



OpenShift Container Platform 4.20

Networking overview

Understanding fundamental networking concepts and general tasks in OpenShift Container Platform

OpenShift Container Platform 4.20 Networking overview

Understanding fundamental networking concepts and general tasks in OpenShift Container Platform

Legal Notice

Copyright © 2025 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This document provides an introduction to core networking concepts, basic architecture, and general networking tasks within OpenShift Container Platform.

Table of Contents

CHAPTER 1. UNDERSTANDING NETWORKING	3
1.1. CORE NETWORK LAYERS AND COMPONENTS	3
1.1.1. The pod network	3
1.1.2. The service network	3
1.2. MANAGING TRAFFIC WITHIN THE CLUSTER	3
1.2.1. Pod-to-pod communication	3
1.2.2. Service discovery with DNS	3
1.3. MANAGING TRAFFIC ENTERING AND LEAVING THE CLUSTER	4
1.3.1. Exposing applications with Ingress and Route objects	4
1.3.2. Distributing traffic with Load Balancers	4
1.3.3. Controlling Egress traffic	4
1.4. SECURING NETWORK TRAFFIC	4
1.4.1. Network policies	5
1.4.2. Administrative network policies	5
CHAPTER 2. ACCESSING HOSTS	6
2.1. ACCESSING HOSTS ON AMAZON WEB SERVICES IN AN INSTALLER-PROVISIONED INFRASTRUCTURE CLUSTER	6
CHAPTER 3. NETWORKING DASHBOARDS	7
3.1. NETWORK OBSERVABILITY OPERATOR	7
3.2. NETWORKING AND OVN-KUBERNETES DASHBOARD	7
3.3. INGRESS OPERATOR DASHBOARD	7
CHAPTER 4. CIDR RANGE DEFINITIONS	8
4.1. MACHINE CIDR	9
4.2. SERVICE CIDR	9
4.3. POD CIDR	9
4.4. HOST PREFIX	9
4.5. CIDR RANGES FOR HOSTED CONTROL PLANES	10

CHAPTER 1. UNDERSTANDING NETWORKING

Understanding networking is essential for building resilient, secure, and scalable applications in OpenShift Container Platform. From basic pod-to-pod communication to complex traffic routing and security rules, every component of your application relies on the network to function correctly.

1.1. CORE NETWORK LAYERS AND COMPONENTS

Red Hat OpenShift Networking is built on two fundamental layers: the **pod network** and the **service network**. The pod network is where your applications live. The service network makes your applications reliably accessible.

1.1.1. The pod network

The pod network is a flat network space where every pod in the cluster receives its own unique IP address. This network is managed by the Container Network Interface (CNI) plugin. The CNI plugin is responsible for wiring each pod into the cluster network.

This design allows pods to communicate directly with each other using their IP addresses, regardless of which node they are running on. However, these pod IPs are ephemeral. This means the IPs are destroyed when the pod is destroyed and a new IP address is assigned when a new pod is created. Because of this, you should never rely on pod IP addresses directly for long-lived communication.

1.1.2. The service network

A service is a networking object that provides a single, stable virtual IP address, called a ClusterIP, and a DNS name for a logical group of pods.

When a request is sent to a service's ClusterIP, OpenShift Container Platform automatically load-balances the traffic to one of the healthy pods backing that service. It uses Kubernetes labels and selectors to keep track of which pods belong to which service. This abstraction makes your applications resilient because individual pods can be created or destroyed without affecting the applications trying to reach them.

1.2. MANAGING TRAFFIC WITHIN THE CLUSTER

Your applications need to communicate with each other inside the cluster. OpenShift Container Platform provides two primary mechanisms for internal traffic: direct pod-to-pod communication for simple exchanges and robust service discovery for reliable connections.

1.2.1. Pod-to-pod communication

Pods communicate directly using the unique IP addresses assigned by the pod network. A pod on one node can send traffic directly to a pod on another node without any network address translation (NAT). This direct communication model is efficient for services that need to exchange data quickly. Applications can simply target another pod's IP address to establish a connection.

1.2.2. Service discovery with DNS

Pods need a reliable way to find each other because pod IP addresses are ephemeral. OpenShift Container Platform uses **CoreDNS**, a built-in DNS server, to provide this service discovery.

Every service you create automatically receives a stable DNS name. A pod can use this DNS name to connect to the service. The DNS system resolves the name to the service's stable **ClusterIP** address. This process ensures reliable communication even when individual pod IPs change.

1.3. MANAGING TRAFFIC ENTERING AND LEAVING THE CLUSTER

You need a way for external users to access your applications and for your applications to securely access external services. OpenShift Container Platform provides several tools to manage this flow of traffic into and out of your cluster.

1.3.1. Exposing applications with Ingress and Route objects

To allow external traffic to reach services inside your cluster, you use an Ingress Controller. This component acts as the front door that directs incoming requests to the correct application. You define the traffic rules using one of two primary resources:

- **Ingress**: The standard Kubernetes resource for managing external access to services, typically for HTTP and HTTPS traffic.
- **Route object**: A resource that provides the same functionality as Ingress but includes additional features like more advanced TLS termination options and traffic splitting. **Route** objects are specific to OpenShift Container Platform.

1.3.2. Distributing traffic with Load Balancers

A Load Balancer provides a single, highly available IP address for directing traffic to your cluster. It typically runs outside the cluster on a cloud provider or using MetalLB on bare-metal infrastructure and distributes incoming requests across multiple nodes that are running the Ingress Controller.

This prevents any single node from becoming a bottleneck or a point of failure to ensure that your applications remain accessible.

1.3.3. Controlling Egress traffic

Egress refers to outbound traffic that originates from a pod inside the cluster and is destined for an external system. OpenShift Container Platform provides several mechanisms to manage this:

- **EgressIP**: You can assign a specific, predictable source IP address to all outbound traffic from a given project. This is useful when you need to access an external service like a database that has a firewall requiring you to allow specific source IPs.
- **Egress Router**: This is a dedicated pod that acts as a gateway for outbound traffic. It allows you to route connections through a single, controlled exit point.
- **Egress Firewall**: This acts as a cluster-level firewall for all outbound traffic. It enhances your security posture by allowing you to create rules that explicitly allow or deny connections from pods to specific external destinations.

1.4. SECURING NETWORK TRAFFIC

OpenShift Container Platform provides tools to secure your network by creating rules that control which components are allowed to communicate. This is primarily managed through two types of policy resources: network policies and administrative network policies.

1.4.1. Network policies

A network policy is a resource that allows you to control the flow of traffic at the IP address or port level. These policies operate at the namespace (project) level. This means they are typically managed by developers or project administrators to secure their specific applications.

By default, all pods in a project can communicate with each other freely. However, when you apply a **NetworkPolicy** to a pod, it adopts a "default-deny" stance. This means it rejects any connection that is not explicitly allowed by a policy rule. You use labels and selectors to define which pods a policy applies to and what ingress and egress traffic is permitted.

1.4.2. Administrative network policies

An **AdminNetworkPolicy** object is a more powerful, cluster-scoped version of a **NetworkPolicy** object. It can only be created and managed by a cluster administrator.

Administrative network policies have a higher priority than standard **NetworkPolicy** objects. This allows administrators to enforce cluster-wide security rules that cannot be overridden by users in their own projects. For example, an administrator could use an **AdminNetworkPolicy** to block all traffic between development and production namespaces or to enforce baseline security rules for the entire cluster.

Additional resources

- [About network policy](#)

CHAPTER 2. ACCESSING HOSTS

Learn how to create a bastion host to access OpenShift Container Platform instances and access the control plane nodes with secure shell (SSH) access.

2.1. ACCESSING HOSTS ON AMAZON WEB SERVICES IN AN INSTALLER-PROVISIONED INFRASTRUCTURE CLUSTER

The OpenShift Container Platform installer does not create any public IP addresses for any of the Amazon Elastic Compute Cloud (Amazon EC2) instances that it provisions for your OpenShift Container Platform cluster. To be able to SSH to your OpenShift Container Platform hosts, you must follow this procedure.

Procedure

1. Create a security group that allows SSH access into the virtual private cloud (VPC) created by the **openshift-install** command.
2. Create an Amazon EC2 instance on one of the public subnets the installer created.
3. Associate a public IP address with the Amazon EC2 instance that you created.
Unlike with the OpenShift Container Platform installation, you should associate the Amazon EC2 instance you created with an SSH keypair. It does not matter what operating system you choose for this instance, as it will simply serve as an SSH bastion to bridge the internet into your OpenShift Container Platform cluster's VPC. The Amazon Machine Image (AMI) you use does matter. With Red Hat Enterprise Linux CoreOS (RHCOS), for example, you can provide keys via Ignition, like the installer does.
4. After you provisioned your Amazon EC2 instance and can SSH into it, you must add the SSH key that you associated with your OpenShift Container Platform installation. This key can be different from the key for the bastion instance, but does not have to be.



NOTE

Direct SSH access is only recommended for disaster recovery. When the Kubernetes API is responsive, run privileged pods instead.

5. Run **oc get nodes**, inspect the output, and choose one of the nodes that is a master. The hostname looks similar to **ip-10-0-1-163.ec2.internal**.
6. From the bastion SSH host you manually deployed into Amazon EC2, SSH into that control plane host. Ensure that you use the same SSH key you specified during the installation:

```
$ ssh -i <ssh-key-path> core@<master-hostname>
```

CHAPTER 3. NETWORKING DASHBOARDS

Networking metrics are viewable in dashboards within the OpenShift Container Platform web console, under **Observe** → **Dashboards**.

3.1. NETWORK OBSERVABILITY OPERATOR

If you have the Network Observability Operator installed, you can view network traffic metrics dashboards by selecting the **Netobserv** dashboard from the **Dashboards** drop-down list. For more information about metrics available in this **Dashboard**, see [Network Observability metrics dashboards](#).

3.2. NETWORKING AND OVN-KUBERNETES DASHBOARD

You can view both general networking metrics as well as OVN-Kubernetes metrics from the dashboard.

To view general networking metrics, select **Networking/Linux Subsystem Stats** from the **Dashboards** drop-down list. You can view the following networking metrics from the dashboard: **Network Utilisation**, **Network Saturation**, and **Network Errors**.

To view OVN-Kubernetes metrics select **Networking/Infrastructure** from the **Dashboards** drop-down list. You can view the following OVN-Kubernetes metrics: **Networking Configuration**, **TCP Latency Probes**, **Control Plane Resources**, and **Worker Resources**.

3.3. INGRESS OPERATOR DASHBOARD

You can view networking metrics handled by the Ingress Operator from the dashboard. This includes metrics like the following:

- Incoming and outgoing bandwidth
- HTTP error rates
- HTTP server response latency

To view these Ingress metrics, select **Networking/Ingress** from the **Dashboards** drop-down list. You can view Ingress metrics for the following categories: **Top 10 Per Route**, **Top 10 Per Namespace**, and **Top 10 Per Shard**

CHAPTER 4. CIDR RANGE DEFINITIONS

If your cluster uses OVN-Kubernetes, you must specify non-overlapping ranges for Classless Inter-Domain Routing (CIDR) subnet ranges.



IMPORTANT

For OpenShift Container Platform 4.17 and later versions, clusters use **169.254.0.0/17** for IPv4 and **fd69::/112** for IPv6 as the default masquerade subnet. These ranges should also be avoided by users. For upgraded clusters, there is no change to the default masquerade subnet.

TIP

You can use the [Red Hat OpenShift Network Calculator](#) to determine your networking needs prior to setting CIDR range during cluster creation.

You must have a Red Hat account to use the calculator.

The following subnet types and are mandatory for a cluster that uses OVN-Kubernetes:

- Join: Uses a join switch to connect gateway routers to distributed routers. A join switch reduces the number of IP addresses for a distributed router. For a cluster that uses the OVN-Kubernetes plugin, an IP address from a dedicated subnet is assigned to any logical port that attaches to the join switch.
- Masquerade: Prevents collisions for identical source and destination IP addresses that are sent from a node as hairpin traffic to the same node after a load balancer makes a routing decision.
- Transit: A transit switch is a type of distributed switch that spans across all nodes in the cluster. A transit switch routes traffic between different zones. For a cluster that uses the OVN-Kubernetes plugin, an IP address from a dedicated subnet is assigned to any logical port that attaches to the transit switch.



NOTE

You can change the join, masquerade, and transit CIDR ranges for your cluster as a post-installation task.

OVN-Kubernetes, the default network provider in OpenShift Container Platform 4.14 and later versions, internally uses the following IP address subnet ranges:

- **V4JoinSubnet: 100.64.0.0/16**
- **V6JoinSubnet: fd98::/64**
- **V4TransitSwitchSubnet: 100.88.0.0/16**
- **V6TransitSwitchSubnet: fd97::/64**
- **defaultV4MasqueradeSubnet: 169.254.0.0/17**
- **defaultV6MasqueradeSubnet: fd69::/112**



IMPORTANT

The previous list includes join, transit, and masquerade IPv4 and IPv6 address subnets. If your cluster uses OVN-Kubernetes, do not include any of these IP address subnet ranges in any other CIDR definitions in your cluster or infrastructure.

Additional resources

- For more information about configuring join subnets or transit subnets, see [Configuring OVN-Kubernetes internal IP address subnets](#).

4.1. MACHINE CIDR

In the Machine classless inter-domain routing (CIDR) field, you must specify the IP address range for machines or cluster nodes.



NOTE

Machine CIDR ranges cannot be changed after creating your cluster.

The default is **10.0.0.0/16**. This range must not conflict with any connected networks.

Additional resources

- [Cluster Network Operator configuration](#)

4.2. SERVICE CIDR

In the Service CIDR field, you must specify the IP address range for services. The range must be large enough to accommodate your workload. The address block must not overlap with any external service accessed from within the cluster. The default is **172.30.0.0/16**.

4.3. POD CIDR

In the pod CIDR field, you must specify the IP address range for pods.

The pod CIDR is the same as the **clusterNetwork** CIDR and the cluster CIDR. The range must be large enough to accommodate your workload. The address block must not overlap with any external service accessed from within the cluster. The default is **10.128.0.0/14**. You can expand the range after cluster installation.

Additional resources

- [Cluster Network Operator configuration](#)
- [Configuring the cluster network range](#)

4.4. HOST PREFIX

In the Host Prefix field, you must specify the subnet prefix length assigned to pods scheduled to individual machines. The host prefix determines the pod IP address pool for each machine.

For example, if the host prefix is set to **/23**, each machine is assigned a **/23** subnet from the pod CIDR address range. The default is **/23**, allowing 510 cluster nodes, and 510 pod IP addresses per node.

4.5. CIDR RANGES FOR HOSTED CONTROL PLANES

For deploying hosted control planes on OpenShift Container Platform, use the following required Classless Inter-Domain Routing (CIDR) subnet ranges:

- **v4InternalSubnet**: 100.65.0.0/16 (OVN-Kubernetes)
- **clusterNetwork**: 10.132.0.0/14 (pod network)
- **serviceNetwork**: 172.31.0.0/16

For more information about OpenShift Container Platform CIDR range definitions, see "CIDR range definitions".