CHAPTER 1. PREPARING FOR BARE METAL CLUSTER INSTALLATION

1.1. PREREQUISITES

- You reviewed details about the OpenShift Container Platform installation and update processes.

- You have read the documentation on selecting a cluster installation method and preparing it for users.

1.2. PLANNING A BARE METAL CLUSTER FOR OPENSHIFT VIRTUALIZATION

If you will use OpenShift Virtualization, it is important to be aware of several requirements before you install your bare metal cluster.

- If you want to use live migration features, you must have multiple worker nodes *at the time of cluster installation.* This is because live migration requires the cluster-level high availability (HA) flag to be set to true. The HA flag is set when a cluster is installed and cannot be changed afterwards. If there are fewer than two worker nodes defined when you install your cluster, the HA flag is set to false for the life of the cluster.

  > **NOTE**
  >
  > You can install OpenShift Virtualization on a single-node cluster, but single-node OpenShift does not support high availability.

- Live migration requires shared storage. Storage for OpenShift Virtualization must support and use the ReadWriteMany (RWX) access mode.

- If you plan to use Single Root I/O Virtualization (SR-IOV), ensure that your network interface controllers (NICs) are supported by OpenShift Container Platform.

Additional resources

- Getting started with OpenShift Virtualization

- Preparing your cluster for OpenShift Virtualization

- About Single Root I/O Virtualization (SR-IOV) hardware networks

- Connecting a virtual machine to an SR-IOV network

1.3. NIC PARTITIONING FOR SR-IOV DEVICES

OpenShift Container Platform can be deployed on a server with a dual port network interface card (NIC). You can partition a single, high-speed dual port NIC into multiple virtual functions (VFs) and enable SR-IOV.

This feature supports the use of bonds for high availability with the Link Aggregation Control Protocol (LACP).
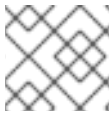
> **NOTE**
>
> Only one LACP can be declared by physical NIC.

An OpenShift Container Platform cluster can be deployed on a bond interface with 2 VFs on 2 physical functions (PFs) using the following methods:

- Agent-based installer

  > **NOTE**
  >
  > The minimum required version of **nmstate** is:
  >
  > - **1.4.2-4** for RHEL 8 versions
  >
  > - **2.2.7** for RHEL 9 versions

- Installer-provisioned infrastructure installation

- User-provisioned infrastructure installation

Additional resources

- Example: Bonds and SR-IOV dual-nic node network configuration

- Optional: Configuring host network interfaces for dual port NIC

- Bonding multiple SR-IOV network interfaces to a dual port NIC interface

1.4. CHOOSING A METHOD TO INSTALL OPENSHIFT CONTAINER PLATFORM ON BARE METAL

The OpenShift Container Platform installation program offers four methods for deploying a cluster:

- **Interactive**: You can deploy a cluster with the web-based Assisted Installer. This is the recommended approach for clusters with networks connected to the internet. The Assisted Installer is the easiest way to install OpenShift Container Platform, it provides smart defaults, and it performs pre-flight validations before installing the cluster. It also provides a RESTful API for automation and advanced configuration scenarios.

- **Local Agent-based**: You can deploy a cluster locally with the agent-based installer for air-gapped or restricted networks. It provides many of the benefits of the Assisted Installer, but you must download and configure the agent-based installer first. Configuration is done with a commandline interface. This approach is ideal for air-gapped or restricted networks.

- **Automated**: You can deploy a cluster on installer-provisioned infrastructure and the cluster it maintains. The installer uses each cluster host's baseboard management controller (BMC) for provisioning. You can deploy clusters with both connected or air-gapped or restricted networks.

- **Full control**: You can deploy a cluster on infrastructure that you prepare and maintain, which provides maximum customizability. You can deploy clusters with both connected or air-gapped or restricted networks.

The clusters have the following characteristics:

- Highly available infrastructure with no single points of failure is available by default.

- Administrators maintain control over what updates are applied and when.

See Installation process for more information about installer-provisioned and user-provisioned installation processes.

1.4.1. Installing a cluster on installer-provisioned infrastructure

You can install a cluster on bare metal infrastructure that is provisioned by the OpenShift Container Platform installation program, by using the following method:

- **Installing an installer-provisioned cluster on bare metal**: You can install OpenShift Container Platform on bare metal by using installer provisioning.

### 1.4.2. Installing a cluster on user-provisioned infrastructure

You can install a cluster on bare metal infrastructure that you provision, by using one of the following methods:

- **Installing a user-provisioned cluster on bare metal**: You can install OpenShift Container Platform on bare metal infrastructure that you provision. For a cluster that contains user-provisioned infrastructure, you must deploy all of the required machines.

- **Installing a user-provisioned bare metal cluster with network customizations**: You can install a bare metal cluster on user-provisioned infrastructure with network-customizations. By customizing your network configuration, your cluster can coexist with existing IP address allocations in your environment and integrate with existing MTU and VXLAN configurations. Most of the network customizations must be applied at the installation stage.

- **Installing a user-provisioned bare metal cluster on a restricted network**: You can install a user-provisioned bare metal cluster on a restricted or disconnected network by using a mirror registry. You can also use this installation method to ensure that your clusters only use container images that satisfy your organizational controls on external content.

CHAPTER 2. USER-PROVISIONED INFRASTRUCTURE

2.1. INSTALLING A USER-PROVISIONED CLUSTER ON BARE METAL

In OpenShift Container Platform 4.18, you can install a cluster on bare metal infrastructure that you provision.
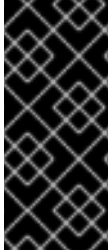
> **IMPORTANT**
>
> While you might be able to follow this procedure to deploy a cluster on virtualized or cloud environments, you must be aware of additional considerations for non-bare metal platforms. Review the information in the guidelines for deploying OpenShift Container Platform on non-tested platforms before you attempt to install an OpenShift Container Platform cluster in such an environment.

2.1.1. Prerequisites

- You reviewed details about the OpenShift Container Platform installation and update processes.

- You read the documentation on selecting a cluster installation method and preparing it for users.

- If you use a firewall, you configured it to allow the sites that your cluster requires access to.

  > **NOTE**
  >
  > Be sure to also review this site list if you are configuring a proxy.

2.1.2. Internet access for OpenShift Container Platform

In OpenShift Container Platform 4.18, you require access to the internet to install your cluster.

You must have internet access to:

- Access OpenShift Cluster Manager to download the installation program and perform subscription management. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster.

- Access Quay.io to obtain the packages that are required to install your cluster.

- Obtain the packages that are required to perform cluster updates.

> **IMPORTANT**
>
> If your cluster cannot have direct internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the required content and use it to populate a mirror registry with the installation packages. With some installation types, the environment that you install your cluster in will not require internet access. Before you update the cluster, you update the content of the mirror registry.

Additional resources

- See Installing a user-provisioned bare metal cluster on a restricted network for more information about performing a restricted network installation on bare metal infrastructure that you provision.

2.1.3. Requirements for a cluster with user-provisioned infrastructure

For a cluster that contains user-provisioned infrastructure, you must deploy all of the required machines.

This section describes the requirements for deploying OpenShift Container Platform on user-provisioned infrastructure.

2.1.3.1. Required machines for cluster installation

The smallest OpenShift Container Platform clusters require the following hosts:

Table 2.1. Minimum required hosts

Hosts	Description
One temporary bootstrap machine	The cluster requires the bootstrap machine to deploy the OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster.
Three control plane machines	The control plane machines run the Kubernetes and OpenShift Container Platform services that form the control plane.
At least two compute machines, which are also known as worker machines.	The workloads requested by OpenShift Container Platform users run on the compute machines.

> **NOTE**
>
> As an exception, you can run zero compute machines in a bare metal cluster that consists of three control plane machines only. This provides smaller, more resource efficient clusters for cluster administrators and developers to use for testing, development, and production. Running one compute machine is not supported.

> **IMPORTANT**
>
> To maintain high availability of your cluster, use separate physical hosts for these cluster machines.

The bootstrap and control plane machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system. However, the compute machines can choose between Red Hat Enterprise Linux CoreOS (RHCOS), Red Hat Enterprise Linux (RHEL) 8.6 and later.

Note that RHCOS is based on Red Hat Enterprise Linux (RHEL) 9.2 and inherits all of its hardware certifications and requirements. See Red Hat Enterprise Linux technology capabilities and limits .

2.1.3.2. Minimum resource requirements for cluster installation

Each cluster machine must meet the following minimum requirements:

Table 2.2. Minimum resource requirements

Machine	Operating System	CPU [1]	RAM	Storage	Input/Output Per Second (IOPS)[2]
Bootstrap	RHCOS	4	16 GB	100 GB	300
Control plane	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS, RHEL 8.6 and later [3]	2	8 GB	100 GB	300

1. One CPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = CPUs.

2. OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.

3. As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.

> **NOTE**
>
> For OpenShift Container Platform version 4.18, RHCOS is based on RHEL version 9.4, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:
>
> - x86-64 architecture requires x86-64-v2 ISA
>
> - ARM64 architecture requires ARMv8.0-A ISA
>
> - IBM Power architecture requires Power 9 ISA
>
> - s390x architecture requires z14 ISA
>
> For more information, see Architectures (RHEL documentation).

If an instance type for your platform meets the minimum requirements for cluster machines, it is supported to use in OpenShift Container Platform.

Additional resources

- Optimizing storage

2.1.3.3. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

Additional resources

- See Configuring a three-node cluster for details about deploying three-node clusters in bare metal environments.

- See Approving the certificate signing requests for your machines for more information about approving cluster certificate signing requests after installation.

2.1.3.4. Requirements for baremetal clusters on vSphere

Ensure you enable the **disk.EnableUUID** parameter on all virtual machines in your cluster.

Additional resources

- See Installing RHCOS and starting the OpenShift Container Platform bootstrap process for details on setting the **disk.EnableUUID** parameter's value to **TRUE** on VMware vSphere for user-provisioned infrastructure.

2.1.3.5. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require networking to be configured in **initramfs** during boot to fetch their Ignition config files.

During the initial boot, the machines require an IP address configuration that is set either through a DHCP server or statically by providing the required boot options. After a network connection is established, the machines download their Ignition config files from an HTTP or HTTPS server. The Ignition config files are then used to set the exact state of each machine. The Machine Config Operator completes more changes to the machines, such as the application of new certificates or keys, after installation.

It is recommended to use a DHCP server for long-term management of the cluster machines. Ensure that the DHCP server is configured to provide persistent IP addresses, DNS server information, and hostnames to the cluster machines.

> **NOTE**
>
> If a DHCP service is not available for your user-provisioned infrastructure, you can instead provide the IP networking configuration and the address of the DNS server to the nodes at RHCOS install time. These can be passed as boot arguments if you are installing from an ISO image. See the *Installing RHCOS and starting the OpenShift Container Platform bootstrap process* section for more information about static IP provisioning and advanced networking options.

The Kubernetes API server must be able to resolve the node names of the cluster machines. If the API servers and worker nodes are in different zones, you can configure a default DNS search zone to allow

the API server to resolve the node names. Another supported approach is to always refer to hosts by their fully-qualified domain names in both the node objects and all DNS requests.

2.1.3.5.1. Setting the cluster node hostnames through DHCP

On Red Hat Enterprise Linux CoreOS (RHCOS) machines, the hostname is set through NetworkManager. By default, the machines obtain their hostname through DHCP. If the hostname is not provided by DHCP, set statically through kernel arguments, or another method, it is obtained through a reverse DNS lookup. Reverse DNS lookup occurs after the network has been initialized on a node and can take time to resolve. Other system services can start prior to this and detect the hostname as **localhost** or similar. You can avoid this by using DHCP to provide the hostname for each cluster node.

Additionally, setting the hostnames through DHCP can bypass any manual DNS record name configuration errors in environments that have a DNS split-horizon implementation.

2.1.3.5.2. Network connectivity requirements

You must configure the network connectivity between machines to allow OpenShift Container Platform cluster components to communicate. Each machine must be able to resolve the hostnames of all other machines in the cluster.

This section provides details about the ports that are required.

IMPORTANT

In connected OpenShift Container Platform environments, all nodes are required to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Table 2.3. Ports used for all-machine to all-machine communications

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	**1936**	Metrics
	9000-**9999**	Host level services, including the node exporter on ports **9100**-**9101** and the Cluster Version Operator on port **9099**.
	10250-**10259**	The default ports that Kubernetes reserves
UDP	**4789**	VXLAN
	6081	Geneve
	9000-**9999**	Host level services, including the node exporter on ports **9100**-**9101**.
	500	IPsec IKE packets

Protocol	Port	Description
	4500	IPsec NAT-T packets
	123	Network Time Protocol (NTP) on UDP port **123** If an external NTP time server is configured, you must open UDP port **123**.
TCP/UDP	**30000**-**32767**	Kubernetes node port
ESP	N/A	IPsec Encapsulating Security Payload (ESP)

Table 2.4. Ports used for all-machine to control plane communications

Protocol	Port	Description
TCP	**6443**	Kubernetes API

Table 2.5. Ports used for control plane machine to control plane machine communications

Protocol	Port	Description
TCP	**2379**-**2380**	etcd server and peer ports

NTP configuration for user-provisioned infrastructure

OpenShift Container Platform clusters are configured to use a public Network Time Protocol (NTP) server by default. If you want to use a local enterprise NTP server, or if your cluster is being deployed in a disconnected network, you can configure the cluster to use a specific time server. For more information, see the documentation for *Configuring chrony time service* .

If a DHCP server provides NTP server information, the chrony time service on the Red Hat Enterprise Linux CoreOS (RHCOS) machines read the information and can sync the clock with the NTP servers.

Additional resources

- Configuring chrony time service

2.1.3.6. User-provisioned DNS requirements

In OpenShift Container Platform deployments, DNS name resolution is required for the following components:

- The Kubernetes API

- The OpenShift Container Platform application wildcard

- The bootstrap, control plane, and compute machines

Reverse DNS resolution is also required for the Kubernetes API, the bootstrap machine, the control plane machines, and the compute machines.

DNS A/AAAA or CNAME records are used for name resolution and PTR records are used for reverse name resolution. The reverse records are important because Red Hat Enterprise Linux CoreOS (RHCOS) uses the reverse records to set the hostnames for all the nodes, unless the hostnames are provided by DHCP. Additionally, the reverse records are used to generate the certificate signing requests (CSR) that OpenShift Container Platform needs to operate.
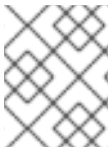
> **NOTE**
>
> It is recommended to use a DHCP server to provide the hostnames to each cluster node. See the *DHCP recommendations for user-provisioned infrastructure* section for more information.

The following DNS records are required for a user-provisioned OpenShift Container Platform cluster and they must be in place before installation. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the base domain that you specify in the **install-config.yaml** file. A complete DNS record takes the form: **<component>.<cluster_name>.<base_domain>.**.

Table 2.6. Required DNS records

Component	Record	Description
Kubernetes API	**api.<cluster_name>.<base_domain>.**	A DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the API load balancer. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
	api-int.<cluster_name>.<base_domain>.	A DNS A/AAAA or CNAME record, and a DNS PTR record, to internally identify the API load balancer. These records must be resolvable from all the nodes within the cluster. **IMPORTANT** The API server must be able to resolve the worker nodes by the hostnames that are recorded in Kubernetes. If the API server cannot resolve the node names, then proxied API calls can fail, and you cannot retrieve logs from pods.
Routes	***.apps.<cluster_name>.<base_domain>.**	A wildcard DNS A/AAAA or CNAME record that refers to the application ingress load balancer. The application ingress load balancer targets the machines that run the Ingress Controller pods. The Ingress Controller pods run on the compute machines by default. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster. For example, **console-openshift-console.apps.<cluster_name>.<base_domain>** is used as a wildcard route to the OpenShift Container Platform console.

Compo nent	Record	Description
Bootstra p machine	**bootstrap.<cluster_name>. <base_domain>.**	A DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the bootstrap machine. These records must be resolvable by the nodes within the cluster.
Control plane machine s	**<control_plane><n>. <cluster_name>. <base_domain>.**	DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the control plane nodes. These records must be resolvable by the nodes within the cluster.
Comput e machine s	**<compute><n>. <cluster_name>. <base_domain>.**	DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the worker nodes. These records must be resolvable by the nodes within the cluster.

NOTE

In OpenShift Container Platform 4.4 and later, you do not need to specify etcd host and SRV records in your DNS configuration.

TIP

You can use the **dig** command to verify name and reverse name resolution. See the section on *Validating DNS resolution for user-provisioned infrastructure* for detailed validation steps.

2.1.3.6.1. Example DNS configuration for user-provisioned clusters

This section provides A and PTR record configuration samples that meet the DNS requirements for deploying OpenShift Container Platform on user-provisioned infrastructure. The samples are not meant to provide advice for choosing one DNS solution over another.

In the examples, the cluster name is **ocp4** and the base domain is **example.com**.

Example DNS A record configuration for a user-provisioned cluster

The following example is a BIND zone file that shows sample A records for name resolution in a user-provisioned cluster.

Example 2.1. Sample DNS zone database

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
```

```
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5
api-int.ocp4.example.com. IN A 192.168.1.5
;
*.apps.ocp4.example.com. IN A 192.168.1.5
;
bootstrap.ocp4.example.com. IN A 192.168.1.96
;
control-plane0.ocp4.example.com. IN A 192.168.1.97
control-plane1.ocp4.example.com. IN A 192.168.1.98
control-plane2.ocp4.example.com. IN A 192.168.1.99
;
compute0.ocp4.example.com. IN A 192.168.1.11
compute1.ocp4.example.com. IN A 192.168.1.7
;
;EOF
```

[1] api.ocp4.example.com. IN A 192.168.1.5

[2] api-int.ocp4.example.com. IN A 192.168.1.5

[3] *.apps.ocp4.example.com. IN A 192.168.1.5

[4] bootstrap.ocp4.example.com. IN A 192.168.1.96

[5] control-plane0.ocp4.example.com. IN A 192.168.1.97

[6] control-plane1.ocp4.example.com. IN A 192.168.1.98

[7] control-plane2.ocp4.example.com. IN A 192.168.1.99

[8] compute0.ocp4.example.com. IN A 192.168.1.11

[9] compute1.ocp4.example.com. IN A 192.168.1.7

[1] Provides name resolution for the Kubernetes API. The record refers to the IP address of the API load balancer.

[2] Provides name resolution for the Kubernetes API. The record refers to the IP address of the API load balancer and is used for internal cluster communications.

[3] Provides name resolution for the wildcard routes. The record refers to the IP address of the application ingress load balancer. The application ingress load balancer targets the machines that run the Ingress Controller pods. The Ingress Controller pods run on the compute machines by default.

> **NOTE**
>
> In the example, the same load balancer is used for the Kubernetes API and application ingress traffic. In production scenarios, you can deploy the API and application ingress load balancers separately so that you can scale the load balancer infrastructure for each in isolation.

[4] Provides name resolution for the bootstrap machine.

[5] [6] [7] Provides name resolution for the control plane machines.

[8] [9] Provides name resolution for the compute machines.

Example DNS PTR record configuration for a user-provisioned cluster

The following example BIND zone file shows sample PTR records for reverse name resolution in a user-provisioned cluster.

Example 2.2. Sample DNS zone database for reverse records

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com.           1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com.       2
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com.    3
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com.   4
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com.   5
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com.   6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com.     7
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com.      8
;
;EOF
```

[1] Provides reverse DNS resolution for the Kubernetes API. The PTR record refers to the record name of the API load balancer.

[2] Provides reverse DNS resolution for the Kubernetes API. The PTR record refers to the record name of the API load balancer and is used for internal cluster communications.

[3] Provides reverse DNS resolution for the bootstrap machine.

[4] [5] [6] Provides reverse DNS resolution for the control plane machines.

[7] [8] Provides reverse DNS resolution for the compute machines.

NOTE

A PTR record is not required for the OpenShift Container Platform application wildcard.

Additional resources

- Validating DNS resolution for user-provisioned infrastructure

2.1.3.7. Load balancing requirements for user-provisioned infrastructure

Before you install OpenShift Container Platform, you must provision the API and application Ingress load balancing infrastructure. In production scenarios, you can deploy the API and application Ingress load balancers separately so that you can scale the load balancer infrastructure for each in isolation.

NOTE

If you want to deploy the API and application Ingress load balancers with a Red Hat Enterprise Linux (RHEL) instance, you must purchase the RHEL subscription separately.

The load balancing infrastructure must meet the following requirements:

1. **API load balancer**: Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:

   - Layer 4 load balancing only. This can be referred to as Raw TCP or SSL Passthrough mode.

   - A stateless load balancing algorithm. The options vary based on the load balancer implementation.

   IMPORTANT

   Do not configure session persistence for an API load balancer. Configuring session persistence for a Kubernetes API server might cause performance issues from excess application traffic for your OpenShift Container Platform cluster and the Kubernetes API that runs inside the cluster.

Configure the following ports on both the front and back of the load balancers:

Table 2.7. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the **/readyz** endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server

NOTE

The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the **/readyz** endpoint to the removal of the API server instance from the pool. Within the time frame after **/readyz** returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

2. **Application Ingress load balancer**. Provides an ingress point for application traffic flowing in from outside the cluster. A working configuration for the Ingress router is required for an OpenShift Container Platform cluster.

   Configure the following conditions:

   - Layer 4 load balancing only. This can be referred to as Raw TCP or SSL Passthrough mode.

   - A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

TIP

If the true IP address of the client can be seen by the application Ingress load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.

Configure the following ports on both the front and back of the load balancers:

Table 2.8. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress Controller pods, compute, or worker, by default.	X	X	HTTPS traffic
80	The machines that run the Ingress Controller pods, compute, or worker, by default.	X	X	HTTP traffic

NOTE

If you are deploying a three-node cluster with zero compute nodes, the Ingress Controller pods run on the control plane nodes. In three-node cluster deployments, you must configure your application Ingress load balancer to route HTTP and HTTPS traffic to the control plane nodes.

2.1.3.7.1. Example load balancer configuration for user-provisioned clusters

This section provides an example API and application Ingress load balancer configuration that meets the load balancing requirements for user-provisioned clusters. The sample is an **/etc/haproxy/haproxy.cfg** configuration for an HAProxy load balancer. The example is not meant to provide advice for choosing one load balancing solution over another.

In the example, the same load balancer is used for the Kubernetes API and application ingress traffic. In production scenarios, you can deploy the API and application ingress load balancers separately so that you can scale the load balancer infrastructure for each in isolation.

NOTE

If you are using HAProxy as a load balancer and SELinux is set to **enforcing**, you must ensure that the HAProxy service can bind to the configured TCP port by running **setsebool -P haproxy_connect_any=1**.

Example 2.3. Sample API and application Ingress load balancer configuration

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                http
  log                 global
  option              dontlognull
  option http-server-close
  option              redispatch
  retries             3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
listen api-server-6443 1
  bind *:6443
  mode tcp
  option  httpchk GET /readyz HTTP/1.0
  option  log-health-checks
  balance roundrobin
  server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2
rise 3 backup 2
  server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
  server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s
fall 2 rise 3
listen machine-config-server-22623 3
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup 4
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 5
  bind *:443
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:443 check inter 1s
  server compute1 compute1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 6
  bind *:80
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:80 check inter 1s
  server compute1 compute1.ocp4.example.com:80 check inter 1s
```