PURESTORAGE®

K8s for SEs Workshop

Andy Clemenko

SE
Pure Storage

January 2026

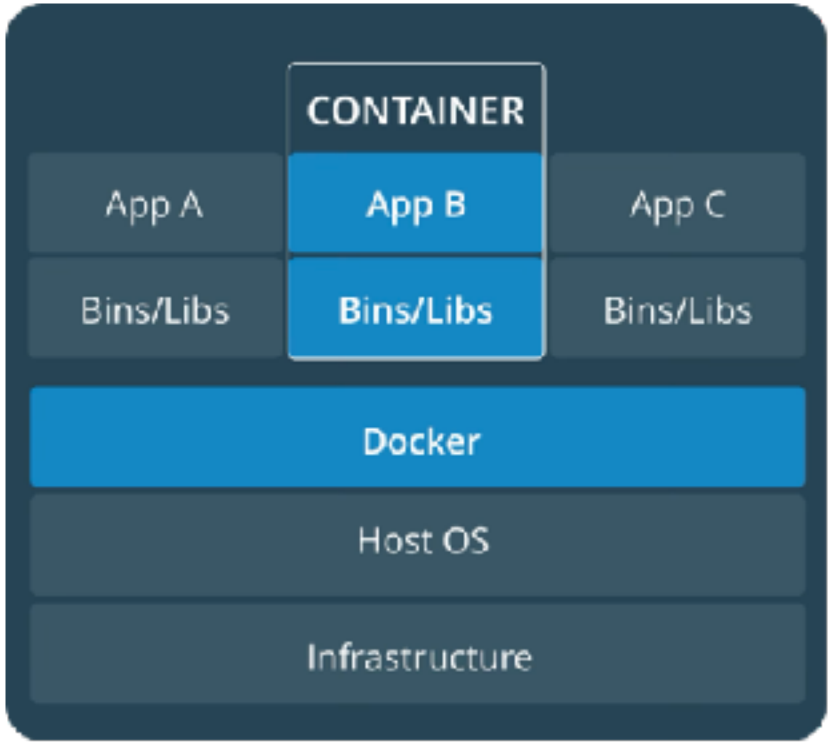
Workshop Environment

rfed.io/workshop

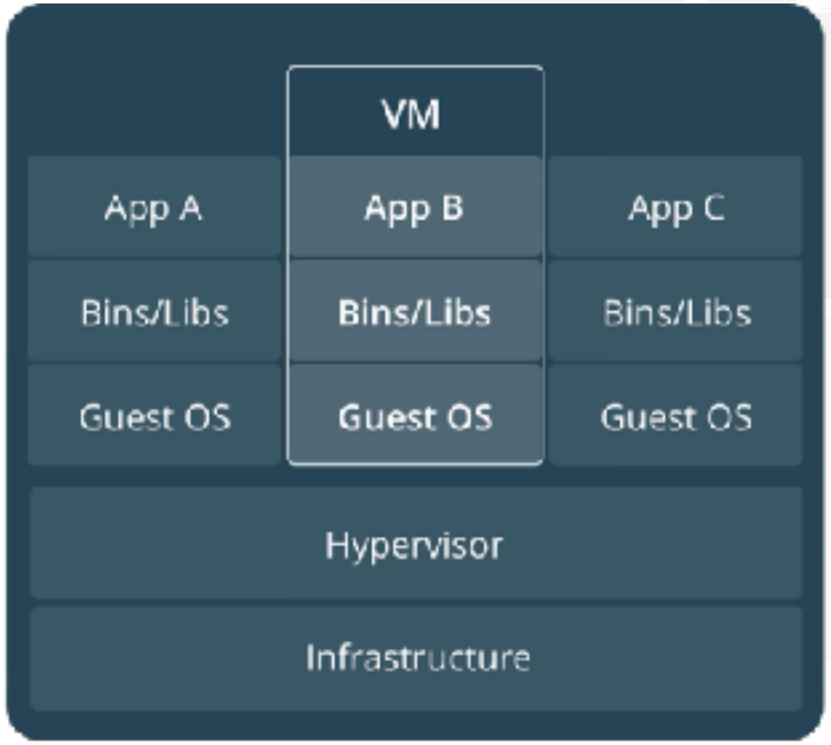
Pa22word

Containers…

Logical Separation

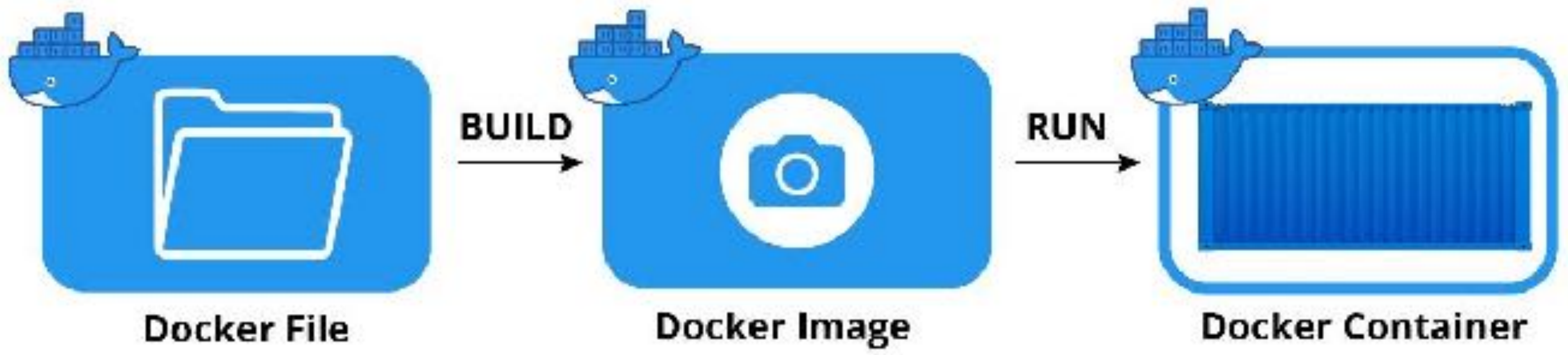


Containers are an app level construct



VMs are an infrastructure level construct to turn one machine into many servers

Life of a Container



Recipe:
- Shell
- Package Manager
- Application/Framework
- Specific code/files

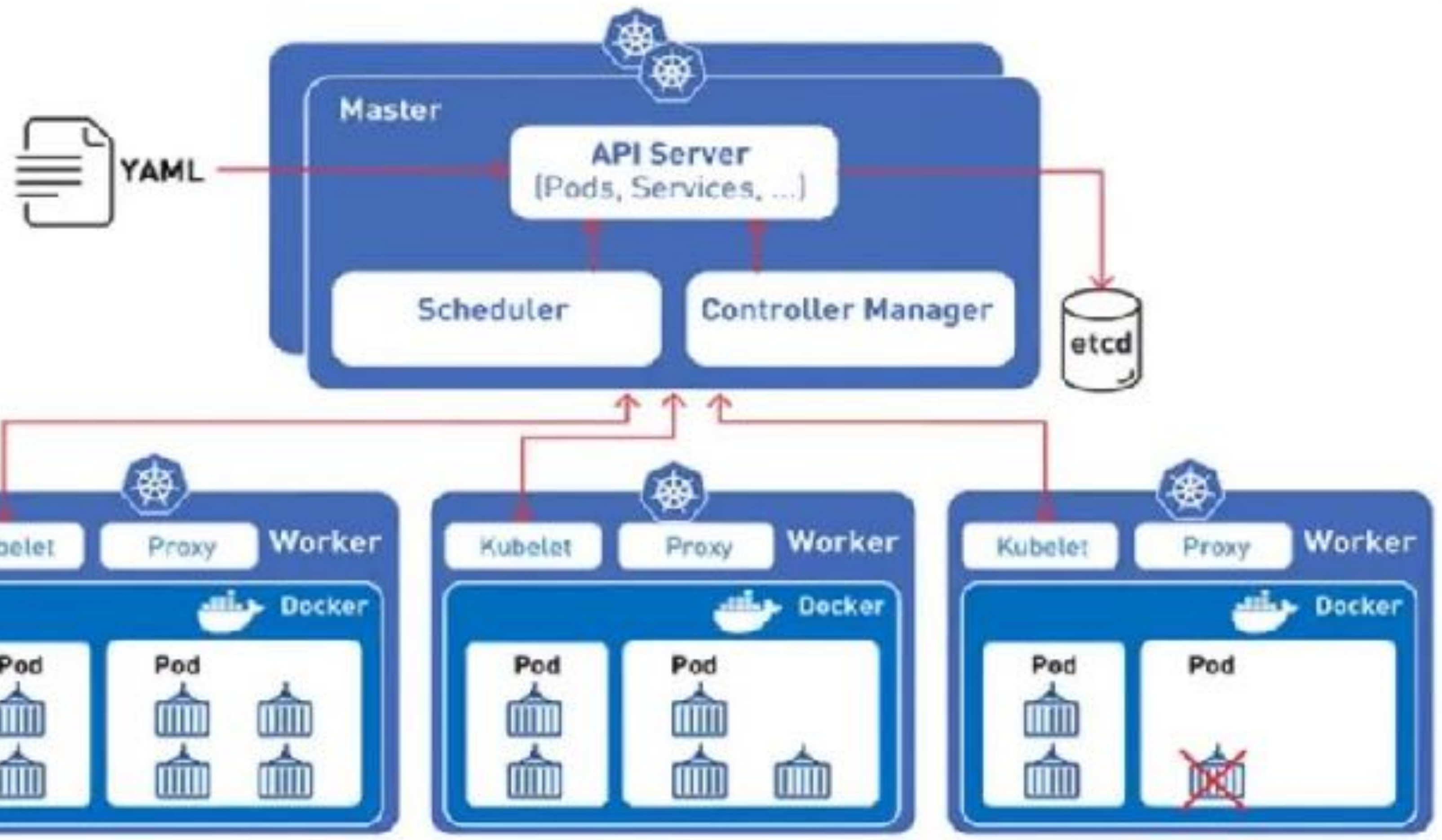
Just a zip file

Copy of the zip file

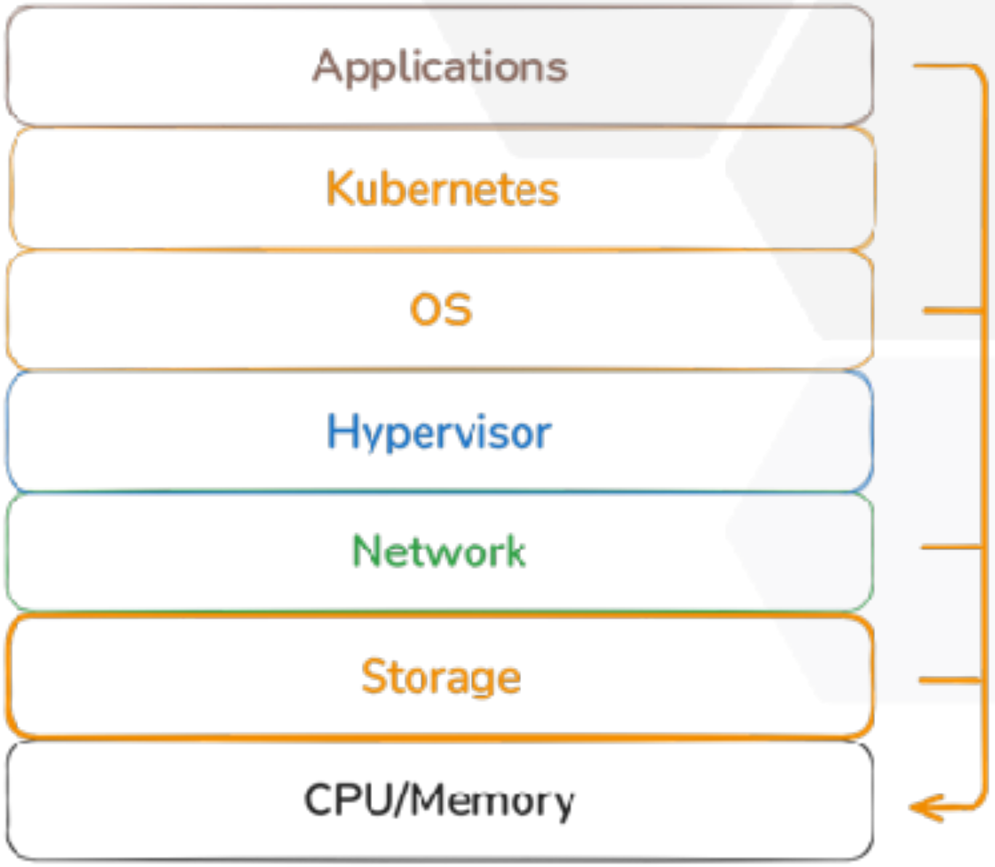
kubernetes

Just a traffic cop!

Kubernetes Architecture



Schedule Resources



K8s Object types: Kinds

https://octopus.com/devops/kubernetes-deployments/kubernetes-yaml-types/

Workload resources

Workload resources describe how containers are deployed and managed in a cluster:

1. **Pod:** The basic execution unit in Kubernetes. A pod encapsulates one or more containers, shared storage, network, and a specification for how to run the containers. Pods are ephemeral and typically managed by higher-level controllers like deployments.
2. **Deployment:** Provides updates for stateless applications. It maintains the desired number of replicas, enables rolling updates, and allows rollback to previous versions if needed. Deployments are commonly used for web services and APIs.
3. **StatefulSet:** Manages stateful applications requiring stable network identities and persistent storage. Unlike deployments, StatefulSets assign each pod a unique, consistent identity across restarts. Suitable for databases and distributed systems.
4. **DaemonSet:** Ensures a pod runs on all (or selected) nodes. Common for system-level services like log collectors, monitoring agents, and network proxies.
5. **Job:** Runs a finite task to completion. Once the task is completed successfully, the pod terminates. Useful for batch processing or data migrations.
6. **CronJob:** Schedules jobs to run periodically using cron syntax. Commonly used for recurring tasks such as backups or report generation.

K8s Object types: Kinds

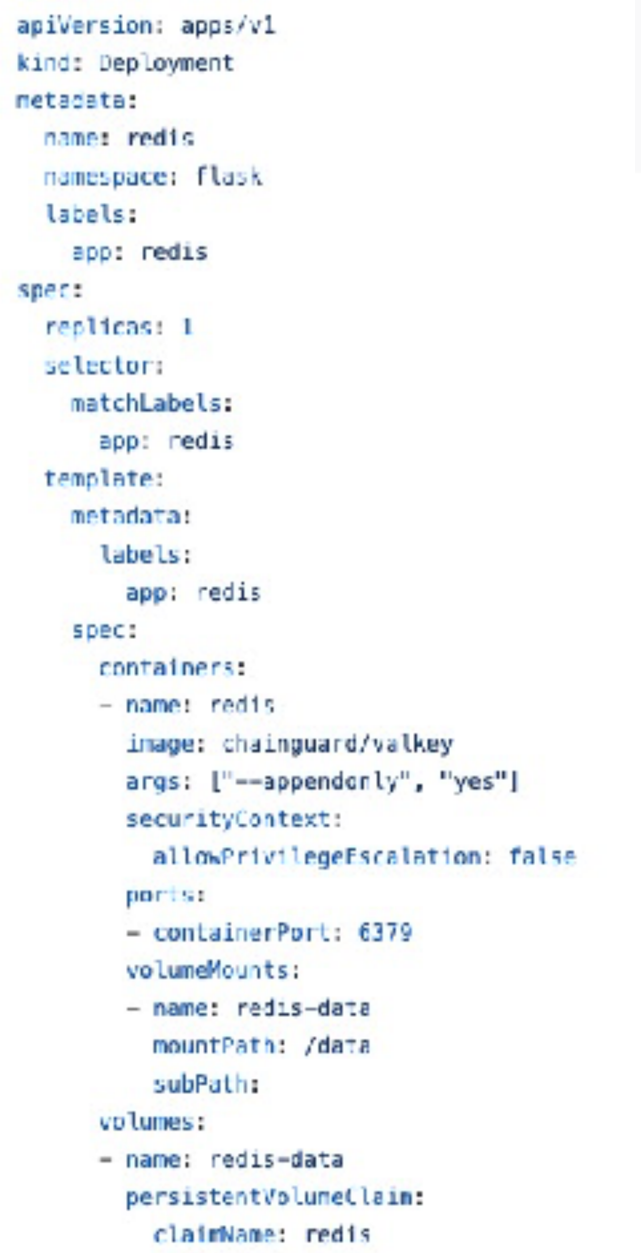
https://octopus.com/devops/kubernetes-deployments/kubernetes-yaml-types/

Storage resources

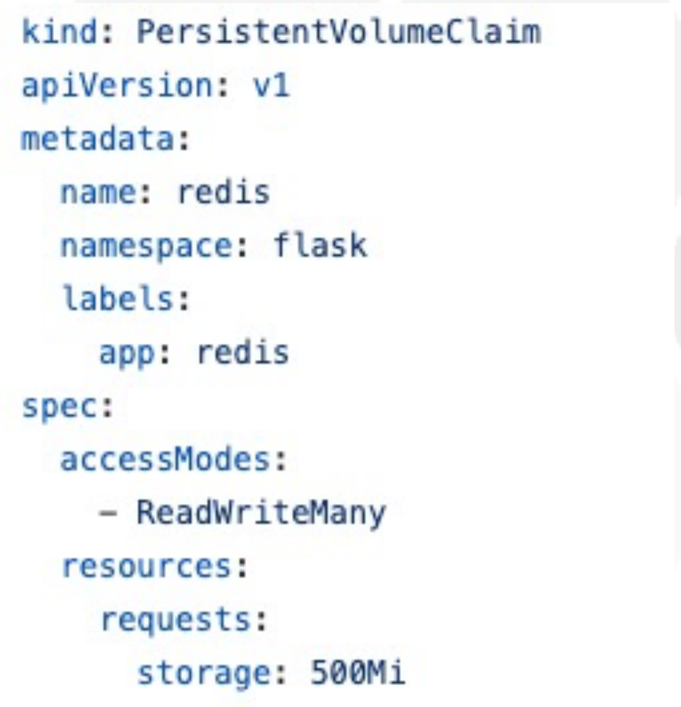
Storage objects provide persistent data handling across pod restarts:

15. **PersistentVolume (PV):** Represents a piece of networked storage provisioned by an admin or dynamically created. It abstracts the underlying storage technology (e.g., NFS, iSCSI, cloud volumes).
16. **PersistentVolumeClaim (PVC):** A user's request for storage, specifying size and access mode. Kubernetes binds the claim to an available PV that matches the request.
17. **StorageClass:** Defines the provisioner and parameters for dynamically creating PVs. Different classes can represent performance tiers or backup policies, allowing flexible storage provisioning.
18. **Volume:** Defined within the pod spec to mount storage. Volumes can be ephemeral (like emptyDir) or persistent (like those backed by PVCs). They provide shared storage between containers in a pod.

K8s Yaml Example:



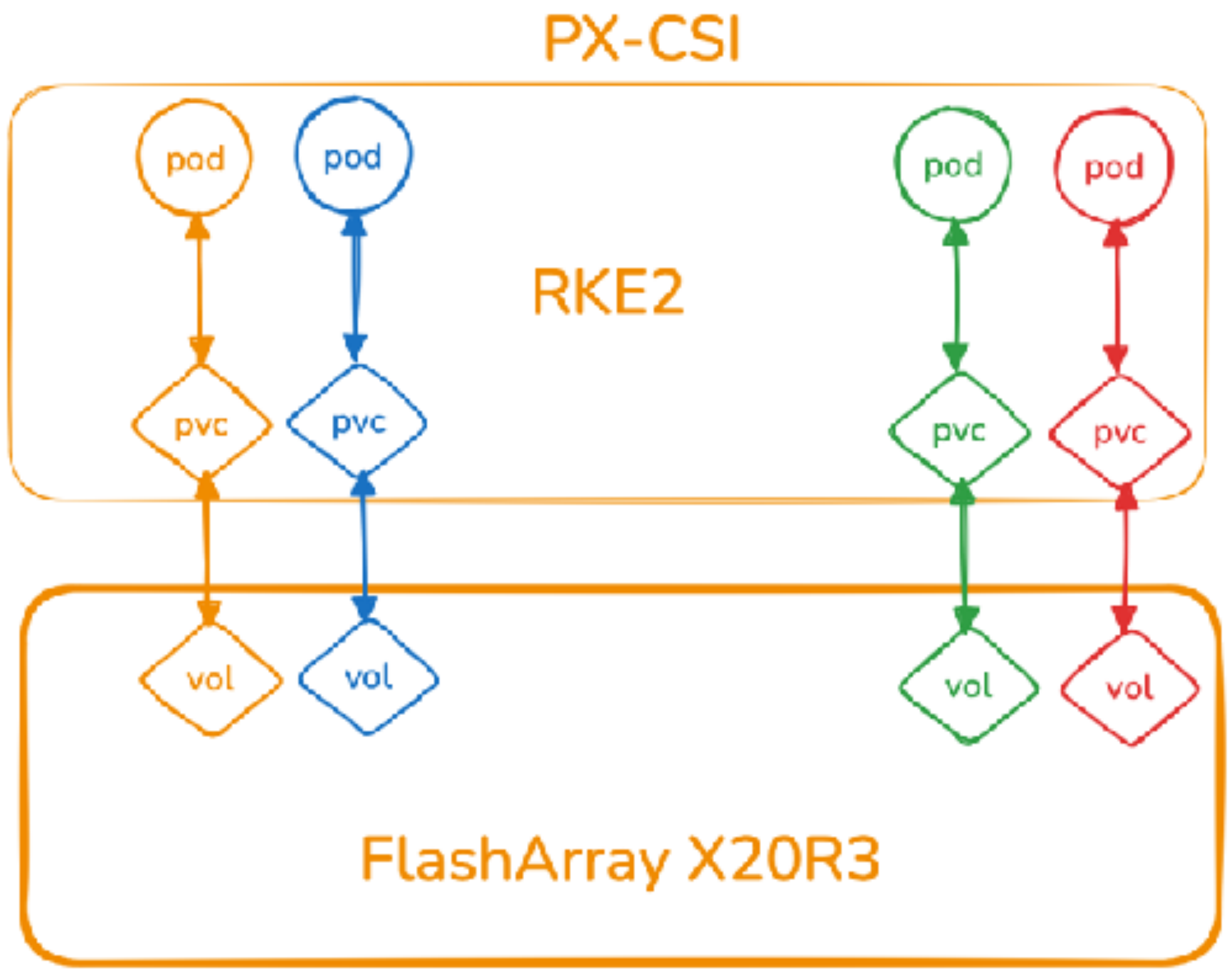
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis
  namespace: flask
  labels:
    app: redis
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
    spec:
      containers:
      - name: redis
        image: chainguard/valkey
        args: ["--appendonly", "yes"]
        securityContext:
          allowPrivilegeEscalation: false
        ports:
        - containerPort: 6379
        volumeMounts:
        - name: redis-data
          mountPath: /data
          subPath:
      volumes:
      - name: redis-data
        persistentVolumeClaim:
          claimName: redis
```



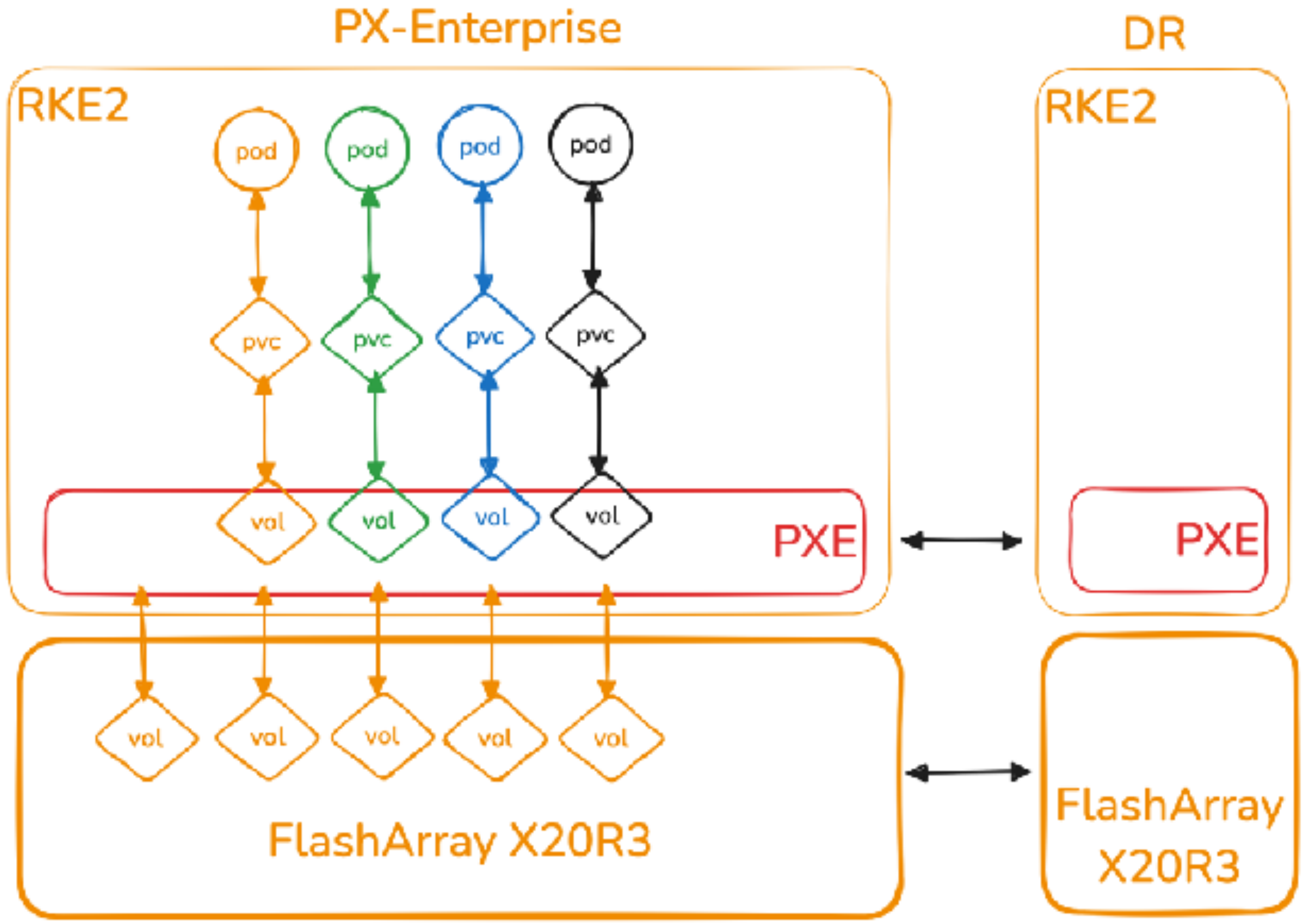
```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: redis
  namespace: flask
  labels:
    app: redis
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 500Mi
```

Your Turn

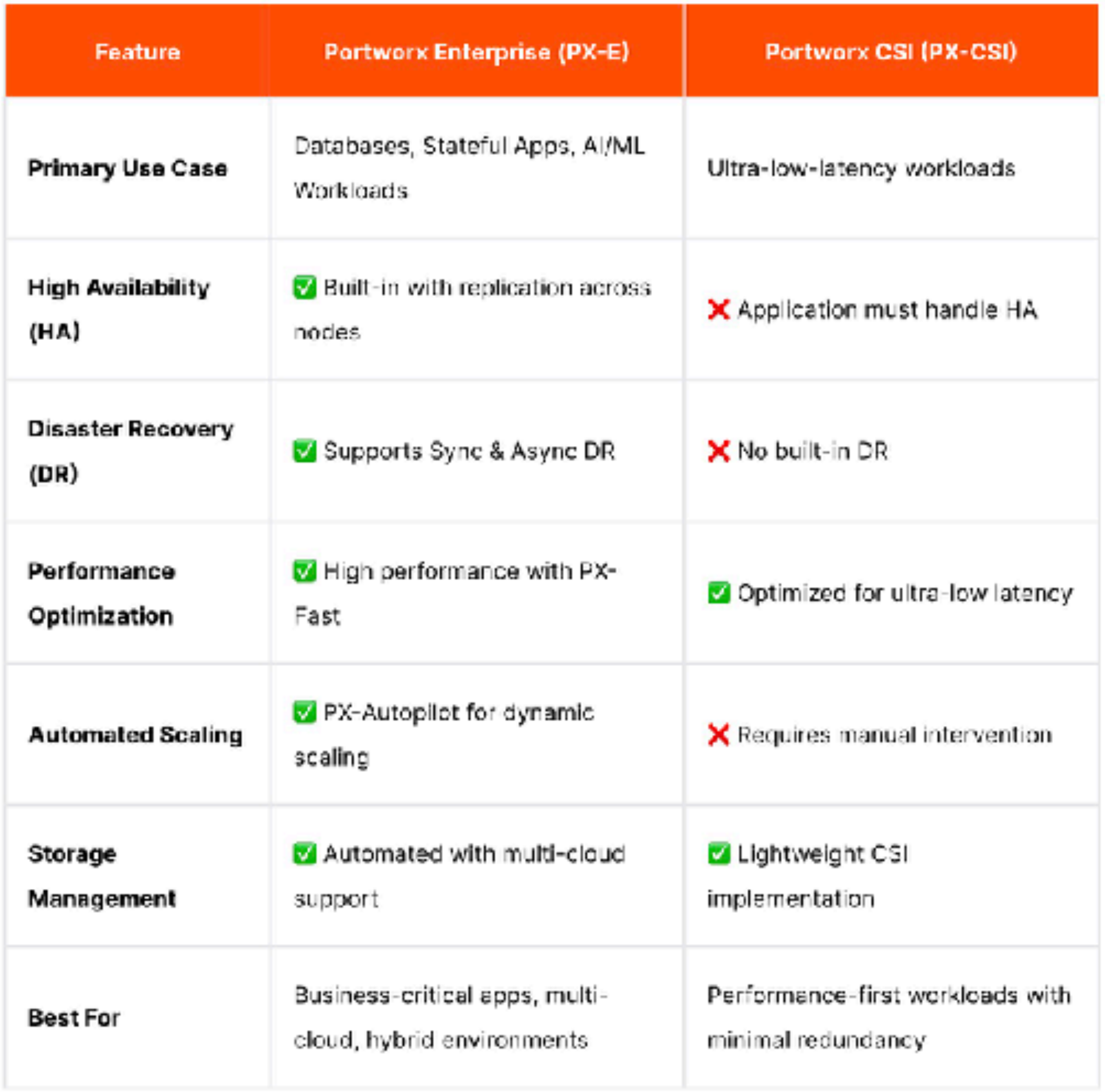
What did we see - Logical View:



PX - Enterprise Logical View:



PX Versions



Feature	Portworx Enterprise (PX-E)	Portworx CSI (PX-CSI)
Primary Use Case	Databases, Stateful Apps, AI/ML Workloads	Ultra-low-latency workloads
High Availability (HA)	✅ Built-in with replication across nodes	❌ Application must handle HA
Disaster Recovery (DR)	✅ Supports Sync & Async DR	❌ No built-in DR
Performance Optimization	✅ High performance with PX-Fast	✅ Optimized for ultra-low latency
Automated Scaling	✅ PX-Autopilot for dynamic scaling	❌ Requires manual intervention
Storage Management	✅ Automated with multi-cloud support	✅ Lightweight CSI implementation
Best For	Business-critical apps, multi-cloud, hybrid environments	Performance-first workloads with minimal redundancy

Mark’s Questions:

Containers:
What is a container
What they run on (a VM or Fe)
What manages them (do you need K8)
How do you assign resources (CPU, Memory, Networking, Storage) to a Container
Types of Container storage
What is K8s

PX :
Where does PortWorx fit? When do Containers need it?
How do you install PortWorx? Where is it installed? Is it a VM? Is it an installed on a container?
How do you manage PortWorx?
How do you create and assign disk (or is it volumes) with PortWorx.