

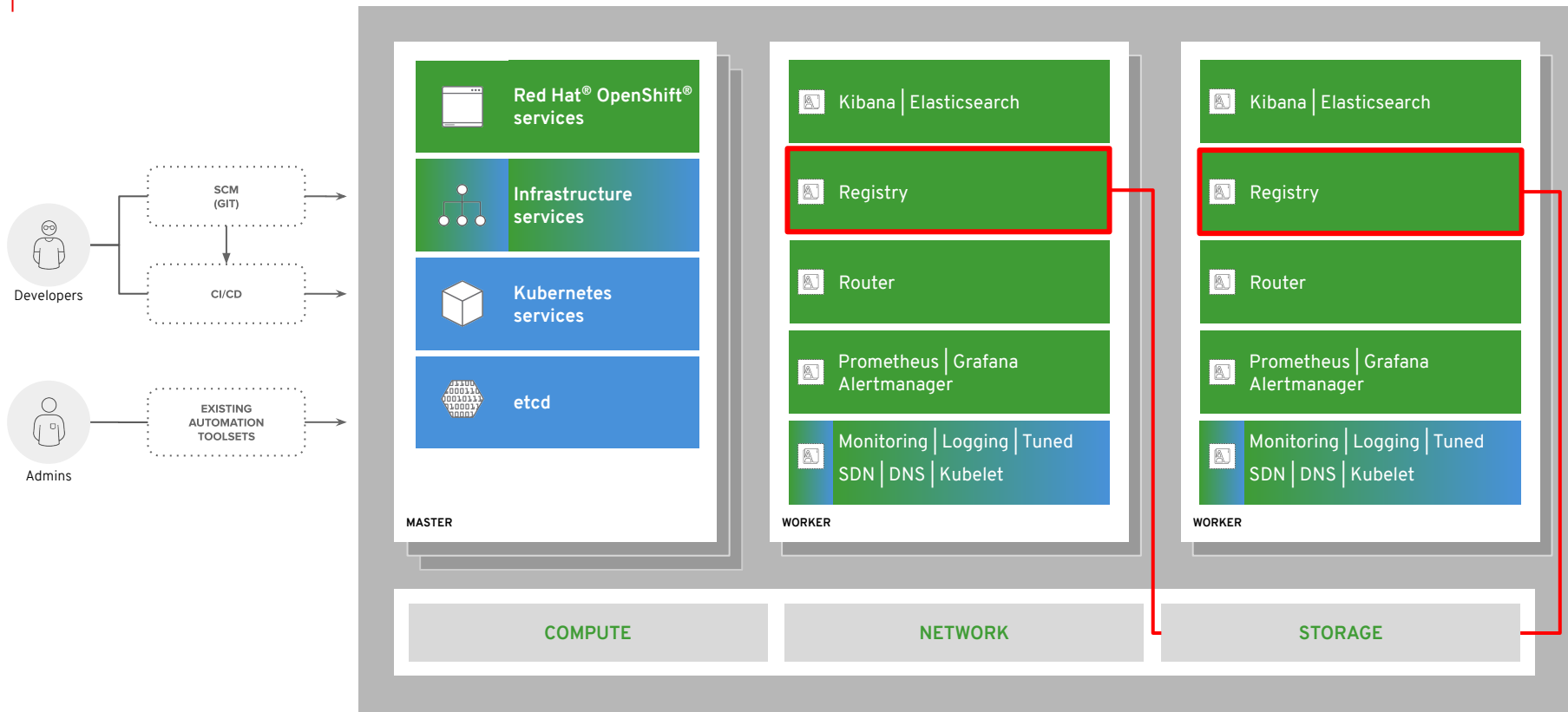


Red Hat OpenShift 4 - Installation

Robert Bohne

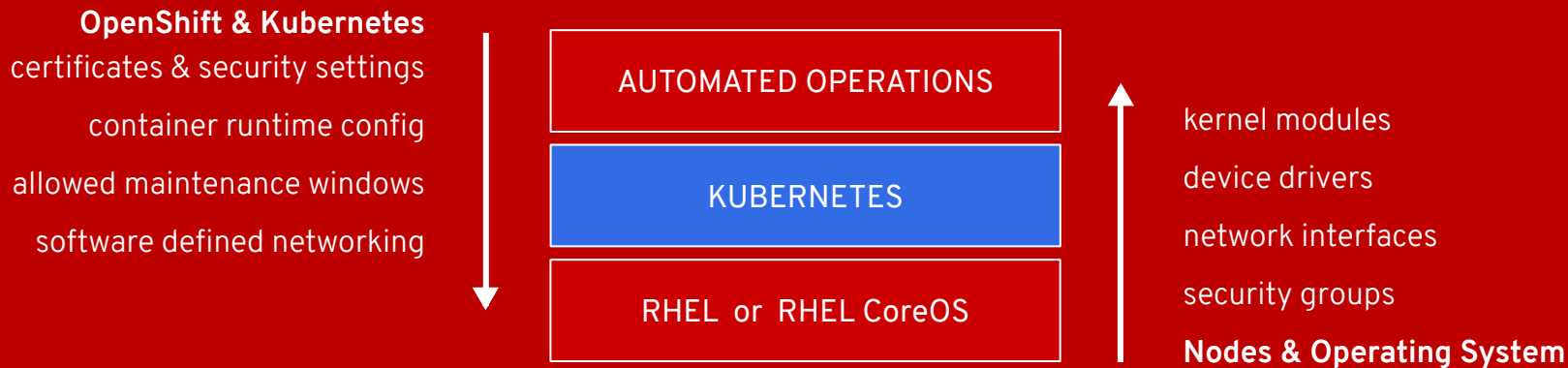
SR. SPECIALIST SOLUTION ARCHITECT | OPENSIFT

Twitter: @RobertBohne



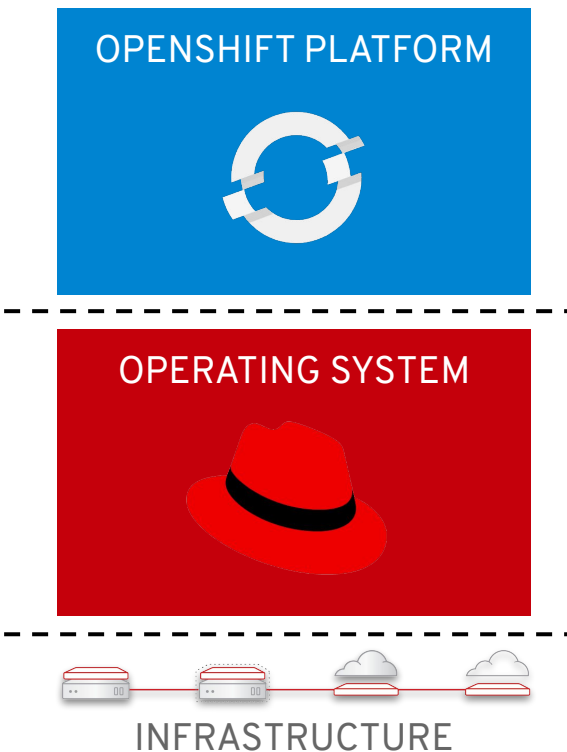
The New Platform Boundary

OpenShift 4 is aware of the entire infrastructure and brings the Operating System under management



Full-stack automated install

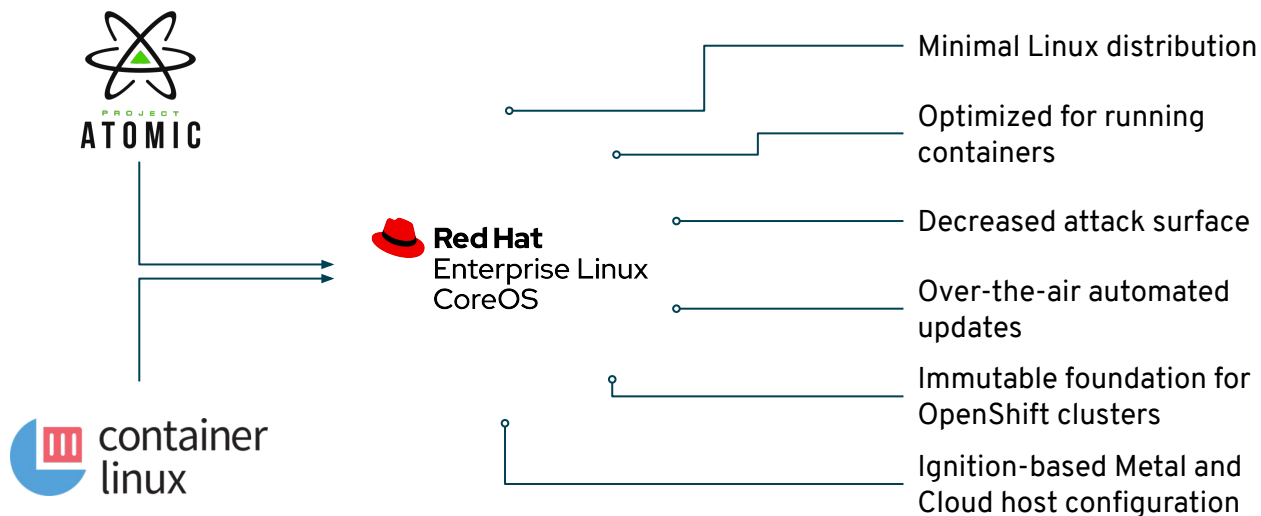
OPENSIFT 3 & 4



OPENSIFT 4 (only)



Red Hat Enterprise Linux CoreOS



Immutable Operating System

Red Hat Enterprise Linux CoreOS is versioned with OpenShift

CoreOS is tested and shipped in conjunction with the platform. Red Hat runs thousands of tests against these configurations

Red Hat Enterprise Linux CoreOS is managed by the Operator

The Operating system is operated as part of the OpenShift platform. The Operator manages the config for components managed by M

Operator:

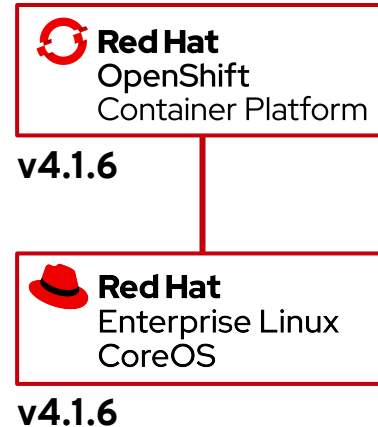
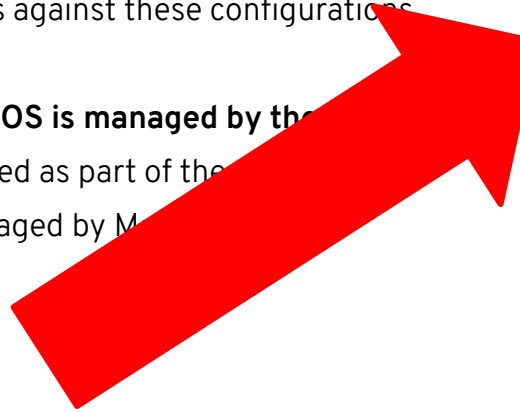
- CRI-O config
- Kubelet config
- Authorized registries
- SSH config

RHEL CoreOS admins are responsible for:

Nothing. 😊🙌

All is installed via Operators and Container Images!

There are no RPM's anymore!





cri-o

A lightweight, OCI-compliant container runtime

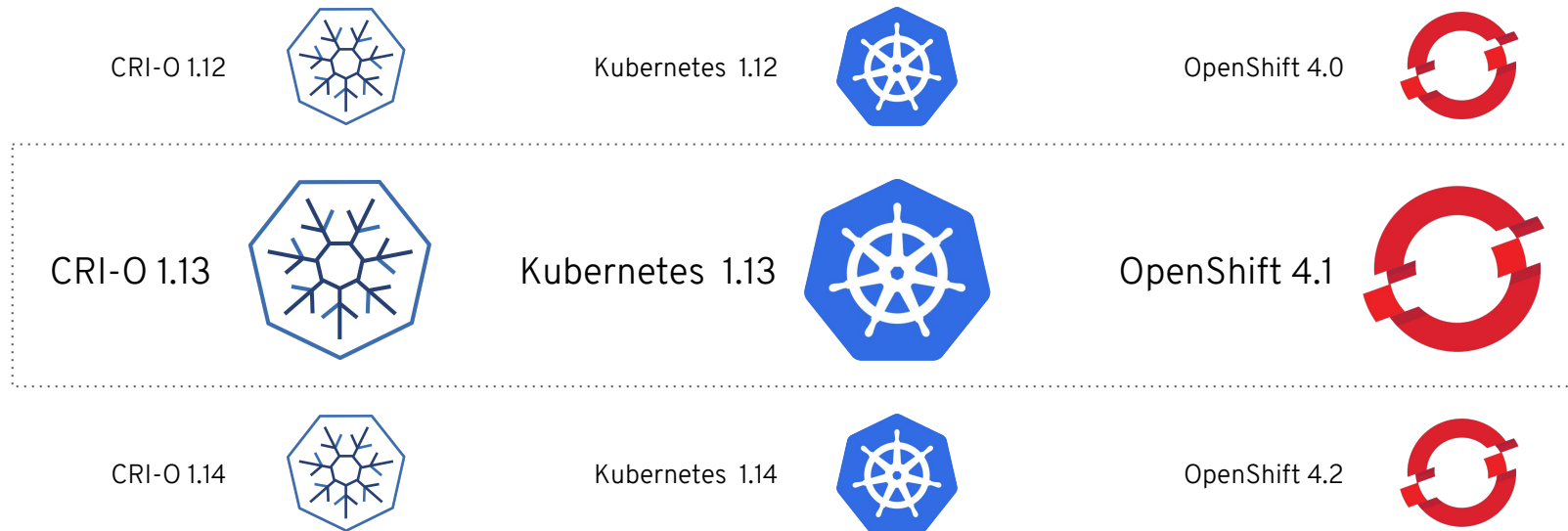
Minimal and Secure
Architecture

Optimized for
Kubernetes

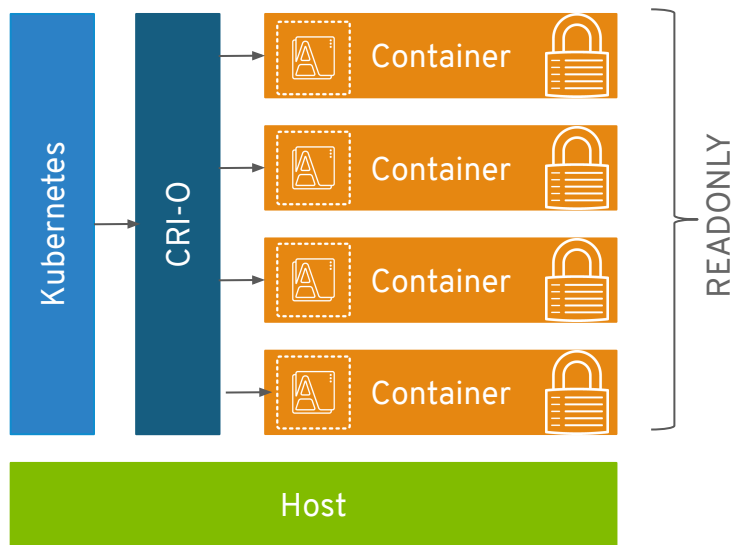
Runs any
OCI-compliant image
(including docker)

CRI-O Support in OpenShift

CRI-O tracks and versions identical to Kubernetes, simplifying support permutations



OCI AND CRI-O



- A Kubernetes thing
- Now part of CNCF! (April 8th)
- OCI daemon
- Implements Kubelet Container Runtime Interface (CRI)



SECURITY FEATURES

Run securely in a production cluster
No daemon
Read-only containers
Enable fewer capabilities
User namespaces
FIPS mode support



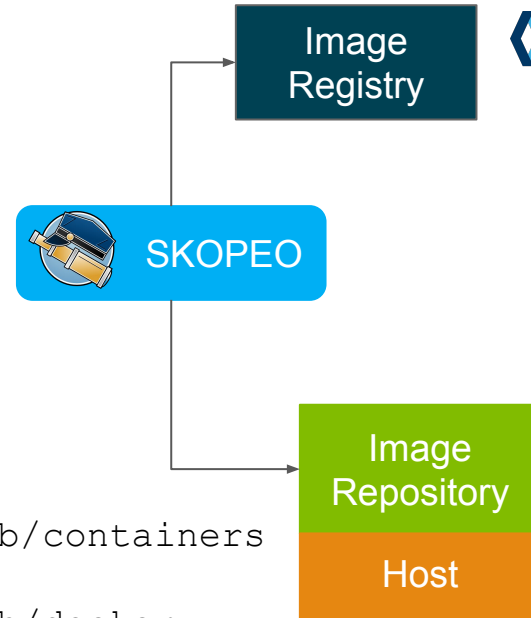



IMAGE COPY WITH SKOPEO

- Built for interfacing with Docker registry
- CLI for images and image registries
- Rejected by upstream Docker `_(ツ)_/`
- Allows remote inspection of image meta-data - no downloading
- Can copy from one storage to another

SECURITY FEATURES

- Share securely
- No daemon
- Inspect remote images
- No pulling potentially malicious images
- Non-root copy. Bridge between registries.



`/var/lib/containers`
or
`/var/lib/docker`

The new container CLI

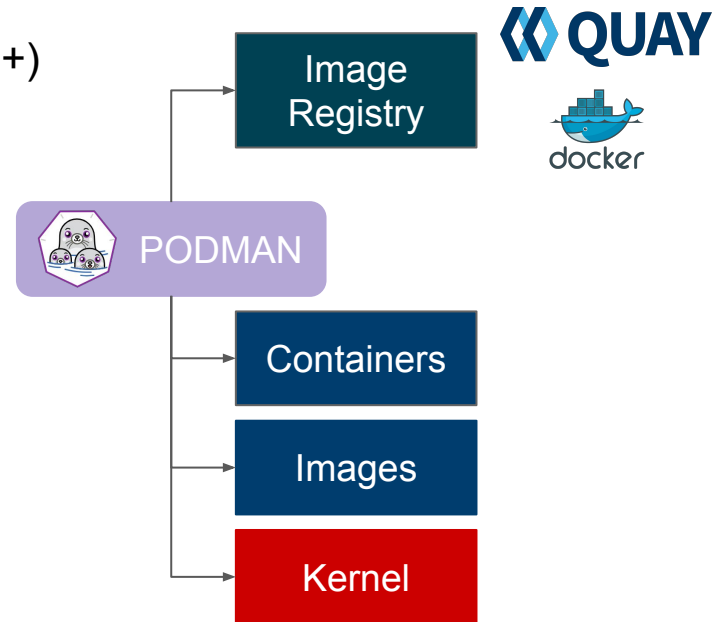


podman

- @ podman.io
- Client only tool, based on the Docker CLI. (same+)
- No daemon!
- Storage for
 - Images - `containers/image`
 - Containers - `containers/storage`
- Runtime - `runc`
- Shares state with CRI-O and with Buildah!

SECURITY FEATURES

Run and develop securely
No daemon
Run without root
Isolate with user namespaces
Audit who runs what

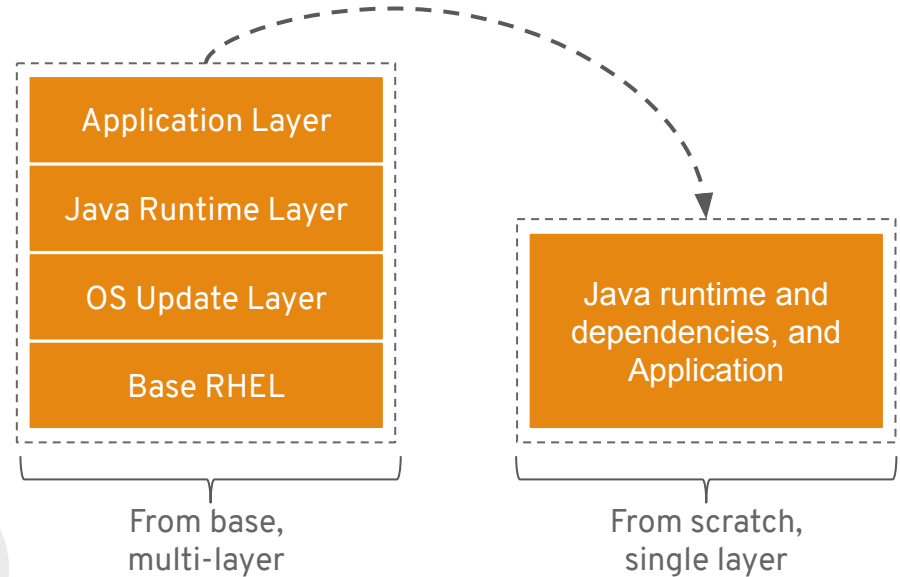


Why use Buildah?



buildah

- Now buildah.io
- Builds OCI compliant images
- No daemon - no “docker socket”
- Does not require a running container
- Can use the host’s user’s secrets.
- Single layer, from scratch images are made easy and it ensures limited manifest.
- If needed you can still maintain Dockerfile based workflow



SECURITY FEATURES

- Build securely
- No daemon
- Shrink the attack surface
- Fine-grained control of the layers
- Run builds isolated
- Better secret management





OPEN CONTAINER
INITIATIVE

- Docker, Red Hat et al. June 2015
- Two specifications
 - Image format
 - How to package an OCI Image with sufficient information to launch the application on the target platform
 - Runtime
 - How to launch a “filesystem bundle” that is unpacked on disk
- Version 1.0 of each released July 19th 2017
- Distribution spec started in April, 2018.

One Touch provisioning via Ignition

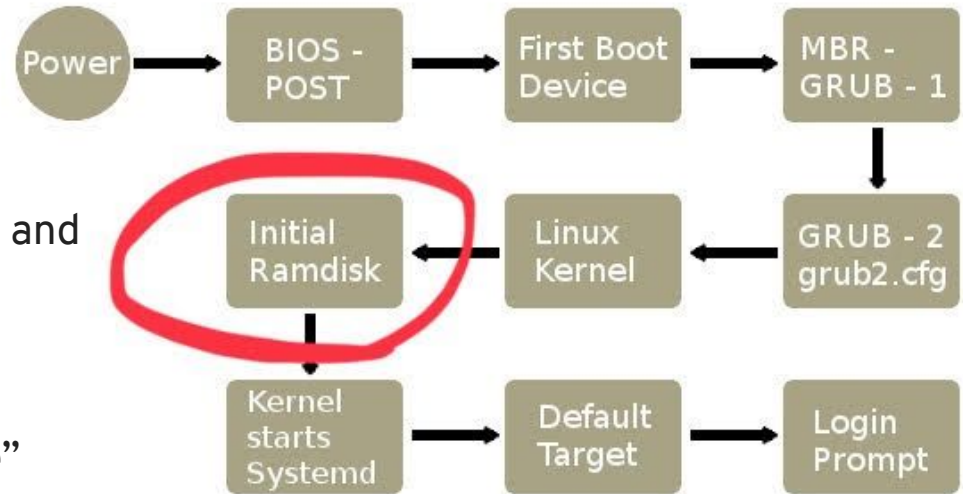
Machine generated; machine validated

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

- Generated via openshift-install & MCO
- Configures storage, systemd units, users, & remote configs
- Executed in the initramfs

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

How Ignition works



- [RHCSA Step 1](#)
 - Boot into “early userspace” and change a file (/etc/shadow)
- Ignition works the same way
 - Boots into “early userspace”
 - Change disks, files based on JSON
 - Starts the machine
- Based on standard [Linux startup process](#)

Network configuration

- **Default: DHCP!**
- Static IP via Kernel Arguments:

```
label linux
  menu label ^Install RHEL CoreOS
  kernel /images/vmlinuz
  append initrd=/images/initramfs.img nomodeset rd.neednet=1 coreos.inst=yes
ip=192.168.122.9::192.168.122.1:255.255.255.0:core2.example.com:enp1s0:none nameserver=192.168.122.1
coreos.inst.install_dev=sda coreos.inst.image_url=http://192.168.122.1/rhcos.raw.gz
coreos.inst.ignition_url=http://192.168.122.1/static.ign
```


Ignition Essentials

- First-boot only
 - Provisioning tool not a CM tool
 - Pass .ign
 - to the firmware via HTTP!
 - or cloud metadata svc (i.e. [AWS user-data](#))
- There's a great deep dive on Ignition [here](#)

If the first-boot cfg needs tweaks, re-provision the node from scratch!

Updating - Patching

Q: How to perform updating, patching?

A: Red Hat Enterprise Linux CoreOS, or RHCOS, is intended to be consumed as an embedded version of RHEL that is purpose built-to run OpenShift v4. RHCOS isn't managed or updated outside of OpenShift.

Updates are pushed to the platform and pick up OS security errata. If a customer is staying on top of OCP updates, then RHCOS will always be current. The management, and the fact that updates are not independent from the platform make RHCOS behave much closer to an appliance than a traditional OS.

Satellite

Q: Can we integrate with Satellite?

A: RHEL CoreOS does not include subscription-manager and **cannot be registered to Satellite like RHEL.**

Satellite can provide value in two major ways for RHEL CoreOS:

- ▶ Provisioning - Satellite is able to serve ignition configurations as well as serve the bits needed for a PXE install.
- ▶ Mirroring updates - Satellite's registry can be used to serve the containerized content in disconnected environments.

Installation Experiences

OPENSIFT CONTAINER PLATFORM

Full Stack Automated

Simplified opinionated “Best Practices” for cluster provisioning

Fully automated installation and updates including host container OS.



Pre-existing Infrastructure

Customer managed resources & infrastructure provisioning

Plug into existing DNS and security boundaries



HOSTED OPENSIFT

Azure Red Hat OpenShift

Deploy directly from the Azure console. Jointly managed by Red Hat and Microsoft Azure engineers.

OpenShift Dedicated

Get a powerful cluster, fully Managed by Red Hat engineers and support.

4.3 Supported Providers

Full Stack Automation (IPI)

amazon web services

Microsoft Azure

Google Cloud Platform

RED HAT OPENSTACK PLATFORM

Pre-existing Infrastructure (UPI)

amazon web services

Microsoft Azure*

Google Cloud Platform

vmware vSphere

IBM Z*

Bare Metal

* Support planned for an upcoming 4.3 z-stream release

Generally Available



What means



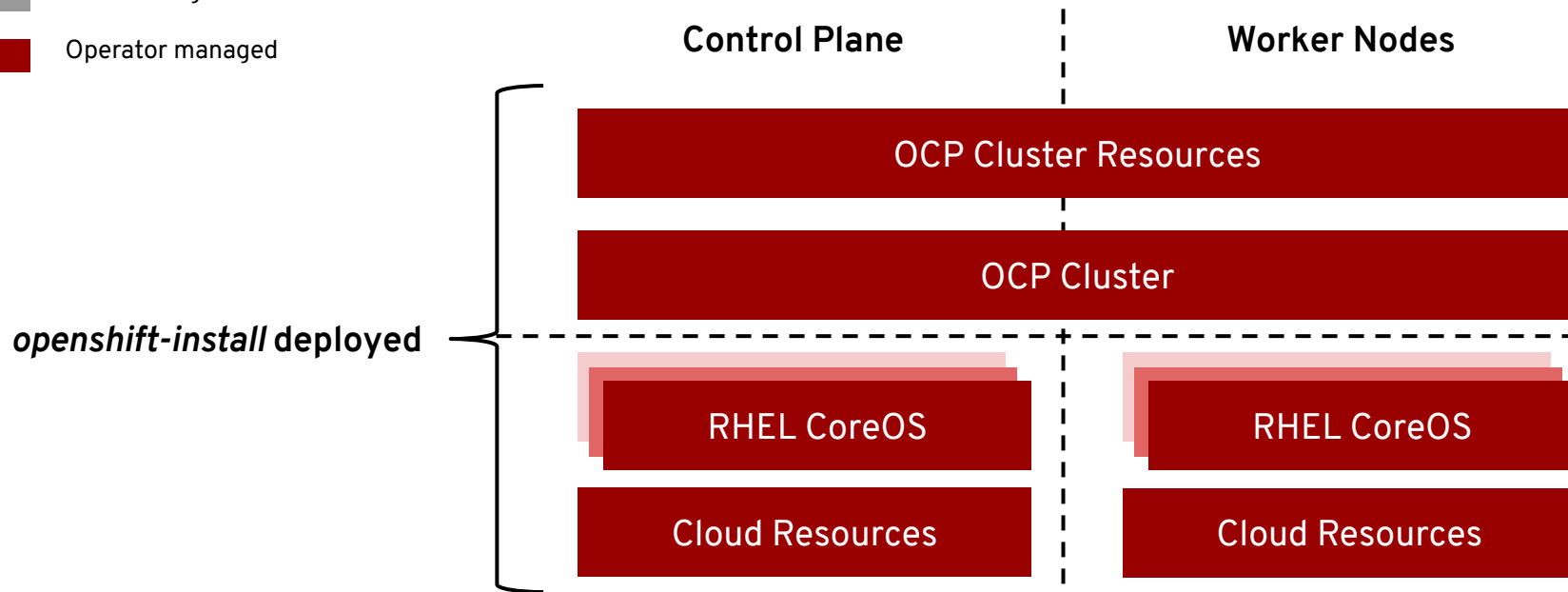
Bare Metal

- Installation on physical hardware ;-)
- Installation on Virtual Machines **WITHOUT** any guest tool / cloud integration!
[Deploying OpenShift 4.x on non-tested platforms using the bare metal install method](#)
For example Hyper-V: [Red Hat Enterprise Linux 8 Ecosystem](#)

Full Stack Automated Deployments

Day 1: openshift-install - Day 2: Operators

- User managed
- Operator managed



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

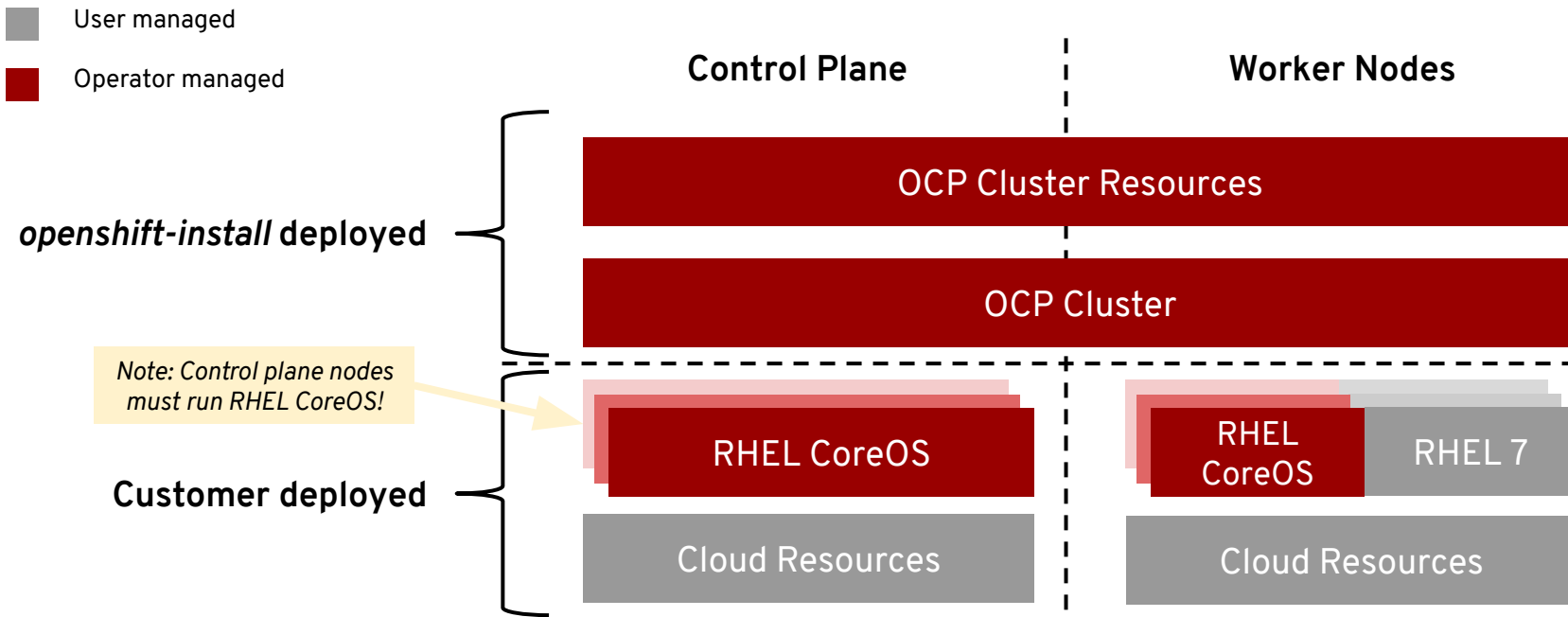
The installer typically finishes within 30 minutes

- Only minimal user input needed with all non-essential install config options now handled by component operator CRD’s
- 4.x provides support for AWS deployments with additional provider support planned in future releases
- [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>
```


Deploying to Pre-existing Infrastructure

Day 1: openshift-install - Day 2: Operators + Customer Managed Infra & Workers



Comparison between deployments methods

	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
Customization & Provider Support	Best Practices: AWS	Yes: AWS, Bare Metal, & VMware

Some dirty details:

Provider	Ignition stored in?	Load balancer?	Registry Storage?
IPI AWS, GCP, Azure	S3	Cloud Provider	S3
IPI OpenStack	Swift	Keepalived	Swift
UPI Bare Metal	Own Web Server	Own	Nothing! NFS?*
UPI VMware	Own Web Server	Own	Nothing! NFS?*

DNS

Nodes

- A records
- PTR records - at every reboot RH Core OS determination hostname via PTR record!

Cluster / Load Balancer

- *.apps.<CN>.<BD>. 86400 IN A <IP of load balancer>
- api.<CN>.<BD>. 86400 IN A <IP of load balancer>
- api-int.<CN>.<BD>. 86400 IN A <IP of load balancer>

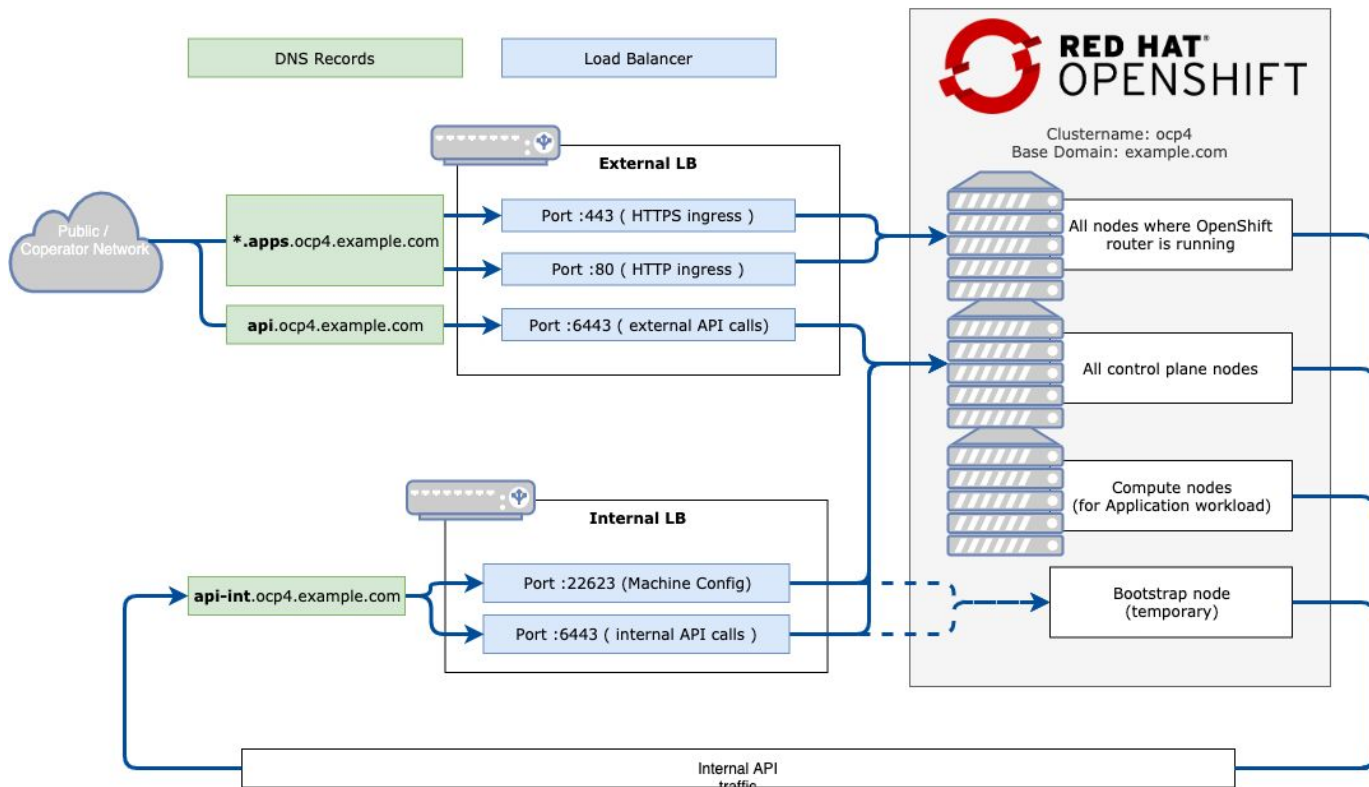
<CN> = Cluster Name
<BD> = Base Domain

Etcd Cluster

- SRV Records
 - # _service._proto.name. TTL class SRV priority weight port target.
 - _etcd-server-ssl._tcp.<CN>.<BD> 86400 IN SRV 0 10 2380 etcd-0.<CN>.<BD>.
 - _etcd-server-ssl._tcp.<CN>.<BD> 86400 IN SRV 0 10 2380 etcd-1.<CN>.<BD>.
 - _etcd-server-ssl._tcp.<CN>.<BD> 86400 IN SRV 0 10 2380 etcd-2.<CN>.<BD>.
- Additional A records to the Node/Host-names
 - etcd-0.<CN>.<BD>. 86400 IN A <IP of controleplan/master node 0>
 - etcd-1.<CN>.<BD>. 86400 IN A <IP of controleplan/master node 1>
 - etcd-2.<CN>.<BD>. 86400 IN A <IP of controleplan/master node 2>

Do NOT create PTR's!

OpenShift 4 DNS & Load Balancer Overview



Load balancer & DHCP



Bildquelle: Timo Klostermeier / pixelio.de

Deploy to pre-existing infrastructure for AWS, Bare Metal, GCP, & VMware!

Cust

Ena
pre

Pitfall

- Generates certificates that are only valid for 24 hours!
- To restart: delete the config dir “./demo” !!

sources and

structure
rdware/VMs

to

lets (such as

node ignition configs and kubeconfig) and aids with cluster bring-up by monitoring for bootstrap-complete and cluster-ready events

- Example native provider templates (AWS CloudFormation and Google Deployment Manager) included to help with user provisioning tasks for creating infrastructure objects
- While RHEL CoreOS is mandatory for the control plane, either RHEL CoreOS or RHEL 7 can be used for the worker/infra nodes

```
$ cat ./demo/install-config.yaml
apiVersion: v1
baseDomain: example.com
compute:
- name: worker
  replicas: 0
controlPlane:
  name: master
...

$ ./openshift-install --dir ./demo create ignition-config
INFO Consuming "Install Config" from target directory

$ ./openshift-install --dir ./demo wait-for bootstrap-complete
INFO Waiting up to 30m0s for the Kubernetes API at
https://api.demo.example.com:6443...
INFO API v1.11.0+c69f926354 up
INFO Waiting up to 30m0s for the bootstrap-complete event...
$ ./openshift-install --dir ./demo wait-for cluster-ready

INFO Waiting up to 30m0s for the cluster at
https://api.demo.example.com:6443 to initialize...
INFO Install complete!
```

Booting all machines



Boot the cluster



Bootstrapping process step by step:

1. Bootstrap machine boots and starts hosting the remote resources required for master machines to boot.
2. Master machines fetch the remote resources from the bootstrap machine and finish booting.
3. Master machines use the bootstrap node to form an etcd cluster.
4. Bootstrap node starts a temporary Kubernetes control plane using the newly-created etcd cluster.
5. Temporary control plane schedules the production control plane to the master machines.
6. Temporary control plane shuts down, yielding to the production control plane.
7. Bootstrap node injects OpenShift-specific components into the newly formed control plane.
8. Installer then tears down the bootstrap node or if user-provisioned, this needs to be performed by the administrator.

Add a node

BareMetal

```
label linux
  menu label ^Install RHEL CoreOS
    kernel /images/vmlinuz
    append initrd=/images/initramfs.img nomodeset rd.neednet=1 coreos.inst=yes
    ip=192.168.122.9::192.168.122.1:255.255.255.0:core2.example.com:enp1s0:none nameserver=192.168.122.1
    coreos.inst.install_dev=sda coreos.inst.image_url=http://192.168.122.1/rhcos.raw.gz
    coreos.inst.ignition_url=http://192.168.122.1/static.ign
```

Machine Set

aws-qvnbv-worker-eu-central x

console-openshift-console.apps.aws.openshift.pub/k8s/ns/openshift-machine-a...

Red Hat OpenShift Container

admin

Builds

Monitoring

Compute

Nodes

Machines

Machine Sets

Machine Autoscalers

Machine Configs

Machine Config Pools

Edit Count

Machine Sets maintain the proper number of healthy machines.

1

Cancel Save

Overview YAML Machines Events

Machine Set Overview

Desired Count	Current Count	Ready Count	Available Count
1 machine	1 machine	1 machine	1 machine

Add a node

1. Boot a node with minimal ignition config:
 - a. certificateAuthorities
 - b. Point to MCO: [https://api-int.demo.openshift.pub:22623/config/\(master|worker\)](https://api-int.demo.openshift.pub:22623/config/(master|worker))
2. Node create a CSR (oc get csr)
3. Someone approved CSR
 - a. UPI: administrator 🧑 - during installation auto-approve!
 - b. IPI: Machine Operator
4. Node join the cluster
5. Node starts all pods they need to be a perfect member! (sdn, registry ca,...)

Add a node RHEL Node



RED HAT[®]
ANSIBLE[®]

36

```
subscription-manager register --username=<user_name> --password=<password>

subscription-manager repos \
  --enable="rhel-7-server-rpms" \
  --enable="rhel-7-server-extras-rpms" \
  --enable="rhel-7-server-ose-4.7-rpms"

# vi inventory/hosts
----
[all:vars]
ansible_user=root
#ansible_become=True

openshift_kubeconfig_path=~/.kube/config"

[new_workers]
mycluster-worker-0.example.com
mycluster-worker-1.example.com

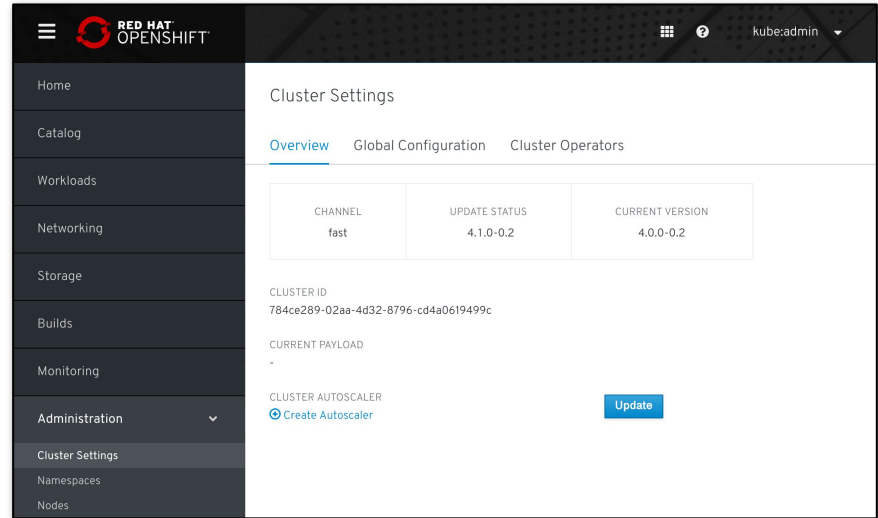
----

ansible-playbook -i /<path>/inventory/hosts playbooks/scaleup.yml
```

Over the Air (OTA) Updates!
WITHOUT

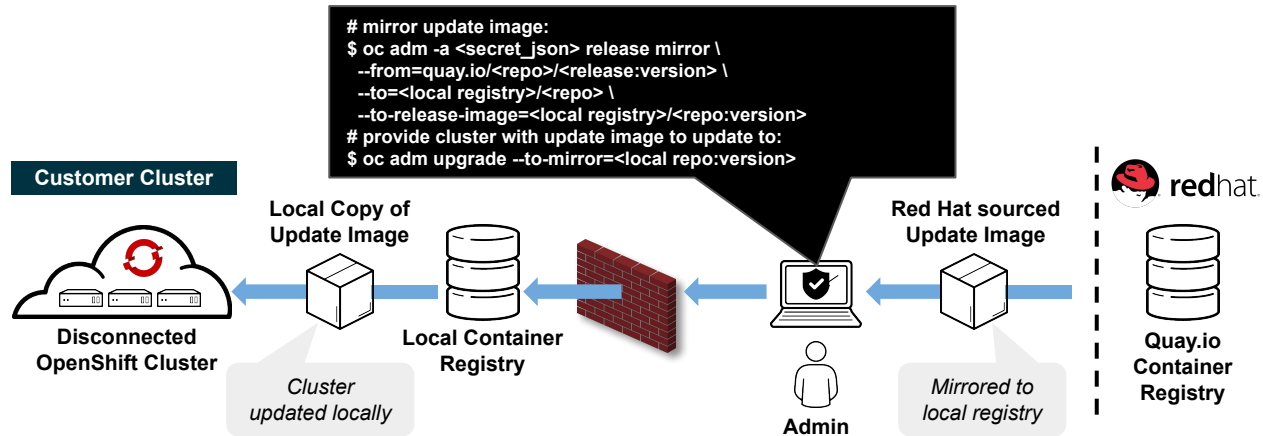
Over the Air (OTA) Updates

- OpenShift retrieves the list of available updates
- Admin selects the target version
- OpenShift is updated over the air
- Auto-update support



Only with Red Hat CoreOS - it doesn't matter how (IPI vs UPI) you install the cluster!

Disconnected “Air-gapped” Installation & Upgrading



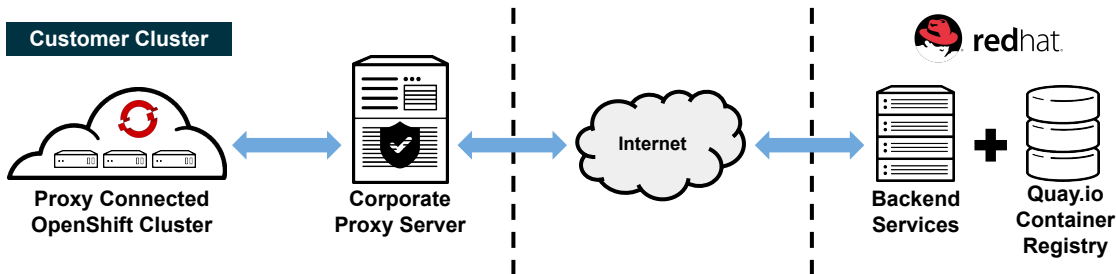
Overview

- 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 `OPENSIFT_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

Cluster-wide Egress Proxy



An admin with privileges can interact with the proxy object using 'oc' commands (use the 'oc edit' command to modify the proxy information.) Here is an example proxy config:

```
$ oc get proxy/cluster -o yaml
apiVersion: config.openshift.io/v1
kind: Proxy
metadata:
  creationTimestamp: "2019-08-21T22:36:49Z"
  generation: 2
  name: cluster
  resourceVersion: "24913"
  selfLink: /apis/config.openshift.io/v1/proxies/cluster
  uid: 2a344b01-d267-11f9-a4f3-025de4b59c38
spec:
  httpProxy: http://<username>:<pswd>@<ip>:<port>
  httpsProxy: https://<username>:<pswd>@<ip>:<port>
  noProxy: example.com
  readinessEndpoints:
  - http://www.google.com
  - https://www.google.com
  trustedCA:
    name: user-ca-bundle
status:
  httpProxy: http://<username>:<pswd>@<ip>:<port>
  httpsProxy: https://<username>:<pswd>@<ip>:<port>
  noProxy:
  10.0.0.0/16,10.128.0.0/14,127.0.0.1,169.254.169.254,172.30
  .0.0/16,api-int.demo.example.com,api.demo.example.openshif
  t.com,etcd-0.demo.example.com,etcd-1.demo.example.com,etcd
  -2.demo.example.com,example.com,localhost
```

Overview


- 4.2 introduces support for installing and updating OpenShift clusters through a corporate proxy server
- Leverages new proxy controller within the cluster-network-operator, which is responsible for:
 - Reconciling a proxy object and writing spec > status upon successful validation.
 - Reconciling user-provided trust bundles referenced by trustedCA, validating the trust bundle certificates, merging the certificates with the system trust bundle and publishing the merged bundle to the openshift-config-managed/trusted-ca-bundle configmap.

Installation Procedure


- Installer will use PROXY* environment variables from the shell it's invoked from
- Generate install-config.yaml: `./openshift-install create install-config --dir <dir>`
 - Edit proxy information (httpProxy, httpsProxy, & noProxy) and CA certificate ('additionalTrustBundle') to install-config.yaml
- Installer validates the provided install-config.yaml parameters, renders the necessary assets to create the cluster, and initiates the installation process based on the install method used:


`./openshift-install create cluster --dir <dir>`

Questions?

 [linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)

 [facebook.com/redhatinc](https://www.facebook.com/redhatinc)

 [youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)

 twitter.com/RedHat