



Red Hat OpenShift

Installation

Installation Paradigms

OPENSIFT CONTAINER PLATFORM

Full Stack Automated (IPI)

Simplified opinionated “Best Practices” for cluster provisioning

Fully automated installation and updates including host container OS.



Pre-existing Infrastructure (UPI)

Customer managed resources & infrastructure provisioning

Plug into existing DNS and security boundaries



HOSTED OPENSIFT

Red Hat OpenShift on IBM Cloud *

Deploy directly from the IBM Cloud console. An IBM service, master nodes are managed by IBM Cloud engineers.

Azure Red Hat OpenShift **

Deploy directly from the Azure console. A MSFT service, jointly managed by Red Hat and Microsoft

OpenShift Dedicated **

Get a powerful cluster, fully managed by Red Hat engineers and support; a Red Hat service.

* Based on OCP v4.3 GA slated for March; public beta available now

** Entitlements of OCP obtained through a Cloud Pak purchase are not transferable to these environments

cloud.redhat.com

[Clusters](#) > [Create](#)

Create an OpenShift cluster





Cloud

Datacenter

Local

Managed services

Create clusters in the cloud using a managed service.

	Offerings	Purchased through		Get started
>	 Red Hat OpenShift Dedicated	Red Hat	Available on AWS and GCP	Learn more
>	 Azure Red Hat OpenShift	Microsoft Azure	Flexible hourly billing	Try it on Azure
>	 Red Hat OpenShift on IBM Cloud	IBM	Flexible hourly billing	Try it on IBM
>	 Red Hat OpenShift Service on AWS	Amazon Web Services	Flexible hourly billing	Learn more

Run it yourself

Run OpenShift clusters on your own by installing from another cloud provider.

Cloud provider	Installation options
AWS	Full stack automation and pre-existing infrastructure
Azure	Full stack automation and pre-existing infrastructure

[Clusters](#) > [Create](#)

Create an OpenShift cluster

Cloud

Datacenter

Local

Red Hat CodeReady Containers

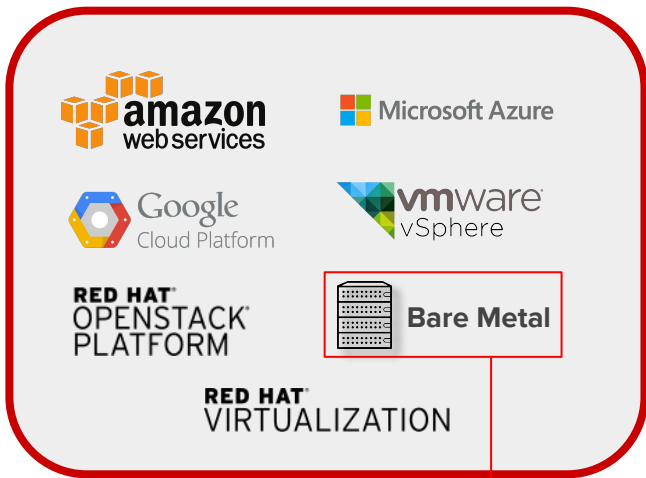
local sandbox

Create a minimal cluster on your desktop/laptop for local development and testing.

Note: Your CodeReady installation won't appear in the OpenShift Cluster Manager unless you enable cluster monitoring and telemetry.

4.6 Supported Providers

Full Stack Automation (IPI)



New addition in OCP 4.6

Pre-existing Infrastructure (UPI)



Now supports deploying to VMware vSphere 7.0



Dashboard

Quick start

Build

Explore IBM Cloud with this selection of easy starter tutorials and services.



Build a Virtual Cloud (VPC)

Create your own space in the IBM Cloud

7 min

Orchestration service


Select the [container platform](#) type and version for your cluster. For more information about versions, including links to the container platform community release notes, [see the docs](#).

 **OpenShift** 

4.5.24 

Infrastructure

Choose which network and compute environment to run your cluster on. [Learn more about the differences](#).

Classic 

Run your cluster with native subnet and VLAN networking on our classic infrastructure.

VPC

Create a fully customizable, software-defined virtual network with superior isolation using IBM Cloud VPC.

Location


Choose your location and configure your VLANs. [Learn more about this](#).

Resource group

Default 

Geography


Asia Pacific

Europe 

North America


South America


Availability

Single zone 

Multizone

Worker zone

Frankfurt 02 

 **No VLANs exist.**
VLANs will be created for you.

Summary

OpenShift cluster

Worker nodes €1.11/hr
b3c.4x16 - 4 vCPUs 16GB RAM

Total estimated cost €801.84/mo

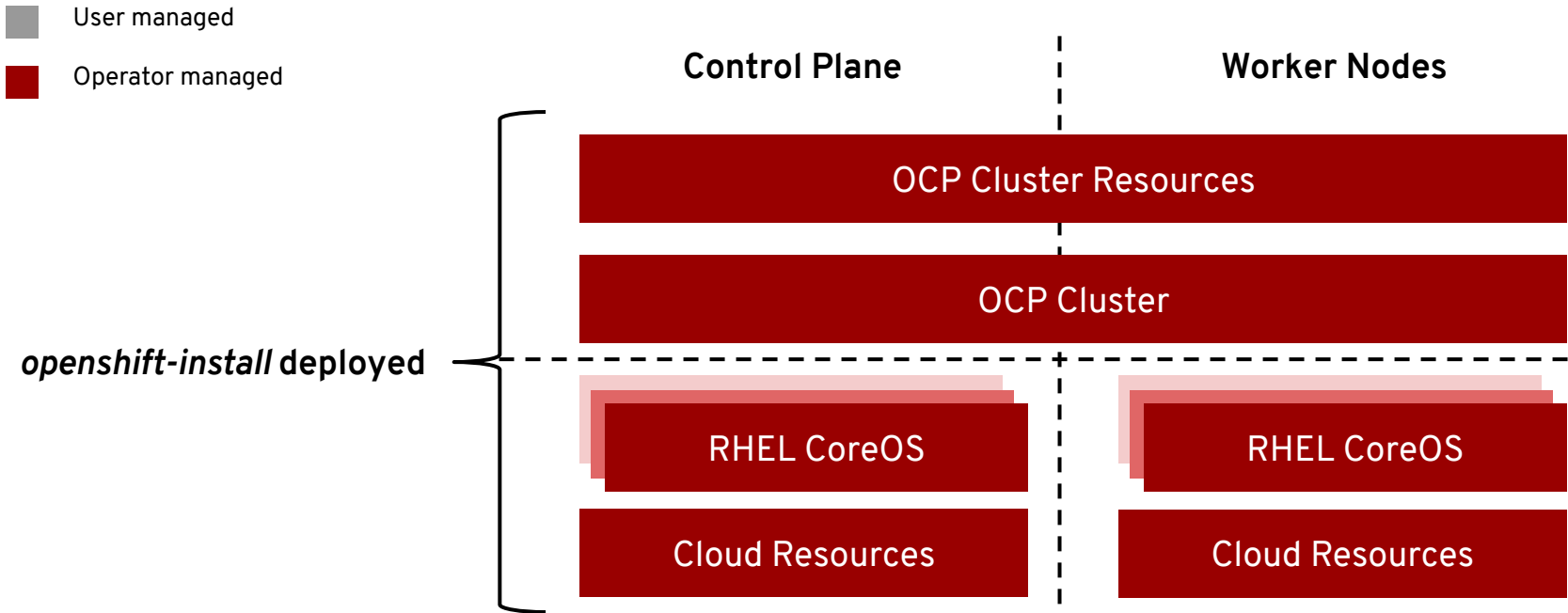
Additional charges for networking and bandwidth might apply.
Actual monthly total will vary with tiered pricing.

Create

Add to estimate



Full-stack Automated Installation (aka IPI)



OpenShift 4 installation

Installer and
user-provisioned
infrastructure,
bootstrap, and more

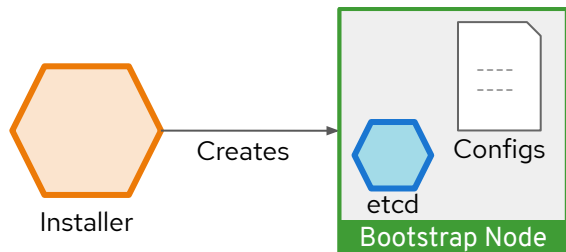
OpenShift Bootstrap Process: Self-Managed

How to boot a self-managed cluster:

Kubernetes

- OpenShift 4 is unique in that management extends all the way down to the operating system
- Every machine boots with a configuration that references resources hosted in the cluster it joins enabling cluster to manage itself
- Downside is that every machine looking to join the cluster is waiting on the cluster to be created
- Dependency loop is broken using a bootstrap machine, which acts as a temporary control plane whose sole purpose is bringing up the permanent control plane nodes
- Permanent control plane nodes get booted and join the cluster leveraging the control plane on the bootstrap machine
- Once the pivot to the permanent control plane takes place, the remaining worker nodes can be booted and join the cluster

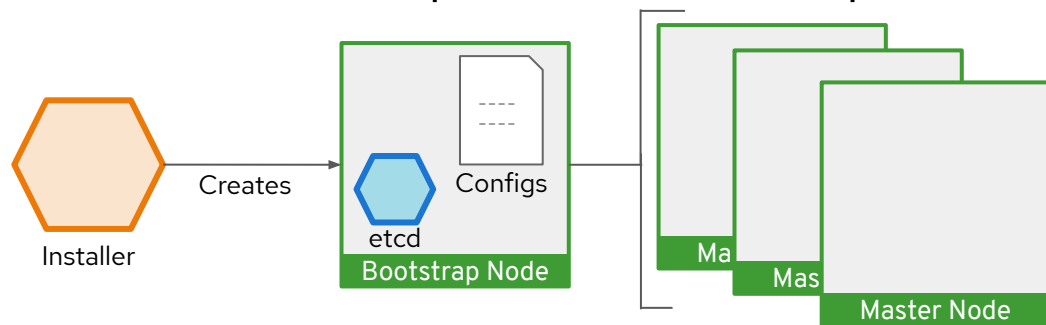
OpenShift Bootstrap Process: Step by Step



Bootstrapping process step by step:

1. Bootstrap machine boots and starts hosting the remote resources required for master machines to boot. Runs one instance of etcd

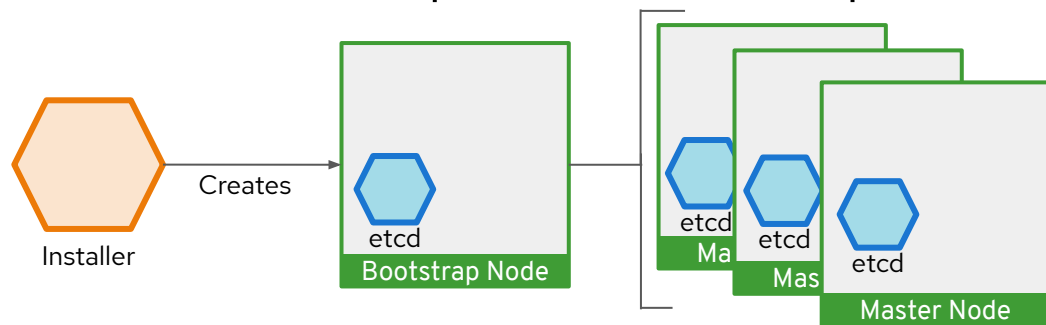
OpenShift Bootstrap Process: Step by Step



Bootstrapping process step by step:

1. Bootstrap machine boots and starts hosting the remote resources required for master machines to boot. Runs one instance of etcd
2. Master machines fetch the remote resources from the bootstrap machine and finish booting.

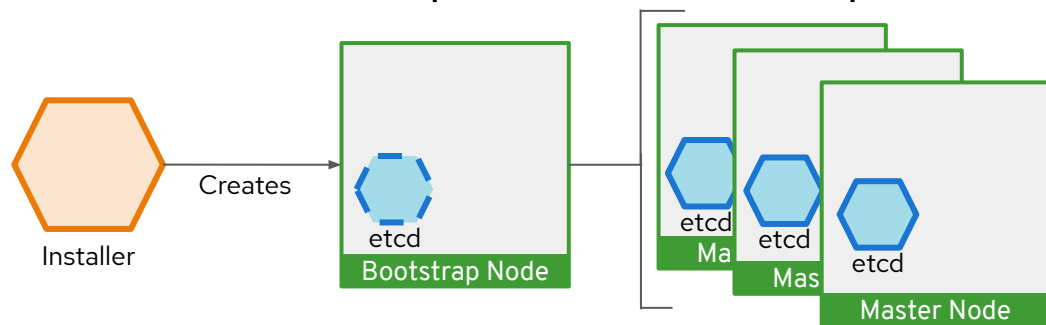
OpenShift Bootstrap Process: Step by Step



Bootstrapping process step by step:

1. Bootstrap machine boots and starts hosting the remote resources required for master machines to boot. Runs one instance of etcd
2. Master machines fetch the remote resources from the bootstrap machine and finish booting.
3. Master machines use the bootstrap node to scale the etcd cluster to 4 total instances.

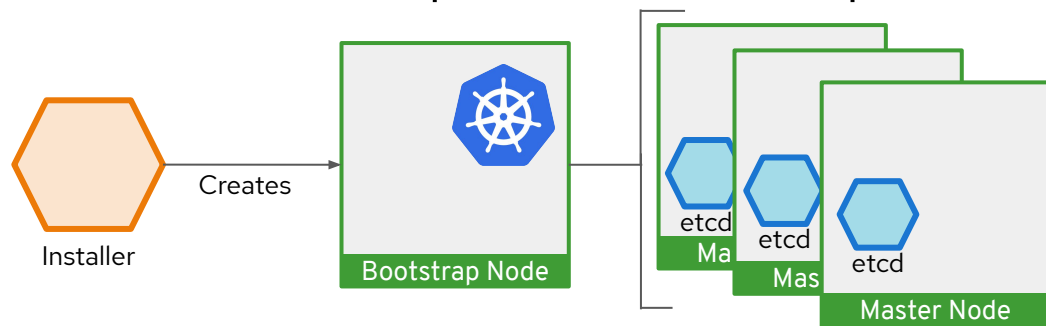
OpenShift Bootstrap Process: Step by Step



Bootstrapping process step by step:

1. Bootstrap machine boots and starts hosting the remote resources required for master machines to boot. Runs one instance of etcd
2. Master machines fetch the remote resources from the bootstrap machine and finish booting.
3. Master machines use the bootstrap node to scale the etcd cluster to 4 total instances.
4. The Etcd operator scales itself down off the bootstrap node, leaving the etcd instance count to 3

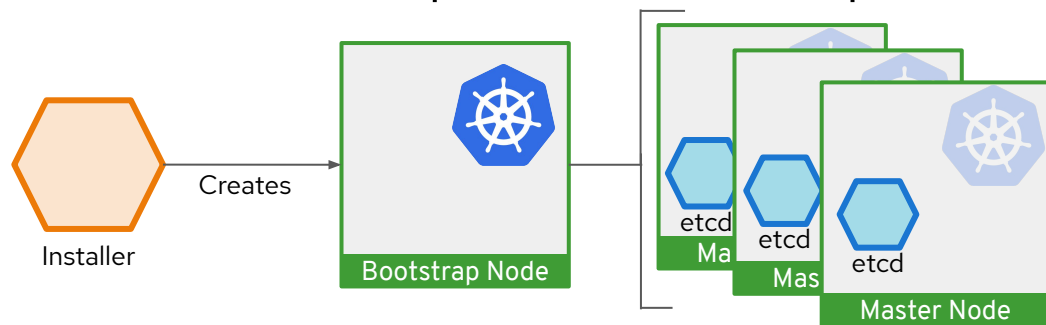
OpenShift Bootstrap Process: Step by Step



Bootstrapping process step by step:

1. Bootstrap machine boots and starts hosting the remote resources required for master machines to boot. Runs one instance of etcd
2. Master machines fetch the remote resources from the bootstrap machine and finish booting.
3. Master machines use the bootstrap node to scale the etcd cluster to 3 instances.
4. The Etcd operator scales itself down off the bootstrap node, then scales back up to 3; all on the Masters
5. Bootstrap node starts a temporary Kubernetes control plane using the newly-created etcd cluster.

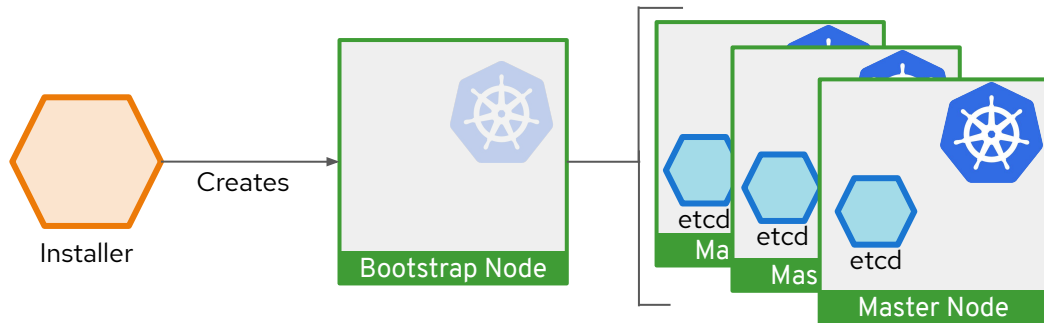
OpenShift Bootstrap Process: Step by Step



Bootstrapping process step by step:

1. Bootstrap machine boots and starts hosting the remote resources required for master machines to boot. Runs one instance of etcd
2. Master machines fetch the remote resources from the bootstrap machine and finish booting.
3. Master machines use the bootstrap node to scale the etcd cluster to 3 instances.
4. The Etcd operator scales itself down off the bootstrap node, then scales back up to 3; all on the Masters
5. Bootstrap node starts a temporary Kubernetes control plane using the newly-created etcd cluster.
6. Temporary control plane schedules the production control plane to the master machines.

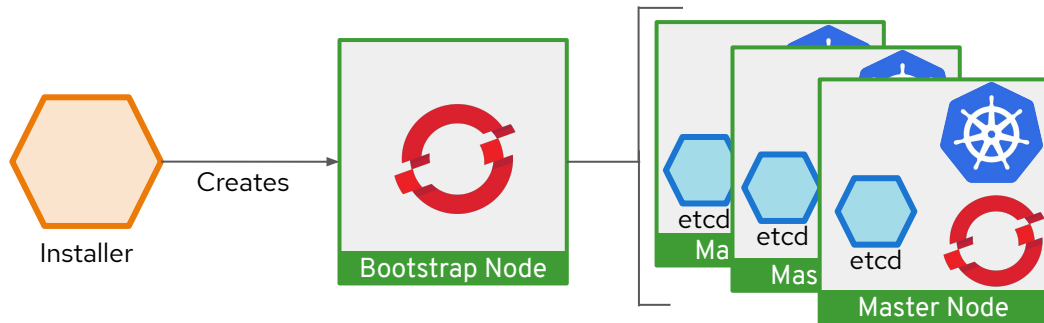
OpenShift Bootstrap Process: Step by Step



Bootstrapping process step by step:

1. Bootstrap machine boots and starts hosting the remote resources required for master machines to boot. Runs one instance of etcd
2. Master machines fetch the remote resources from the bootstrap machine and finish booting.
3. Master machines use the bootstrap node to scale the etcd cluster to 3 instances.
4. The Etcd operator scales itself down off the bootstrap node, then scales back up to 3; all on the Masters
5. Bootstrap node starts a temporary Kubernetes control plane using the newly-created etcd cluster.
6. Temporary control plane schedules the production control plane to the master machines.
7. Temporary control plane shuts down, yielding to the production control plane.

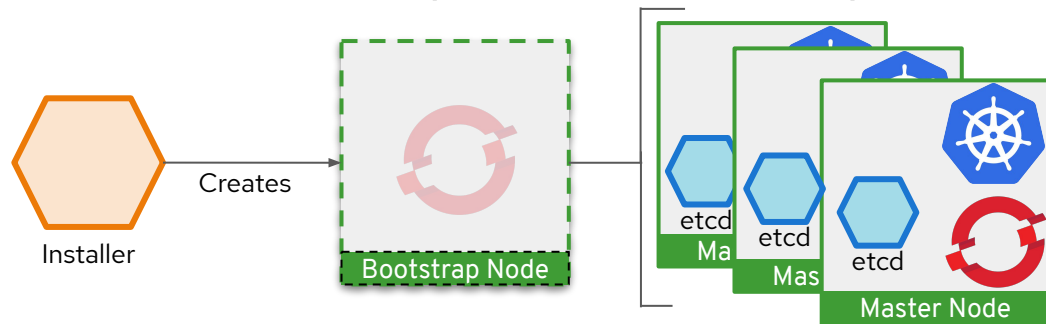
OpenShift Bootstrap Process: Step by Step



Bootstrapping process step by step:

1. Bootstrap machine boots and starts hosting the remote resources required for master machines to boot. Runs one instance of etcd
2. Master machines fetch the remote resources from the bootstrap machine and finish booting.
3. Master machines use the bootstrap node to scale the etcd cluster to 3 instances.
4. The Etcd operator scales itself down off the bootstrap node, then scales back up to 3; all on the Masters
5. Bootstrap node starts a temporary Kubernetes control plane using the newly-created etcd cluster.
6. Temporary control plane schedules the production control plane to the master machines.
7. Temporary control plane shuts down, yielding to the production control plane.
8. Bootstrap node injects OpenShift-specific components into the newly formed control plane.

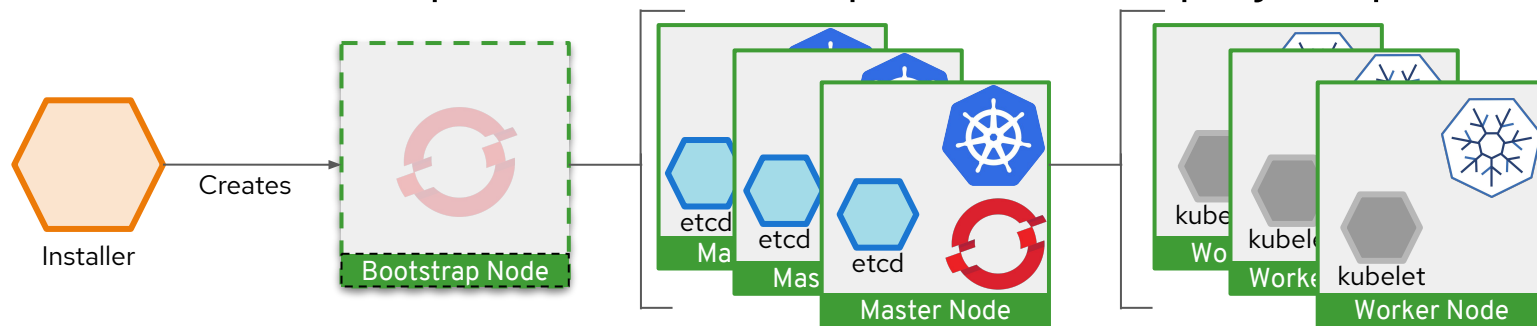
OpenShift Bootstrap Process: Step by Step



Bootstrapping process step by step:

1. Bootstrap machine boots and starts hosting the remote resources required for master machines to boot. Runs one instance of etcd
2. Master machines fetch the remote resources from the bootstrap machine and finish booting.
3. Master machines use the bootstrap node to scale the etcd cluster to 3 instances.
4. The Etcd operator scales itself down off the bootstrap node, then scales back up to 3; all on the Masters
5. Bootstrap node starts a temporary Kubernetes control plane using the newly-created etcd cluster.
6. Temporary control plane schedules the production control plane to the master machines.
7. Temporary control plane shuts down, yielding to the production control plane.
8. Bootstrap node injects OpenShift-specific components into the newly formed control plane.
9. Installer then tears down the bootstrap node or if user-provisioned, this needs to be performed by the administrator.

OpenShift Bootstrap Process: Step by Step



Bootstrapping process step by step:

1. Bootstrap machine boots and starts hosting the remote resources required for master machines to boot. Runs one instance of etcd
2. Master machines fetch the remote resources from the bootstrap machine and finish booting.
3. Master machines use the bootstrap node to scale the etcd cluster to 3 instances.
4. The Etcd operator scales itself down off the bootstrap node, then scales back up to 3; all on the Masters
5. Bootstrap node starts a temporary Kubernetes control plane using the newly-created etcd cluster.
6. Temporary control plane schedules the production control plane to the master machines.
7. Temporary control plane shuts down, yielding to the production control plane.
8. Bootstrap node injects OpenShift-specific components into the newly formed control plane.
9. Installer then tears down the bootstrap node or if user-provisioned, this needs to be performed by the administrator.
10. Worker machines fetch remote resources from masters and finish booting.

How everything deployed comes under management

Masters (Special)

- Full Stack Automated: Installer provisions minimal viable masters
- User Provisioned: User/Administrator provisions minimal viable masters
- Machine API adopts existing masters post-provision
- Each master is a standalone Machine object
- Termination protection (avoid self-destruction)

Workers

- Each Machine Pool corresponds to MachineSet
- Optionally autoscale (min,max) and health check (replace if not ready > X minutes)

Multi-AZ

- MachineSets scoped to single AZ
- Installer stripes N machine sets across AZs by default
- Post-install best effort balance via cluster autoscaler

One Touch provisioning via Ignition

Machine generated; Machine validated

Ignition applies a declarative node configuration early in the boot process. Unifies kickstart and cloud-init.

- Generated via openshift-install
- Configures storage, systemd units, users, & remote configs
- Executed in the initramfs
- Configuration for masters & workers is served from the control plane and sourced from Machine Configs

```
{
  "ignition": {
    "config": {},
    "timeouts": {},
    "version": "2.1.0"
  },
  "passwd": {
    "users": [
      {
        "name": "core",
        "passwordHash": "$6$43y3tkl...",
        "sshAuthorizedKeys": [
          "key1"
        ]
      }
    ]
  },
  "storage": {},
  "systemd": {}
}
```

Full Stack Automated Deployments

Simplified Cluster Creation

Designed to easily provision a “best practices” OpenShift cluster

- New CLI-based installer with interactive guided workflow that allows for customization at each step
- Installer takes care of provisioning the underlying Infrastructure significantly reducing deployment complexity
- Leverages RHEL CoreOS for all node types enabling full stack automation of installation and updates of both platform and host OS content

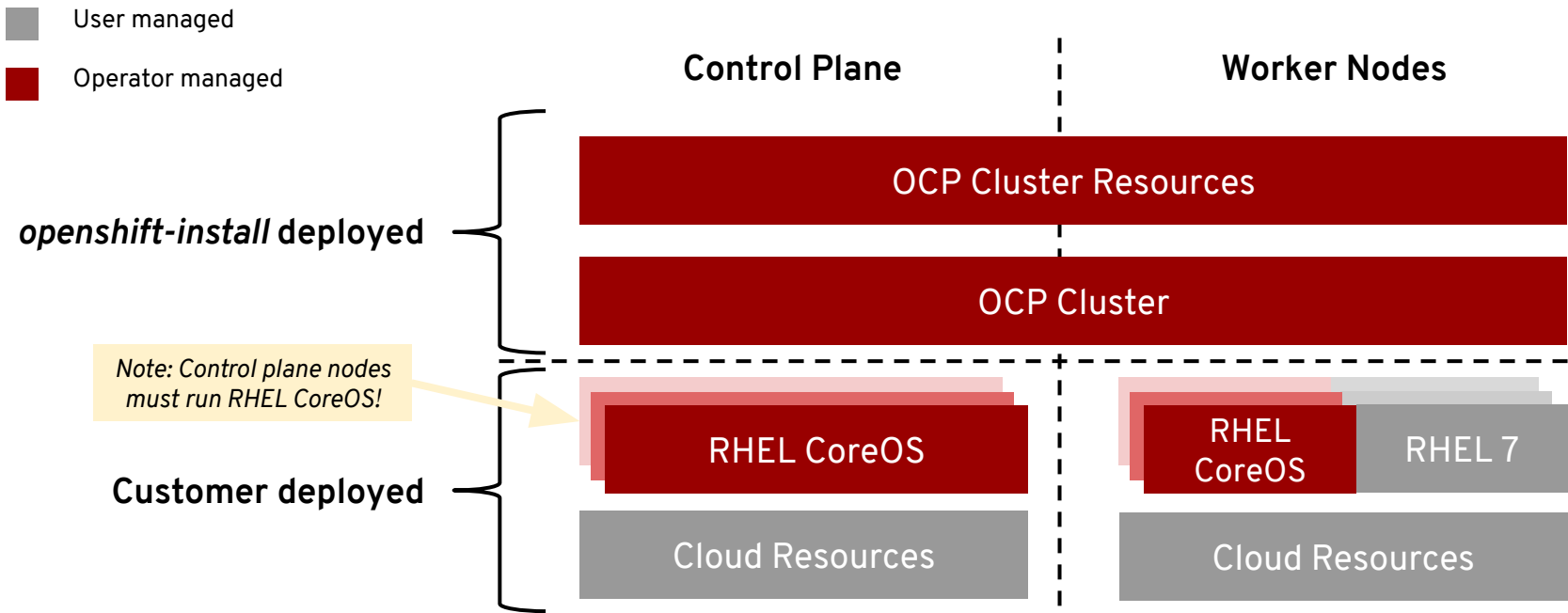
Faster Install

The installer typically finishes within 30 minutes

- Only minimal user input needed with all non-essential install config options now handled by component operator CRD's
- [See the OpenShift documentation for more details](#)

```
$ ./openshift-install --dir ./demo create cluster
? SSH Public Key /Users/demo/.ssh/id_rsa.pub
? Platform aws
? Region us-west-2
? Base Domain example.com
? Cluster Name demo
? Pull Secret [? for help]
*****
INFO Creating cluster...
INFO Waiting up to 30m0s for the Kubernetes API...
INFO API v1.11.0+c69f926354 up
INFO Waiting up to 30m0s for the bootstrap-complete event...
INFO Destroying the bootstrap resources...
INFO Waiting up to 10m0s for the openshift-console route to be created...
INFO Install complete!
INFO Run 'export KUBECONFIG=<your working directory>/auth/kubeconfig' to
manage the cluster with 'oc', the OpenShift CLI.
INFO The cluster is ready when 'oc login -u kubeadmin -p <provided>'
succeeds (wait a few minutes).
INFO Access the OpenShift web-console here:
https://console-openshift-console.apps.demo.example.com
INFO Login to the console with user: kubeadmin, password: <provided>
```

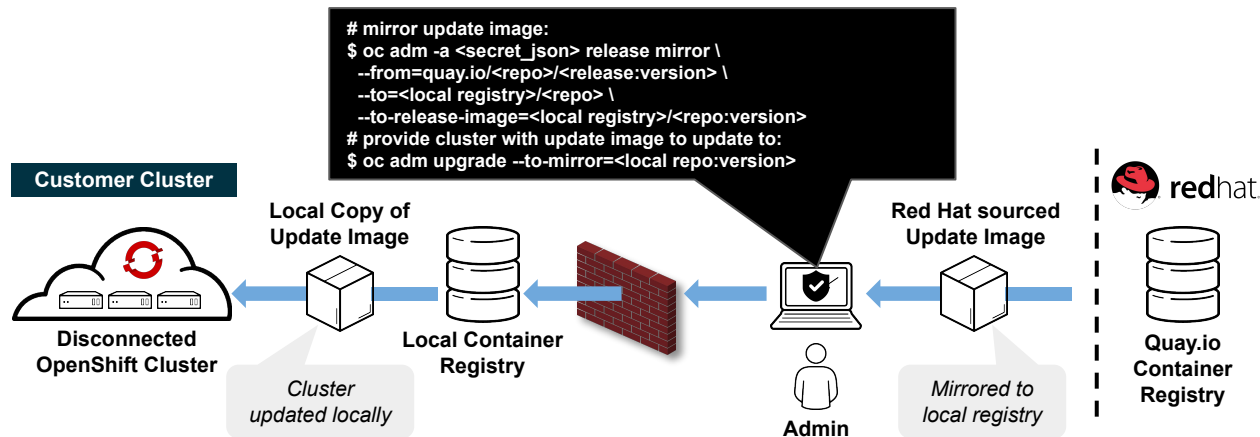
Pre-existing Infrastructure Installation (aka UPI)



Comparison of Paradigms

	Full Stack Automation	Pre-existing Infrastructure
Build Network	Installer	User
Setup Load Balancers	Installer	User
Configure DNS	Installer	User
Hardware/VM Provisioning	Installer	User
OS Installation	Installer	User
Generate Ignition Configs	Installer	Installer
OS Support	Installer: RHEL CoreOS	User: RHEL CoreOS + RHEL 7
Node Provisioning / Autoscaling	Yes	Only for providers with OpenShift Machine API support

Disconnected “Air-gapped” Installation & Upgrading



Overview

- 4.2 introduces support for installing and updating OpenShift clusters in disconnected environments
- Requires local Docker 2.2 spec compliant container registry to host OpenShift content
- Designed to work with the user provisioned infrastructure deployment method
 - *Note: Will not work with Installer provisioned infrastructure deployments*

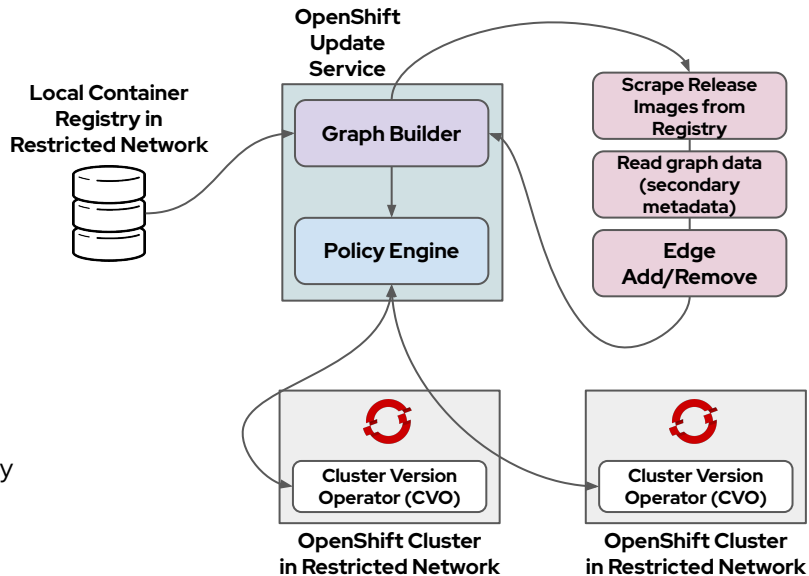
Installation Procedure

- Mirror OpenShift content to local container registry in the disconnected environment
- Generate install-config.yaml: `./openshift-install create install-config --dir <dir>`
 - Edit and add pull secret (PullSecret), CA certificate (AdditionalTrustBundle), and image content sources (ImageContentSources) to install-config.yaml
- Set the `OPENSHIFT_INSTALL_RELEASE_IMAGE_OVERRIDE` environment variable during the creation of the ignition configs
- Generate the ignition configuration: `./openshift-install create ignition-configs --dir <dir>`
- Use the resulting ignition files to bootstrap the cluster deployment

OpenShift Update Service

Update manager for your clusters in restricted or disconnected networks

- OpenShift Update Service (OSUS) is the on-premise release of Red Hat's hosted update service
- Supports the publishing of upgrade graph information to clusters in restricted networks
- Provides clusters with a list of next recommended update versions based on the current version installed on the cluster
- Comprised of two services:
 - **Graph Builder:** Fetches OpenShift release payload information (primary metadata) from any container registry (compatible with [Docker registry V2 API](#)) and builds a [directed acyclic graph](#) (DAG) representing valid upgrade edges
 - **Policy Engine:** Responsible for selectively serving updates to every cluster by altering a client's view of the graph with a set of filters
- GA release planned for post-4.6 and will be distributed on Operator Hub as an optional add-on operator
- [Blog post announcing OpenShift Update Service](#)





linkedin.com/company/red-hat



youtube.com/user/RedHatVideos



facebook.com/redhatinc



twitter.com/RedHat