**
  - Transform: **Apache Spark**, **Apache Kafka**, **PostgreSQL**
  - Load: **Apache Spark**, **Apache Kafka** with **KSQL**, **PostgreSQL**
  - Storage on top of PVs: **Minio**, **Ceph**
- This list is by no means complete

# Let's practice!

INTRODUCTION TO KUBERNETES

# MLOps on Kubernetes

## INTRODUCTION TO KUBERNETES

**Frank Heilmann**

Platform Architect and Freelance Instructor

datacamp

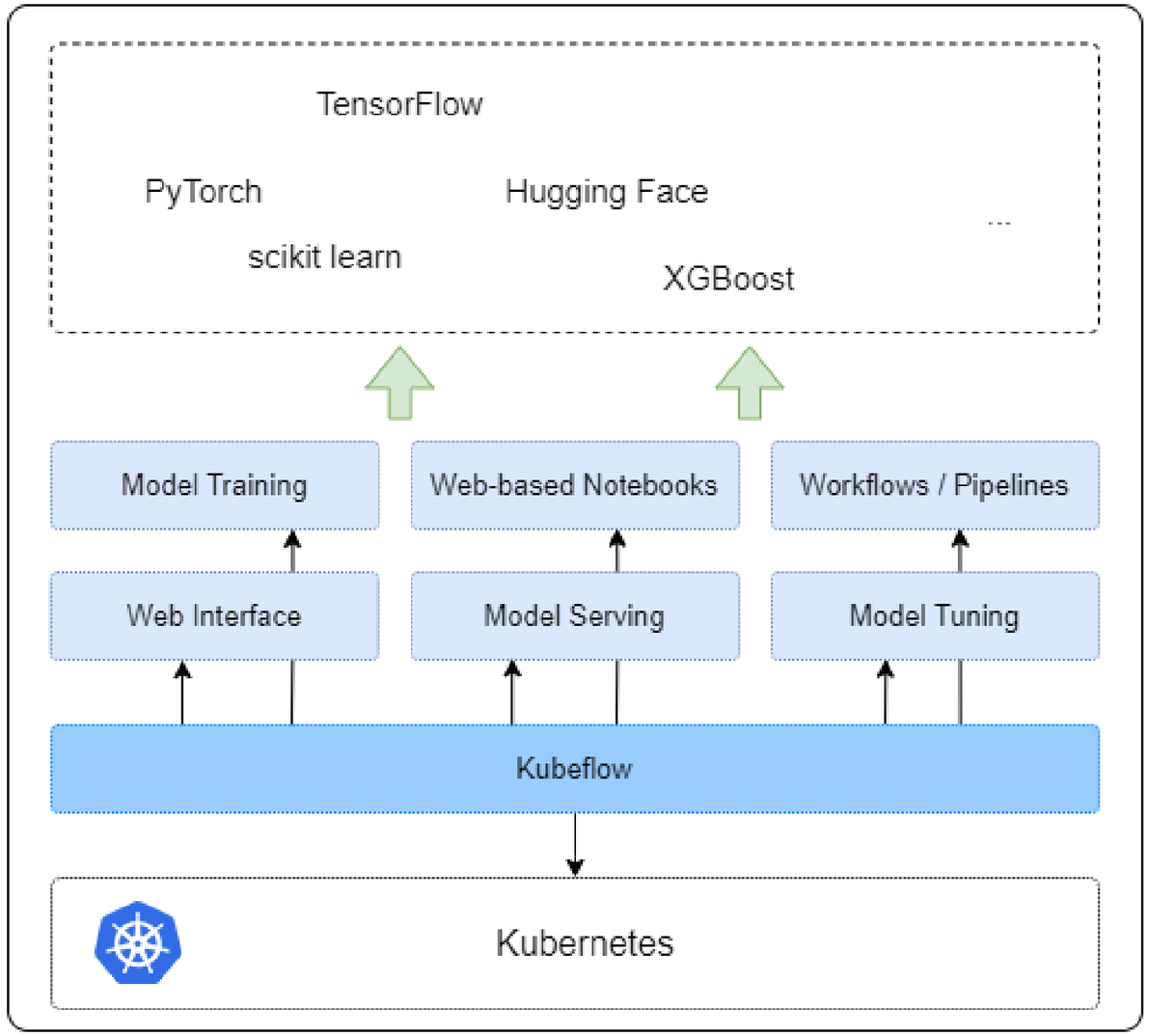
# What is MLOps?

- A paradigm to deploy and maintain machine learning models in production
- A set of best-practice workflows with focus on continuous development of such models
- Inspired by *DevOps*:
  - Machine learning models are developed and tested in isolated experimental systems, and then deployed to production
  - When in production, continuous monitoring; retraining may be triggered
- Data scientists, data engineers, and IT teams can work on deployed models synchronously and ensure model accuracy

# Implementing MLOps on Kubernetes

- The MLOps paradigm maps very well to Kubernetes:
  - Isolated experimental systems: can be realized via Pods and Kubernetes Storage
  - Monitoring production ML models: enabled via lifecycle of Pods (and deployed image versions)
  - Synchronous work on model accuracy: built in from the very beginning by Kubernetes architecture
- Several frameworks for MLOps exist; the two best-known open-source solutions are
  - **mlflow**
  - **Kubeflow**

# Kubeflow - An Overview




- **Kubeflow** allows simple deployments of ML workflows specifically on Kubernetes
- Covers each step of the ML model lifecycle
- Consists of several components which cover these steps, working independently
- Python can be used to develop and deploy ML models via an API
  - no need to use `kubectl`

# Let's practice!

INTRODUCTION TO KUBERNETES

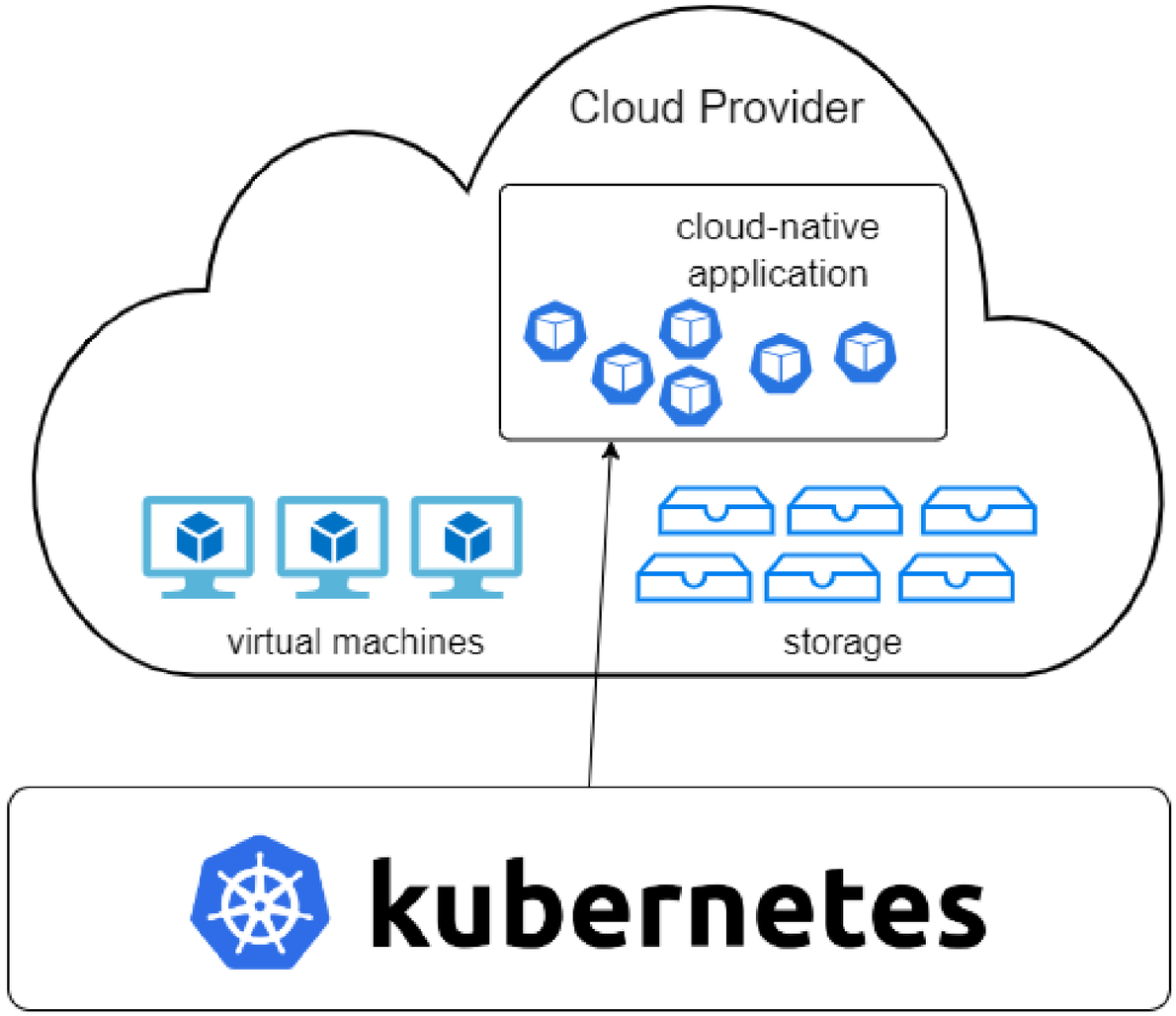
# Wrap-Up

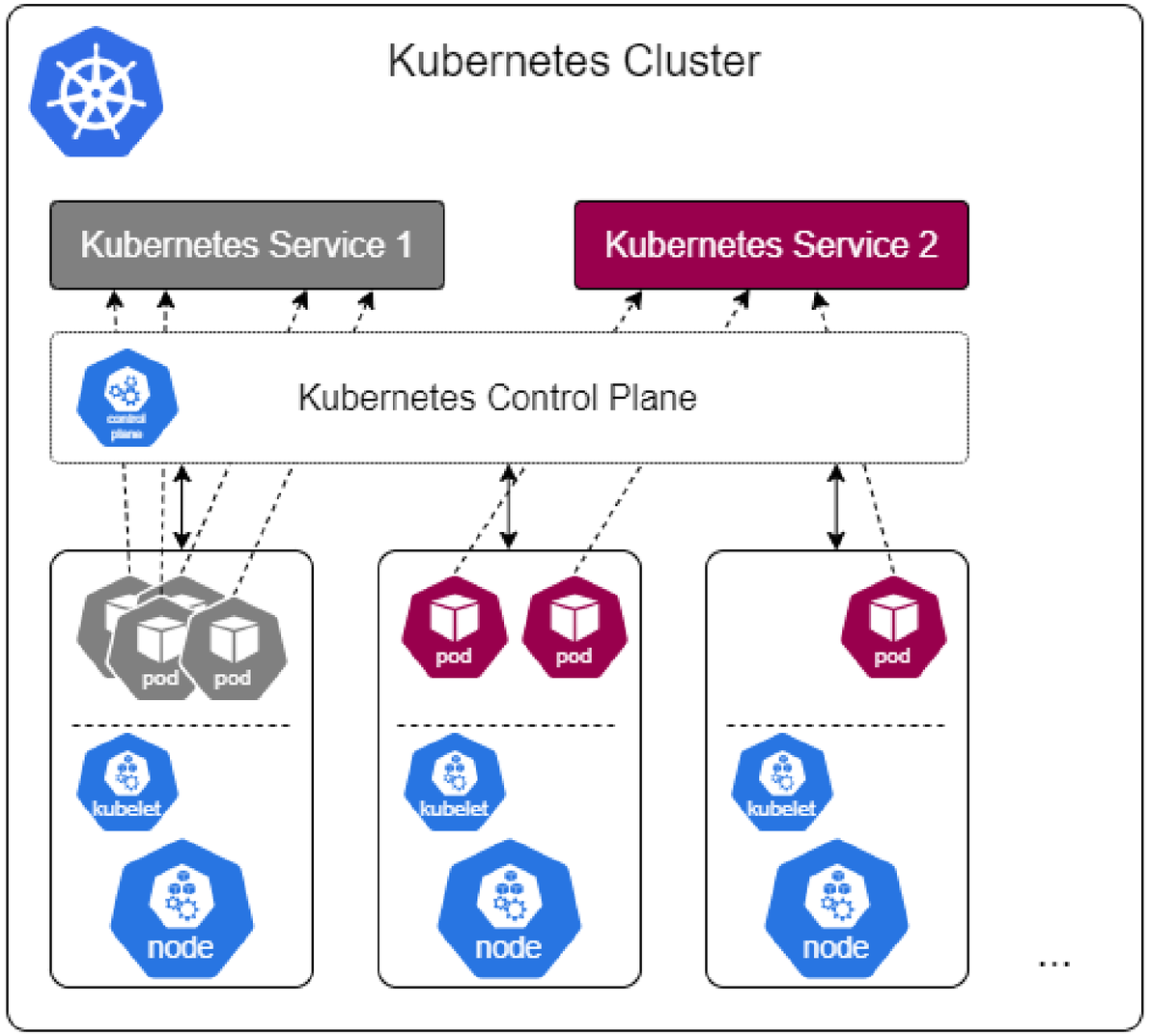
## INTRODUCTION TO KUBERNETES

**Frank Heilmann**

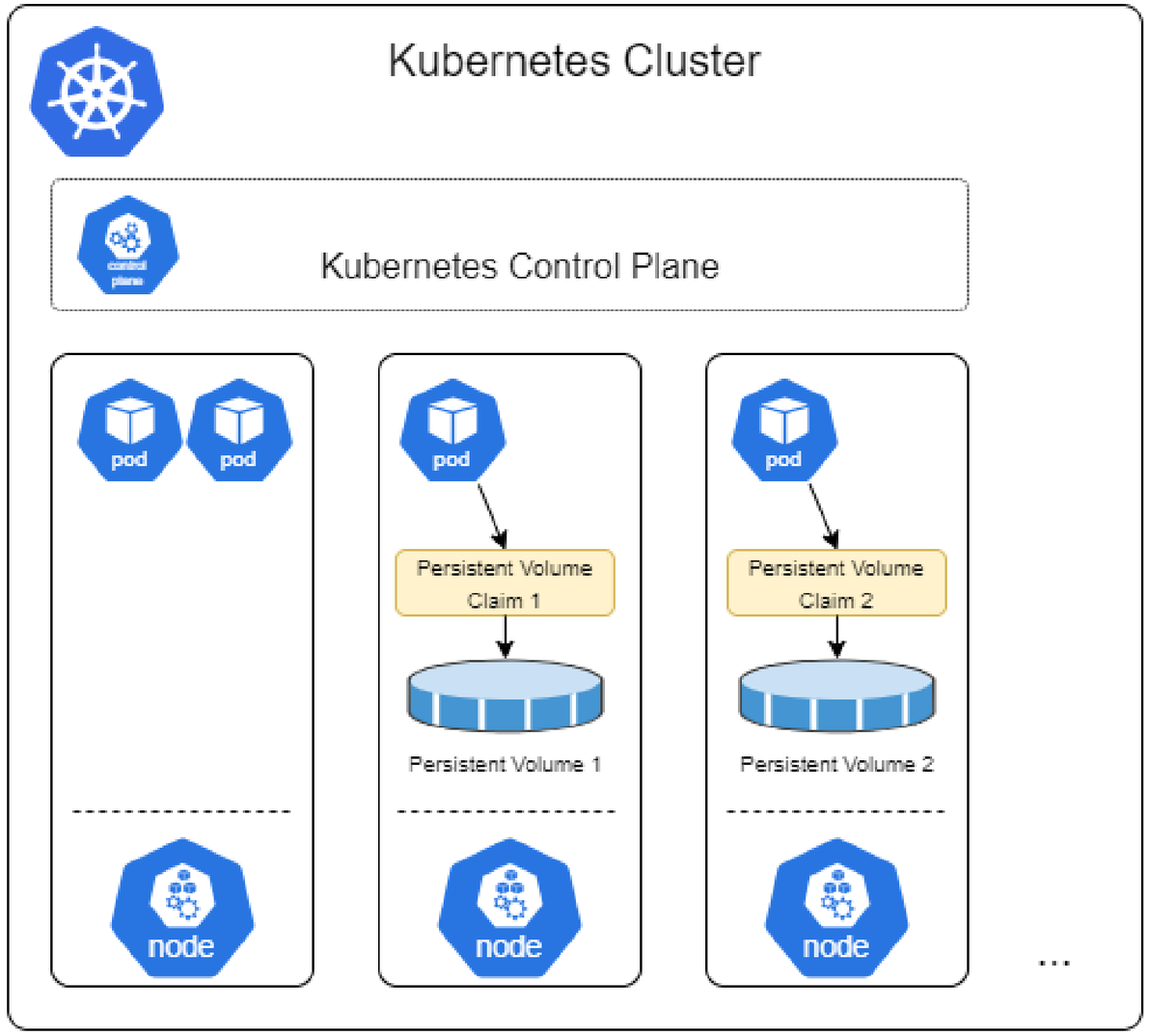
Platform Architect and Freelance Instructor

# What we have learned (1/2)



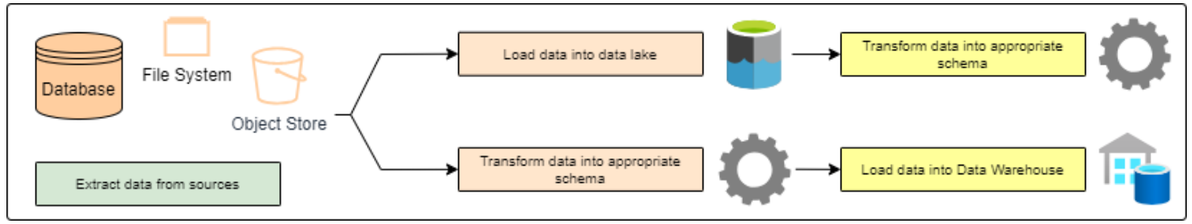



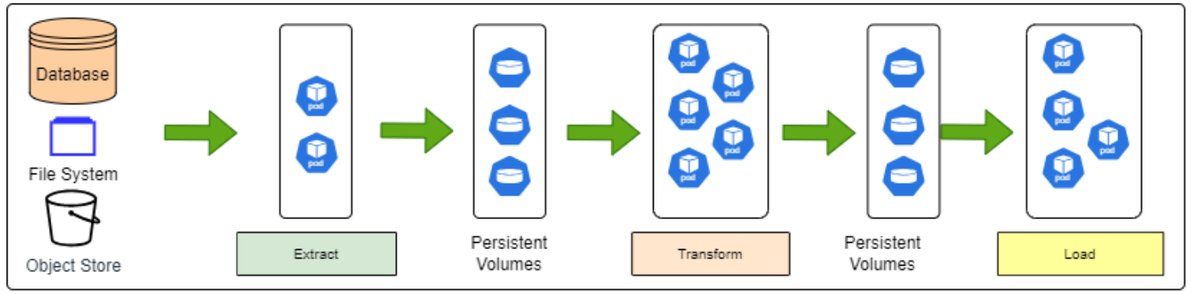
# What we have learned (2/2)




```
kubectl apply ...
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  ...
spec:
  ...
```



# Where to go from here

- Security for cloud-native applications: **Service Accounts**, **Secrets**, **RBAC**, etc.
- Packaging and application lifecycle: **Helm Charts** and **Kubernetes Operators**
- Advanced storage concepts: **Rook Operators** and **Ceph**
- CI/CD
- Serverless applications: **KNative**

# Thank You!

INTRODUCTION TO KUBERNETES