OpenShift Container Platform 4.20

Networking overview

Understanding fundamental networking concepts and general tasks in OpenShift Container Platform

Last Updated: 2026-01-20

OpenShift Container Platform 4.20 Networking overview

Understanding fundamental networking concepts and general tasks in OpenShift Container Platform

Legal Notice

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

To build resilient and secure applications in OpenShift Container Platform, configure the networking infrastructure for your cluster. Defining reliable pod-to-pod communication and traffic routing rules ensures that every application component functions correctly within the environment.

1.1. CORE NETWORK LAYERS AND COMPONENTS

To build and expose resilient applications in OpenShift Container Platform, configure the pod and service network layers. Defining these foundational layers ensures that your application workloads have a secure environment to run and remain reliably accessible to other services.

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 IP addresses are ephemeral. This means the IP addresses 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.

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 the ClusterIP of the service, OpenShift Container Platform automatically load balances the traffic to one of the healthy pods backing that service. OpenShift Container Platform 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

To ensure reliable communication between applications in OpenShift Container Platform, configure pod-to-pod traffic and service discovery mechanisms. Implementing these mechanisms allows cluster workloads to exchange data efficiently through either direct connections or robust discovery rules.

Pod-to-pod communication

Pods communicate directly by 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 the IP address of another pod to establish a connection.

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

To enable external access and securely manage traffic flow into and out of your OpenShift Container Platform cluster, configure ingress and egress mechanisms. Establishing these traffic rules ensures that external users can reach your applications reliably while maintaining secure communication with external services.

Exposing applications with Ingress and Route objects

To allow external traffic to reach services inside your cluster, you use an Ingress Controller. The Ingress Controller 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.

Distributing traffic with load balancers

A load balancer provides a single, highly available IP address for directing traffic to your cluster. A load balancer typically runs outside the cluster on a cloud provider or can use MetalLB on bare-metal infrastructure to distribute 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.

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. Consider this configuration when you need to access an external service like a database that has a firewall where you need to allow specific source IPs.
- Egress Router: This is a dedicated pod that acts as a gateway for outbound traffic. By using an Egress Router, you can route connections through a single, controlled exit point.
- Egress Firewall: This acts as a cluster-level firewall for all outbound traffic. The Egress Firewall enhances your security posture so that you can 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

To establish secure administrative access to OpenShift Container Platform instances and control plane nodes, create a bastion host.

Configuring a bastion host provides an entry point for Secure Shell (SSH) traffic, ensuring that your cluster remains protected while allowing for remote management.

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

To establish Secure Shell (SSH) access to OpenShift Container Platform hosts on Amazon EC2 instances that lack public IP addresses, configure a bastion host or secure gateway. Defining this access path ensures that you can safely manage and troubleshoot your private infrastructure within an installer-provisioned environment.

Procedure

1. Create a security group that allows SSH access into the virtual private cloud (VPC) that the **openshift-install** command-line interface creates.

2. Create an Amazon EC2 instance on one of the public subnets the installation program created.

3. Associate a public IP address with the Amazon EC2 instance that you created.
   Unlike with the OpenShift Container Platform installation, associate the Amazon EC2 instance you created with an SSH keypair. The operating system selection is not important for this instance, because the instanace serves as an SSH bastion to bridge the internet into the VPC of your OpenShift Container Platform cluster. The Amazon Machine Image (AMI) you use does matter. With Red Hat Enterprise Linux CoreOS (RHCOS), for example, you can provide keys through Ignition by using a similar method to the installation program.

4. After you provisioned your Amazon EC2 instance and can SSH into the instance, 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 this is not a strict requirement.

   > **NOTE**
   >
   > Use direct SSH access only 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 control plane. The hostname looks similar to **ip-10-0-1-163.ec2.internal**.

6. From the bastion SSH host that you manually deployed into Amazon EC2, SSH into that control plane host by entering the following command. Ensure that you use the same SSH key that you specified during installation:

   ```
   $ ssh -i <ssh-key-path> core@<control_plane_hostname>
   ```

CHAPTER 3. NETWORKING DASHBOARDS

To monitor and analyze network performance within your cluster, view networking metrics in the OpenShift Container Platform web console. By accessing these dashboards through **Observe → Dashboards**, you can identify traffic patterns and troubleshoot connectivity issues to ensure consistent workload availability.

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.

Networking and OVN-Kubernetes dashboard

You can view both general networking metrics and 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**.

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

To ensure stable and accurate network routing in OpenShift Container Platform clusters that use OVN-Kubernetes, define non-overlapping Classless Inter-Domain Routing (CIDR) subnet ranges. Establishing unique ranges prevents IP address conflicts so that internal traffic reaches its intended destination without interference.

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. You must avoid these ranges. For upgraded clusters, there is no change to the default masquerade subnet.

TIP

You can use the Red Hat OpenShift Network Calculator to decide your networking needs before setting CIDR range during cluster creation.

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

The following subnet types 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 postinstallation 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 earlier 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

- Configuring OVN-Kubernetes internal IP address subnets

4.1. MACHINE CIDR

To establish the network scope for cluster nodes in OpenShift Container Platform, specify an IP address range in the Machine Classless Inter-Domain Routing (CIDR) parameter. Defining this range ensures that all machines within the environment have valid, routable addresses for internal cluster communication.

NOTE

You cannot change Machine CIDR ranges after you create 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

To allocate IP addresses for cluster services in OpenShift Container Platform, specify an IP address range in the Service Classless Inter-Domain Routing (CIDR) parameter. Defining this range ensures that internal services have a dedicated block of addresses for reliable communication without overlapping with node or pod networks.

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

To allocate internal network addresses for cluster workloads in OpenShift Container Platform, specify an IP address range in the pod Classless Inter-Domain Routing (CIDR) field. Defining this range ensures that pods can communicate with each other reliably without overlapping with the node or service networks.

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

To allocate a dedicated pool of IP addresses for pods on each node in OpenShift Container Platform, specify the subnet prefix length in the hostPrefix parameter. Defining an appropriate prefix ensures that every machine has sufficient unique addresses to support its scheduled workloads without exhausting the cluster's network resources.

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.

Consider another example where you set the **clusterNetwork.cidr** parameter to **10.128.0.0/16**, you define the complete address space for the cluster. This assigns a pool of 65,536 IP addresses to your cluster. If you then set the **hostPrefix** parameter to **/23**, you define a subnet slice to each node in the cluster, where the **/23** slice becomes a subnet of the **/16** subnet network. This assigns 512 IP addresses to each node, where 2 IP addresses get reserved for networking and broadcasting purposes. The following example calculation uses these IP address figures to determine the maximum number of nodes that you can create for your cluster:

```
65536 / 512 = 128
```

You can use the Red Hat OpenShift Network Calculator to calculate the maximum number of nodes for your cluster.

4.5. CIDR RANGES FOR HOSTED CONTROL PLANES

To successfully deploy hosted control planes on OpenShift Container Platform, define the network environment by using specific Classless Inter-Domain Routing (CIDR) subnet ranges. Establishing these nonoverlapping ranges ensures reliable communication between cluster components and prevents internal IP address conflicts.

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