Redhat OCP Migration

From 3.x to 4.x

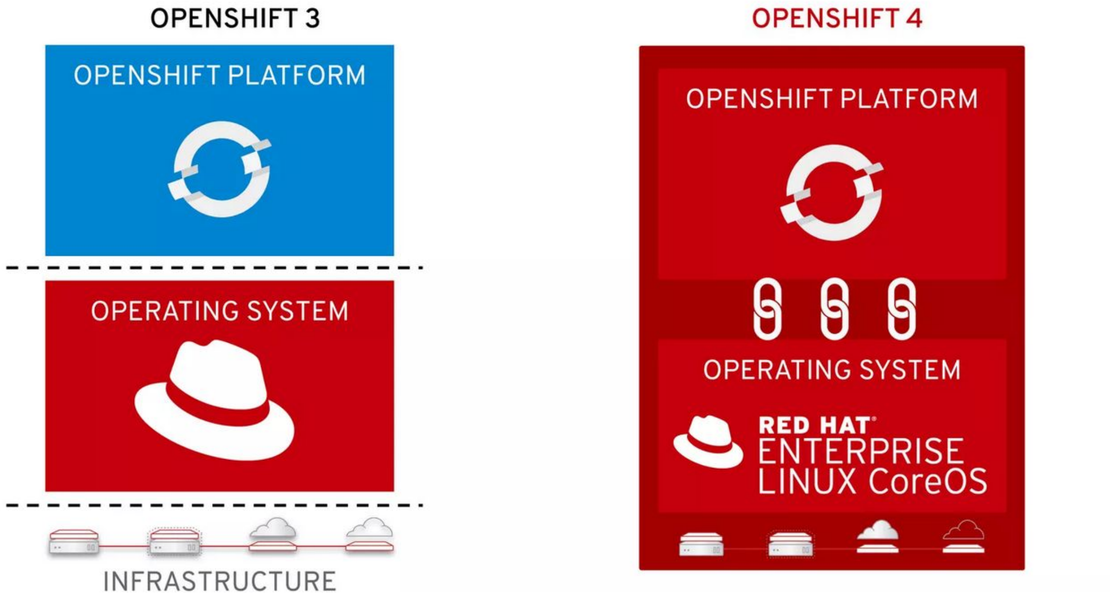
Why

- **OCP 3.9 reached EOL 1 Year back**
    - Security risk
        - Redhat doesn’t provide security patches anymore for this version
- **Outdated K8s version**
    - OCP 3.9 based on K8s 1.9 (current version 1.26)
        - Outdated K8s API versions
- **Bring the infrastructure to state-of-the-art**

OCP 3.x vs OCP 4.x

Difference:

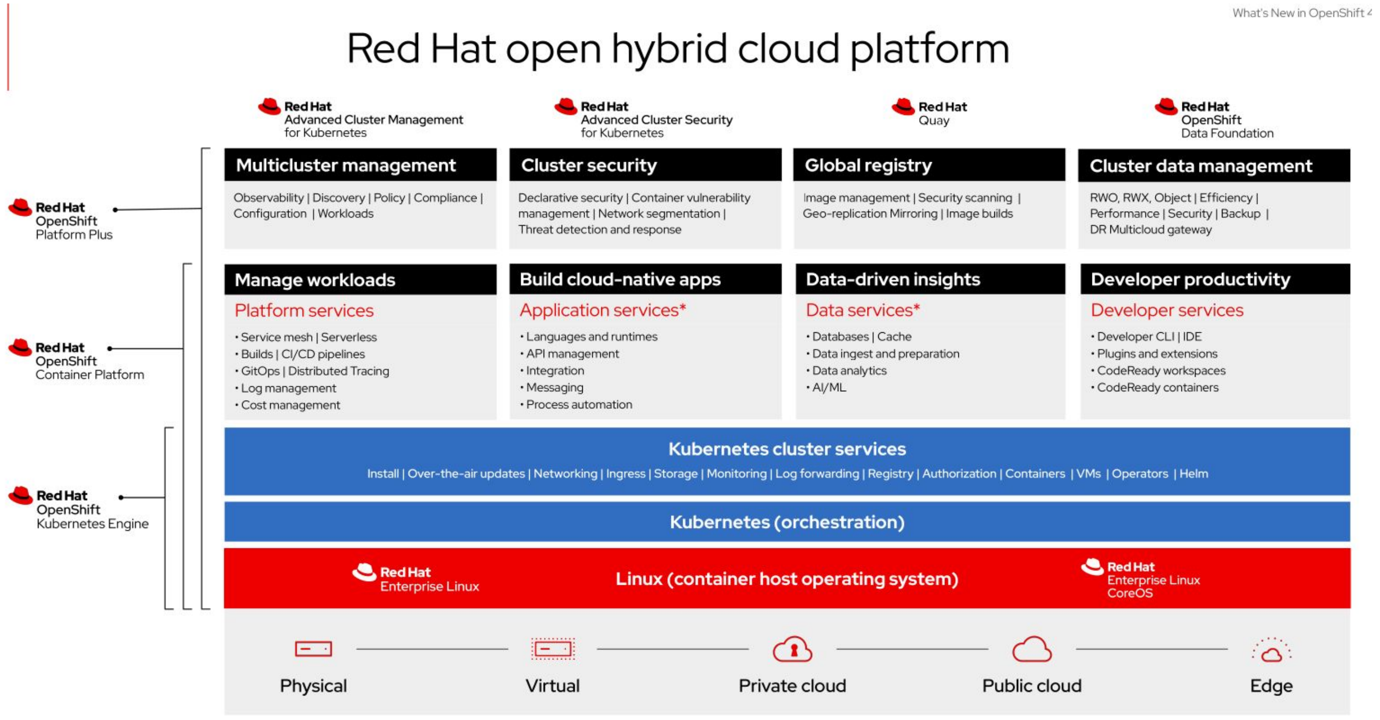
- **Installation**
- **Architecture**
- **Maintenance**



OCP 4.x

- **Red Hat CoreOS (RHCOS)**
  - Immutable, lower footprint, optimized for running containers and managing Kubernetes clusters at scale
- **Operator-based management**
  - In OCP 3.x, the cluster is upgraded by running Ansible playbooks
  - In OCP 4.x, the cluster manages its own updates, including updates to RHCOS on cluster nodes
- **Automated installation and day-2 Operations**
  - Makes it easier to administrate and upgrade

Redhat OCP 4.x platform



What's New in OpenShift 4

Red Hat open hybrid cloud platform

Red Hat Advanced Cluster Management for Kubernetes — Red Hat Advanced Cluster Security for Kubernetes — Red Hat Quay — Red Hat OpenShift Data Foundation

Multicluster management
Observability | Discovery | Policy | Compliance | Configuration | Workloads

Cluster security
Declarative security | Container vulnerability management | Network segmentation | Threat detection and response

Global registry
Image management | Security scanning | Geo-replication Mirroring | Image builds

Cluster data management
RWO, RWX, Object | Efficiency | Performance | Security | Backup | DR Multicloud gateway

Manage workloads
Platform services
- Service mesh | Serverless
- Builds | CI/CD pipelines
- GitOps | Distributed Tracing
- Log management
- Cost management

Build cloud-native apps
Application services*
- Languages and runtimes
- API management
- Integration
- Messaging
- Process automation

Data-driven insights
Data services*
- Databases | Cache
- Data ingest and preparation
- Data analytics
- AI/ML

Developer productivity
Developer services
- Developer CLI | IDE
- Plugins and extensions
- CodeReady workspaces
- CodeReady containers

Kubernetes cluster services
Install | Over-the-air updates | Networking | Ingress | Storage | Monitoring | Log forwarding | Registry | Authorization | Containers | VMs | Operators | Helm

Kubernetes (orchestration)

Red Hat Enterprise Linux — **Linux (container host operating system)** — Red Hat Enterprise Linux CoreOS

Red Hat OpenShift Platform Plus
Red Hat OpenShift Container Platform
Red Hat OpenShift Kubernetes Engine
Physical
Virtual
Private cloud
Public cloud
Edge

Migration

- **Plan**
  - PoC -> Non-prod -> Prod
  - Cutover from 3.x to 4.x
  - Validation (Functional tests, performance tests, e2e tests with sonobuoy)
  - Decommission of the old Cluster
- **Migration requirements**
  - Stable and reliable OCP 4.x using WMware vSphere on-Premises DC
  - Ensure Platform stays in Redhat Supported version
  - Migrate workloads from 3.x to 4.x with Kubernetes API conformance
  - Business process continuity while supporting & enabling migration
- **Post migration**
  - Improve experience for Developers/SRE for application operations with GitOps
  - Improve security posture of application operations

OCP 4.x Installation

- **HA control plane**
  - Minimum of 3 master nodes (Raft Consensus Algorithm)
- **Load balancing with Metallb**
  - Adds a fault-tolerant external IP address for the K8s services
- **Cluster autoscaler**
  - Automatically adjusts the size of a Kubernetes Cluster so that all pods have a place to run and there are no unneeded nodes.
- **Multicluster**
  - Prod
  - Staging
  - Dev

MultiCluster

- **Dev Cluster**
    - Contains development builds, workloads and CI/CD
- **Prod Cluster**
    - Where applications are securely deployed and monitored
- **Staging Clusters**
    - Replicates the Prod and dev Cluster
    - Used to test changes/ regular upgrades without impacting the production to ensure that the cluster is running the latest features and bug fixes
    - Could be immutable to save costs

MultiCluster Pros & Cons

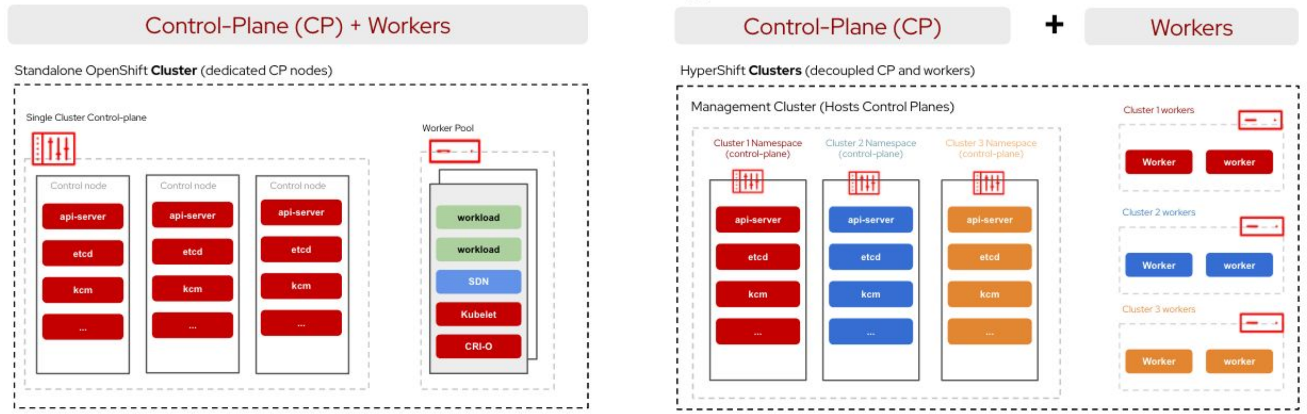
- + **Better isolation for security**
- + **Testing upgrades without impacting prod**
- + **Separation of workloads (Dev cluster heavy daily developer builds)**
- - **Control plane overhead (CP pro Cluster)**
- - **Maintenance & Administration overhead**

MultiCluster/MultiCloud with Hypershift

- **K8s in K8s**
  - Control Plain as pods
- **Decoupled Control Plane and Workload Clusters**
  - Deploy worker nodes across cloud providers
  - Upgrade Management Cluster und Workload Clusters separately
- **Centralized management**
  - Centralized Logging and monitoring
  - Centralized Image registry
  - …
- **Migrating existing Cluster to Hypershift hosted cluster will not be supported**
  - Workload migration
- **Still in Tech preview!**

MultiCluster/MultiCloud with Hypershift

Standalone OpenShift **HyperShift**



Control-Plane (CP) + Workers
Standalone OpenShift **Cluster** (dedicated CP nodes)
Single Cluster Control-plane
Control node
api-server
etcd
kcm
...
Worker Pool
workload
workload
SDN
Kubelet
CRI-O
Control-Plane (CP)
+
Workers
HyperShift **Clusters** (decoupled CP and workers)
Management Cluster (Hosts Control Planes)
Cluster 1 Namespace (control-plane)
Cluster 2 Namespace (control-plane)
Cluster 3 Namespace (control-plane)
api-server
etcd
kcm
...
Cluster 1 workers
Cluster 2 workers
Cluster 3 workers
Worker
worker

Low CAPEX and OPEX costs (bundling of CPs + CP as pods)

Central Management of CPs (easy operation & maintenance)

Multi-arch support (e.g. CP x86, workers ARM)

Network & Trust segmentation

Mixed Iaas For CP and Workers

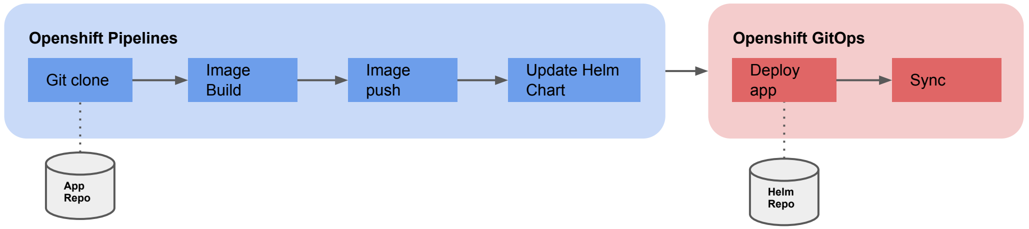
Fast cluster bootstrapping (CP as Pods)

Workload migration from 3.x to 4.x

- **Two different types of workloads**
  - Stateless workloads
  - Stateful workloads
- **Stateless workloads with Helm**
  - Convert existing manifest into Helm charts
    - Use https://github.com/google/shifter to convert Openshift manifests into K8s conform manifests
    - A.k.
      - Templates -> Deployment, Helm Chart
      - DeploymentConfigs -> Deployment
      - …
  - Write Helm chart templates using the K8s manifests
- **Stateful workloads using Redhat Migration Toolkit for containers**
  - Migrate the PVCs using Direct Volume Migration
  - Migrate the manifests similar to the stateless workloads section with Helm
  - Make sure to attach the PVCs to the corresponding Stateful sets/Deployments
- **Why Helm?**
  - Versioning
  - Reusability of charts across multiple environments
  - Release Rollbacks
- **Blue / Green deployments to eliminate downtime during migration**

GitOps for developers

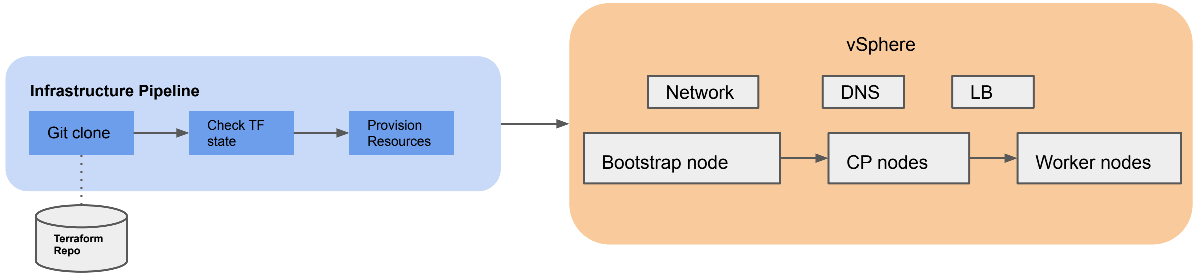
- **Two different repositories for application and Helm charts**
- **Red Hat OpenShift Pipelines (Tekton) for CI**
    - Push based
    - Code testing
    - Build the dockerfiles and push the images to the registry
    - Update the Helm charts with the new image tag
- **Red Hat OpenShift GitOps (ArgoCD) for CD**
    - Pull based
    - Sync the state of the Helm chart to the corresponding K8s cluster



GitOps for SREs

Chicken or the egg

- **Red Hat OpenShift Pipelines & Red Hat OpenShift Gitops**
  - Can be used similarly as the developers for infrastructure applications on OCP
- **Infrastructure Pipeline**
  - Red Hat OpenShift Pipelines is a part of Red Hat OCP to provision the initial infrastructure on vSphere another pipeline is needed
- **Terraform**
  - Redhat provides Terraform modules to provision OCP 4 infrastructure on vSphere using openshift-installer



Securing Applications

- **Securing the applications deployed is also about securing the supply chain**
  - Not all applications are built in-house
- **SBOM**
  - Identify and track all third-party components, in particular open source components
- **Redhat Advanced Cluster Security (Stackrox)**
  - Integrates with every stage of container lifecycle: build, deploy and runtime
    - Build: Fails the CI builds when images matches the condition of the policy
    - Deploy: Blocks creation of deployments that match the conscious of the policy
    - Runtime: Kills pods that match the conditions of the policy
  - Manages network policies
    - secure access to and from applications
  - Vulnerability management
    - Scan Docker image layers for vulnerabilities
- **Open Policy Agent**
  - Enforce policies utilizing K8s validating and mutating admission controllers
  - E.g.
    - Prohibit {insecure registries, insecure capabilities}
    - Enforce {labeling, network policies}