



OpenShift Container Platform 4.1

Installing

Installing OpenShift Container Platform 4.1 clusters

OpenShift Container Platform 4.1 Installing

Installing OpenShift Container Platform 4.1 clusters

Legal Notice

Copyright © 2019 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 instructions for installing and uninstalling OpenShift Container Platform 4.1 clusters on all supported infrastructure types.

Table of Contents

CHAPTER 1. INSTALLING ON AWS	5
1.1. CONFIGURING AN AWS ACCOUNT	5
1.1.1. Configuring Route53	5
1.1.2. AWS account limits	5
1.1.3. Required AWS permissions	7
1.1.4. Creating an IAM user	13
1.1.5. Supported AWS regions	14
1.2. INSTALLING A CLUSTER QUICKLY ON AWS	14
1.2.1. Internet and Telemetry access for OpenShift Container Platform	15
1.2.2. Generating an SSH private key and adding it to the agent	15
1.2.3. Obtaining the installation program	16
1.2.4. Deploy the cluster	17
1.2.5. Installing the OpenShift Command-line Interface	18
1.2.6. Logging into the cluster	19
1.3. INSTALLING A CLUSTER ON AWS WITH CUSTOMIZATIONS	19
1.3.1. Internet and Telemetry access for OpenShift Container Platform	20
1.3.2. Generating an SSH private key and adding it to the agent	20
1.3.3. Obtaining the installation program	21
1.3.4. Creating the installation configuration file	22
1.3.4.1. Installation configuration parameters	23
1.3.4.2. Sample customized install-config.yaml file for AWS	27
1.3.5. Deploy the cluster	28
1.3.6. Installation command options	30
1.3.7. Installing the OpenShift Command-line Interface	31
1.3.8. Logging into the cluster	31
1.4. INSTALLING A CLUSTER ON AWS WITH NETWORK CUSTOMIZATIONS	32
1.4.1. Internet and Telemetry access for OpenShift Container Platform	32
1.4.2. Generating an SSH private key and adding it to the agent	33
1.4.3. Obtaining the installation program	33
1.4.4. Creating the installation configuration file	34
1.4.4.1. Installation configuration parameters	35
1.4.4.2. Network configuration parameters	39
1.4.4.3. Sample customized install-config.yaml file for AWS	39
1.4.5. Modifying advanced network configuration parameters	41
1.4.6. Cluster Network Operator custom resource (CR)	42
1.4.7. Deploy the cluster	43
1.4.8. Installing the OpenShift Command-line Interface	45
1.4.9. Logging into the cluster	45
1.5. UNINSTALLING A CLUSTER ON AWS	46
1.5.1. Removing a cluster from AWS	46
CHAPTER 2. INSTALLING ON USER-PROVISIONED AWS	48
2.1. INSTALLING A CLUSTER ON AWS USING CLOUDFORMATION TEMPLATES	48
2.1.1. Internet and Telemetry access for OpenShift Container Platform	48
2.1.2. Required AWS infrastructure components	49
2.1.2.1. Cluster machines	49
2.1.2.2. Other infrastructure components	50
2.1.2.3. Required AWS permissions	56
2.1.3. Obtaining the installation program	61
2.1.4. Generating an SSH private key and adding it to the agent	62
2.1.5. Creating the installation files for AWS	63

2.1.6. Extracting the infrastructure name	65
2.1.7. Creating a VPC in AWS	66
2.1.7.1. CloudFormation template for the VPC	68
2.1.8. Creating networking and load balancing components in AWS	75
2.1.8.1. CloudFormation template for the network and load balancers	78
2.1.9. Creating security group and roles in AWS	85
2.1.9.1. CloudFormation template for security objects	87
2.1.10. RHCOS AMIs for the AWS infrastructure	93
2.1.11. Creating the bootstrap node in AWS	94
2.1.11.1. CloudFormation template for the bootstrap machine	98
2.1.12. Creating the control plane machines in AWS	102
2.1.12.1. CloudFormation template for control plane machines	107
2.1.13. Initializing the bootstrap node on AWS with user-provisioned infrastructure	114
2.1.13.1. Creating the worker nodes in AWS	115
2.1.13.1.1. CloudFormation template for worker machines	118
2.1.14. Installing the OpenShift Command-line Interface	121
2.1.15. Logging into the cluster	121
2.1.16. Approving the CSRs for your machines	122
2.1.17. Initial Operator configuration	123
2.1.17.1. Image registry storage configuration	124
2.1.17.1.1. Configuring registry storage for AWS with user-provisioned infrastructure	124
2.1.17.1.2. Configuring storage for the image registry in non-production clusters	125
2.1.18. Completing an AWS installation on user-provisioned infrastructure	125
CHAPTER 3. INSTALLING ON BARE METAL	127
3.1. INSTALLING A CLUSTER ON BARE METAL	127
3.1.1. Internet and Telemetry access for OpenShift Container Platform	127
3.1.2. Machine requirements for a cluster with user-provisioned infrastructure	127
3.1.2.1. Required machines	128
3.1.2.2. Network connectivity requirements	128
3.1.2.3. Minimum resource requirements	128
3.1.2.4. Certificate signing requests management	128
3.1.3. Creating the user-provisioned infrastructure	129
3.1.3.1. Networking requirements for user-provisioned infrastructure	129
Network topology requirements	130
3.1.3.2. User-provisioned DNS requirements	130
3.1.4. Generating an SSH private key and adding it to the agent	133
3.1.5. Obtaining the installation program	134
3.1.6. Installing the OpenShift Command-line Interface	134
3.1.7. Manually creating the installation configuration file	135
3.1.7.1. Sample install-config.yaml file for bare metal	135
3.1.8. Creating the Ignition config files	137
3.1.9. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines	138
3.1.9.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image	138
3.1.9.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE or iPXE booting	139
3.1.10. Creating the cluster	142
3.1.11. Logging into the cluster	142
3.1.12. Approving the CSRs for your machines	143
3.1.13. Initial Operator configuration	144
3.1.13.1. Image registry storage configuration	145
3.1.13.1.1. Configuring registry storage for bare metal	145
3.1.13.1.2. Configuring storage for the image registry in non-production clusters	146
3.1.14. Completing installation on user-provisioned infrastructure	147

CHAPTER 4. INSTALLING ON VSPHERE	150
4.1. INSTALLING A CLUSTER ON VSPHERE	150
4.1.1. Internet and Telemetry access for OpenShift Container Platform	150
4.1.2. VMware vSphere infrastructure requirements	150
4.1.3. Machine requirements for a cluster with user-provisioned infrastructure	151
4.1.3.1. Required machines	151
4.1.3.2. Network connectivity requirements	151
4.1.3.3. Minimum resource requirements	151
4.1.3.4. Certificate signing requests management	152
4.1.4. Creating the user-provisioned infrastructure	152
4.1.4.1. Networking requirements for user-provisioned infrastructure	152
Network topology requirements	153
4.1.4.2. User-provisioned DNS requirements	154
4.1.5. Generating an SSH private key and adding it to the agent	156
4.1.6. Obtaining the installation program	157
4.1.7. Manually creating the installation configuration file	158
4.1.7.1. Sample install-config.yaml file for VMware vSphere	159
4.1.8. Creating the Ignition config files	160
4.1.9. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines in vSphere	161
4.1.10. Installing the OpenShift Command-line Interface	164
4.1.11. Creating the cluster	164
4.1.12. Logging into the cluster	165
4.1.13. Approving the CSRs for your machines	166
4.1.14. Initial Operator configuration	167
4.1.14.1. Image registry storage configuration	168
4.1.14.1.1. Configuring registry storage for VMware vSphere	168
4.1.14.1.2. Configuring storage for the image registry in non-production clusters	169
4.1.15. Completing installation on user-provisioned infrastructure	170
CHAPTER 5. GATHERING INSTALLATION LOGS	172
5.1. GATHERING LOGS FROM A FAILED INSTALLATION	172
CHAPTER 6. INSTALLATION CONFIGURATION	174
6.1. AVAILABLE CLUSTER CUSTOMIZATIONS	174
6.1.1. Cluster configuration resources	174
6.1.2. Operator configuration resources	175
6.1.3. Additional configuration resources	175
6.1.4. Informational Resources	175
6.2. CONFIGURING YOUR FIREWALL	176
6.2.1. Configuring your firewall for OpenShift Container Platform	176

CHAPTER 1. INSTALLING ON AWS

1.1. CONFIGURING AN AWS ACCOUNT

Before you can install OpenShift Container Platform, you must configure an Amazon Web Services (AWS) account.

1.1.1. Configuring Route53

To install OpenShift Container Platform, the Amazon Web Services (AWS) account you use must have a dedicated public hosted zone in your Route53 service. This zone must be authoritative for the domain. The Route53 service provides cluster DNS resolution and name lookup for external connections to the cluster.

Procedure

1. Identify your domain, or subdomain, and registrar. You can transfer an existing domain and registrar or obtain a new one through AWS or another source.



NOTE


If you purchase a new domain through AWS, it takes time for the relevant DNS changes to propagate. For more information about purchasing domains through AWS, see [Registering Domain Names Using Amazon Route 53](#) in the AWS documentation.

2. If you are using an existing domain and registrar, migrate its DNS to AWS. See [Making Amazon Route 53 the DNS Service for an Existing Domain](#) in the AWS documentation.
3. Create a public hosted zone for your domain or subdomain. See [Creating a Public Hosted Zone](#) in the AWS documentation.
Use an appropriate root domain, such as **openshiftcorp.com**, or subdomain, such as **clusters.openshiftcorp.com**.
4. Extract the new authoritative name servers from the hosted zone records. See [Getting the Name Servers for a Public Hosted Zone](#) in the AWS documentation.
5. Update the registrar records for the AWS Route53 name servers that your domain uses. For example, if you registered your domain to a Route53 service in a different accounts, see the following topic in the AWS documentation: [Adding or Changing Name Servers or Glue Records](#).
6. If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain.

1.1.2. AWS account limits

The OpenShift Container Platform cluster uses a number of Amazon Web Services (AWS) components, and the default [Service Limits](#) affect your ability to install OpenShift Container Platform clusters. If you use certain cluster configurations, deploy your cluster in certain AWS regions, or run multiple clusters from your account, you might need to request additional resources for you AWS account.

The following table summarizes the AWS components whose limits can impact your ability to install and run OpenShift Container Platform clusters.

Component	Number of clusters available by default	Default AWS limit	Description
Instance Limits	Varies	Varies	<p>By default, each cluster creates the following instances:</p> <ul style="list-style-type: none"> • One bootstrap machine, which is removed after installation • Three master nodes • Three worker nodes <p>These instance type counts are within a new account's default limit. To deploy more worker nodes, enable autoscaling, deploy large workloads, or use a different instance type, review your account limits to ensure that your cluster can deploy the machines that you need.</p> <p>In most regions, the bootstrap and worker machines use an m4.large machines and the master machines use m4.xlarge instances. In some regions, including all regions that do not support these instance types, m5.large and m5.xlarge instances are used instead.</p>
Elastic IPs (EIPs)	0 to 1	5 EIPs per account	<p>To provision the cluster in a highly available configuration, the installation program creates a public and private subnet for each availability zone within a region. Each private subnet requires a NAT Gateway, and each NAT gateway requires a separate elastic IP. Review the AWS region map to determine how many availability zones are in each region. You can install a single cluster in many regions without increasing your EIP limit, but to take advantage of the default high availability, install the cluster in a region with at least three availability zones.</p> <div>  <p>IMPORTANT</p> <p>To use the us-east-1 region, you must increase the EIP limit for your account.</p> </div>
Virtual Private Clouds (VPCs)	5	5 VPCs per region	Each cluster creates its own VPC.

Component	Number of clusters available by default	Default AWS limit	Description
Elastic Load Balancing (ELB/NLB)	3	20 per region	By default, each cluster creates an internal and external network load balancers for the master API server and a single classic elastic load balancer for the router. Deploying more Kubernetes LoadBalancer Service objects will create additional load balancers .
NAT Gateways	5	5 per availability zone	The cluster deploys one NAT gateway in each availability zone.
Elastic Network Interfaces (ENIs)	At least 12	350 per region	The default installation creates 21 ENIs and an ENI for each availability zone in your region. For example, the us-east-1 region contains six availability zones, so a cluster that is deployed in that zone uses 27 ENIs. Review the AWS region map to determine how many availability zones are in each region. Additional ENIs are created for additional machines and elastic load balancers that are created by cluster usage and deployed workloads.
VPC Gateway	20	20 per account	Your AWS account uses VPC Gateways for S3 access. Each cluster creates a single VPC Gateway for S3 access.
S3 buckets	99	100 buckets per account	Because the installation process creates a temporary bucket and the registry component in each cluster creates a bucket, you can create only 99 OpenShift Container Platform clusters per AWS account.
Security Groups	250	2,500 per account	Each cluster creates 10 distinct security groups.

1.1.3. Required AWS permissions

When you attach the **AdministratorAccess** policy to the IAM user that you create, you grant that user all of the required permissions. To deploy an OpenShift Container Platform cluster, the IAM user requires the following permissions:

Required EC2 permissions for installation

- **ec2:AllocateAddress**
- **ec2:AssociateAddress**

- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**
- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2:CreateDhcpOptions**
- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**
- **ec2:CreateSecurityGroup**
- **ec2:CreateSubnet**
- **ec2:CreateTags**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:CreateVolume**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInternetGateways**
- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**
- **ec2:DescribeNetworkAcls**

- **ec2:DescribePrefixLists**
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**
- **ec2:DescribeVpcAttribute**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcClassicLink**
- **ec2:DescribeVpcClassicLinkDnsSupport**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**
- **ec2:RevokeSecurityGroupEgress**
- **ec2:RunInstances**
- **ec2:TerminateInstances**
- **ec2:RevokeSecurityGroupIngress**
- **ec2:ReplaceRouteTableAssociation**
- **ec2:DescribeNetworkInterfaces**
- **ec2:ModifyNetworkInterfaceAttribute**

Required Elasticloadbalancing permissions for installation

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:CreateListener**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateLoadBalancerListeners**

- **elasticloadbalancing:CreateTargetGroup**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:DescribeInstanceHealth**
- **elasticloadbalancing:DescribeListeners**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeTags**
- **elasticloadbalancing:DescribeTargetGroupAttributes**
- **elasticloadbalancing:DescribeTargetHealth**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterTargets**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**

Required IAM permissions for installation

- **iam:AddRoleToInstanceProfile**
- **iam:CreateInstanceProfile**
- **iam:CreateRole**
- **iam:DeleteInstanceProfile**
- **iam>DeleteRole**
- **iam>DeleteRolePolicy**
- **iam:GetInstanceProfile**
- **iam:GetRole**
- **iam:GetRolePolicy**
- **iam:GetUser**
- **iam:ListInstanceProfilesForRole**

- **iam:ListRoles**
- **iam:ListUsers**
- **iam:PassRole**
- **iam:PutRolePolicy**
- **iam:RemoveRoleFromInstanceProfile**
- **iam:SimulatePrincipalPolicy**
- **iam:TagRole**

Required Route53 permissions for installation

- **route53:ChangeResourceRecordSets**
- **route53:ChangeTagsForResource**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:CreateHostedZone**
- **route53:ListHostedZones**
- **route53:ListHostedZonesByName**
- **route53:ListResourceRecordSets**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**

Required S3 permissions for installation

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketCors**
- **s3:GetBucketLocation**
- **s3:GetBucketLogging**
- **s3:GetBucketObjectLockConfiguration**
- **s3:GetBucketReplication**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketTagging**

- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **s3:GetEncryptionConfiguration**
- **s3:GetLifecycleConfiguration**
- **s3:GetReplicationConfiguration**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

S3 permissions that cluster Operators require

- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:GetObjectVersion**
- **s3:DeleteObject**

All additional permissions that are required to uninstall a cluster

- **autoscaling:DescribeAutoScalingGroups**
- **ec2:DeleteDhcpOptions**
- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteNetworkInterface**
- **ec2:DeleteRoute**
- **ec2:DeleteRouteTable**
- **ec2:DeleteSnapshot**
- **ec2:DeleteSecurityGroup**
- **ec2:DeleteSubnet**

- `ec2:DeleteVolume`
- `ec2:DeleteVpc`
- `ec2:DeleteVpcEndpoints`
- `ec2:DeregisterImage`
- `ec2:DetachInternetGateway`
- `ec2:DisassociateRouteTable`
- `ec2:ReleaseAddress`
- `elasticloadbalancing:DescribeTargetGroups`
- `elasticloadbalancing>DeleteTargetGroup`
- `elasticloadbalancing>DeleteLoadBalancer`
- `iam:ListInstanceProfiles`
- `iam:ListRolePolicies`
- `iam:ListUserPolicies`
- `route53>DeleteHostedZone`
- `tag:GetResources`

1.1.4. Creating an IAM user

Each Amazon Web Services (AWS) account contains a root user account that is based on the email address you used to create the account. This is a highly-privileged account, and it is recommended to use it for only initial account and billing configuration, creating an initial set of users, and securing the account.

Before you install OpenShift Container Platform, create a secondary IAM administrative user. As you complete the [Creating an IAM User in Your AWS Account](#) procedure in the AWS documentation, set the following options:

Procedure

1. Specify the IAM user name and select **Programmatic access**.
2. Attach the **AdministratorAccess** policy to ensure that the account has sufficient permission to create the cluster. This policy provides the cluster with the ability to grant credentials to each OpenShift Container Platform component. The cluster grants the components only the credentials that they require.



NOTE

While it is possible to create a policy that grants the all of the required AWS permissions and attach it to the user, this is not the preferred option. The cluster will not have the ability to grant additional credentials to individual components, so the same credentials are used by all components.

3. Optional: Add metadata to the user by attaching tags.
4. Confirm that the user name that you specified is granted the **AdministratorAccess** policy.
5. Record the access key ID and secret access key values. You must use these values when you configure your local machine to run the installation program.



IMPORTANT

You cannot use a temporary session token that you generated while using a multi-factor authentication device to authenticate to AWS when you deploy a cluster. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use key-based, long-lived credentials.

1.1.5. Supported AWS regions

You can deploy an OpenShift Container Platform cluster to the following regions:

- ap-northeast-1 (Tokyo)
- ap-northeast-2 (Seoul)
- ap-south-1 (Mumbai)
- ap-southeast-1 (Singapore)
- ap-southeast-2 (Sydney)
- ca-central-1 (Central)
- eu-central-1 (Frankfurt)
- eu-west-1 (Ireland)
- eu-west-2 (London)
- eu-west-3 (Paris)
- sa-east-1 (São Paulo)
- us-east-1 (N. Virginia)
- us-east-2 (Ohio)
- us-west-1 (N. California)
- us-west-2 (Oregon)

Next steps

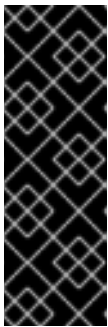
- Install an OpenShift Container Platform cluster. You can [install a customized cluster](#) or [quickly install a cluster](#) with default options.

1.2. INSTALLING A CLUSTER QUICKLY ON AWS

In OpenShift Container Platform version 4.1, you can install a cluster on Amazon Web Services (AWS) that uses the default configuration options.

Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure an AWS account](#) to host the cluster.



IMPORTANT

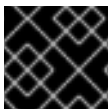
If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use key-based, long-lived credentials. To generate appropriate keys, see [Managing Access Keys for IAM Users](#) in the AWS documentation. You can supply the keys when you run the installation program.

- If you use a firewall, you must [configure it to access Red Hat Insights](#) .

1.2.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.1, Telemetry is the component that provides metrics about cluster health and the success of updates. To perform subscription management, including legally entitling your purchase from Red Hat, you must use the Telemetry service and access the [OpenShift Infrastructure Providers](#) page.

Because there is no disconnected subscription management, you cannot both opt out of sending data back to Red Hat and entitle your purchase. Support for disconnected subscription management might be added in future releases of OpenShift Container Platform



IMPORTANT

Your machines must have direct internet access to install the cluster.

You must have internet access to:

- Access the [OpenShift Infrastructure Providers](#) page to download the installation program
- Access [quay.io](#) to obtain the packages that are required to install your cluster
- Obtain the packages that are required to perform cluster updates
- Access [Red Hat's software as a service page](#) to perform subscription management

1.2.2. Generating an SSH private key and adding it to the agent

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, you must provide an SSH key that your **ssh-agent** process uses to the installer.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.

**NOTE**

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t rsa -b 4096 -N "" \
  -f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the SSH key.

Running this command generates an SSH key that does not require a password in the location that you specified.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
Agent pid 31874
```

3. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

When you install OpenShift Container Platform, provide the SSH public key to the installer. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

1.2.3. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 300 MB of local disk space to download the installation program.

Procedure

1. Access the [OpenShift Infrastructure Providers](#) page. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

2. Download the installation program for your operating system and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.

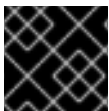
3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [OpenShift Infrastructure Providers](#) page, download your installation pull secret. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

1.2.4. Deploy the cluster

You can install OpenShift Container Platform on a compatible cloud.



IMPORTANT

You can run the installation program only once, during initial installation.

Prerequisites

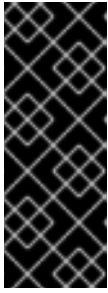
- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ ❶
--log-level debug ❷
```

- ❶ For **<installation_directory>**, specify the directory name to store the files that the installation program creates.
- ❷ Optionally, include the **--log-level debug** option to view installation details.

**IMPORTANT**

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

Provide values at the prompts:

- a. Optional: Select an SSH key to use to access your cluster machines.

**NOTE**

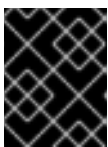
For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, you must provide an SSH key that your **ssh-agent** process uses to the installation program.

- b. Select AWS as the platform to target.
- c. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.
- d. Select the AWS region to deploy the cluster to.
- e. Select the base domain for the Route53 service that you configured for your cluster.
- f. Enter a descriptive name for your cluster.
- g. Paste the pull secret that you obtained from the [OpenShift Infrastructure Providers](#) page.

**NOTE**

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.

**IMPORTANT**

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

2. Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

1.2.5. Installing the OpenShift Command-line Interface

You can download and install the OpenShift Command-line Interface (CLI), commonly known as **oc**.

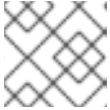


IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.1. You must download and install the new version of **oc**.

Procedure

1. From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**
2. From the site that is displayed, download the compressed file for your operating system.



NOTE

You can install **oc** on Linux, Windows, or macOS.

3. Extract the compressed file and place it in a directory that is on your PATH.

1.2.6. Logging into the cluster

You can log into your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
$ oc whoami
system:admin
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of telemetry](#).

1.3. INSTALLING A CLUSTER ON AWS WITH CUSTOMIZATIONS

In OpenShift Container Platform version 4.1, you can install a customized cluster on infrastructure that the installation program provisions on Amazon Web Services (AWS). To customize the installation, you modify some parameters in the **install-config.yaml** file before you install the cluster.

Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure an AWS account](#) to host the cluster.



IMPORTANT

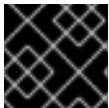
If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use long-lived credentials. To generate appropriate keys, see [Managing Access Keys for IAM Users](#) in the AWS documentation. You can supply the keys when you run the installation program.

- If you use a firewall, you must [configure it to access Red Hat Insights](#) .

1.3.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.1, Telemetry is the component that provides metrics about cluster health and the success of updates. To perform subscription management, including legally entitling your purchase from Red Hat, you must use the Telemetry service and access the [OpenShift Infrastructure Providers](#) page.

Because there is no disconnected subscription management, you cannot both opt out of sending data back to Red Hat and entitle your purchase. Support for disconnected subscription management might be added in future releases of OpenShift Container Platform



IMPORTANT

Your machines must have direct internet access to install the cluster.

You must have internet access to:

- Access the [OpenShift Infrastructure Providers](#) page to download the installation program
- Access [quay.io](#) to obtain the packages that are required to install your cluster
- Obtain the packages that are required to perform cluster updates
- Access [Red Hat's software as a service page](#) to perform subscription management

1.3.2. Generating an SSH private key and adding it to the agent

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, you must provide an SSH key that your **ssh-agent** process uses to the installer.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t rsa -b 4096 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the SSH key.

Running this command generates an SSH key that does not require a password in the location that you specified.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
Agent pid 31874
```

3. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

When you install OpenShift Container Platform, provide the SSH public key to the installer. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

1.3.3. Obtaining the installation program

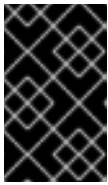
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 300 MB of local disk space to download the installation program.

Procedure

1. Access the [OpenShift Infrastructure Providers](#) page. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Download the installation program for your operating system and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [OpenShift Infrastructure Providers](#) page, download your installation pull secret. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

1.3.4. Creating the installation configuration file

You can customize your installation of OpenShift Container Platform on a compatible cloud.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

1

For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

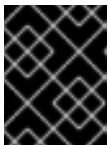
Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.

**NOTE**

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, you must provide an SSH key that your **ssh-agent** process uses to the installation program.

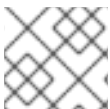
- ii. Select **AWS** as the platform to target.
 - iii. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.
 - iv. Select the AWS region to deploy the cluster to.
 - v. Select the base domain for the Route53 service that you configured for your cluster.
 - vi. Enter a descriptive name for your cluster.
 - vii. Paste the pull secret that you obtained from the [OpenShift Infrastructure Providers](#) page.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section and in the [Go documentation](#).
3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

**IMPORTANT**

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

1.3.4.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your Amazon Web Services (AWS) account and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.

**NOTE**



You cannot modify these parameters after installation.


Table 1.1. Required parameters

Parameter	Description	Values
-----------	-------------	--------

Parameter	Description	Values
baseDomain	The base domain of your cloud provider. This value is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
controlPlane.platform	The cloud provider to host the control plane machines. This parameter value must match the compute.platform parameter value.	aws
compute.platform	The cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws
metadata.name	The name of your cluster.	A string that contains uppercase or lowercase letters, such as dev .
platform.aws.region	The region to deploy your cluster in.	A valid AWS region, such as us-east-1 .
pullSecret	The pull secret that you obtained from the OpenShift Infrastructure Providers page. You use this pull secret to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>

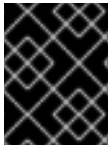
Table 1.2. Optional AWS platform parameters

Parameter	Description	Values
sshKey	<p>The SSH key to use to access your cluster machines.</p> <div>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, you must provide an SSH key that your ssh-agent process uses to the installation program.</p> </div>	A valid, local public SSH key that you added to the ssh-agent process.
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div>	Enabled or Disabled
compute.platform.aws.rootVolume.iops	The Input/Output Operations Per Second (IOPS) that is reserved for the root volume.	Integer, for example 4000 .
compute.platform.aws.rootVolume.size	The size in GiB of the root volume.	Integer, for example 500 .
compute.platform.aws.rootVolume.type	The instance type of the root volume.	Valid AWS EBS instance type , such as io1 .
compute.platform.aws.type	The EC2 instance type for the compute machines.	Valid AWS instance type , such as c5.9xlarge .

Parameter	Description	Values
compute.platform.aws.zones	The availability zones where the installation program creates machines for the compute MachinePool.	A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence .
compute.aws.region	The AWS region that the installation program creates compute resources in.	Valid AWS region , such as us-east-1 .
compute.replicas	The number of compute, or worker, machines to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane.hypertreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div>	Enabled or Disabled
controlPlane.platform.aws.type	The EC2 instance type for the control plane machines.	Valid AWS instance type , such as c5.9xlarge .
controlPlane.platform.aws.zones	The availability zones where the installation program creates machines for the control plane MachinePool.	A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence .
controlPlane.aws.region	The AWS region that the installation program creates control plane resources in.	Valid AWS region , such as us-east-1 .
controlPlane.replicas	The number of control plane machines to provision.	A positive integer greater than or equal to 3 . The default value is 3 .
platform.aws.userTags	A map of keys and values that the installation program adds as tags to all resources that it creates.	Any valid YAML map, such as key value pairs in the <key>: <value> format. For more information about AWS tags, see Tagging Your Amazon EC2 Resources in the AWS documentation.

1.3.4.2. Sample customized install-config.yaml file for AWS

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.



IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it.

```

apiVersion: v1
baseDomain: example.com ❶
controlPlane: ❷
  hyperthreading: Enabled ❸ ❹
  name: master
  platform:
    aws:
      zones:
        - us-west-2a
        - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1
        type: m5.xlarge ❺
      replicas: 3
compute: ❻
  - hyperthreading: Enabled ❼
    name: worker
    platform:
      aws:
        rootVolume:
          iops: 2000
          size: 500
          type: io1 ❽
          type: c5.4xlarge
        zones:
          - us-west-2c
        replicas: 3
metadata:
  name: test-cluster ❾
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  machineCIDR: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
    - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 ❿
    userTags:
      adminContact: jdoe

```

```
costCenter: 7536
pullSecret: '{"auths": ...}' 11
sshKey: ssh-ed25519 AAAA... 12
```

1 9 10 11 Required. The installation program prompts you for this value.

2 6 If you do not provide these parameters and values, the installation program provides the default value.

3 7 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

4 5 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.

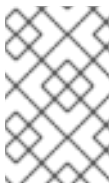


IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger instance types, such as **m4.2xlarge** or **m5.2xlarge**, for your machines if you disable simultaneous multithreading.

8 To configure faster storage for etcd, especially for larger clusters, set the storage type as **io1** and set **iops** to **2000**.

12 You can optionally provide the **sshKey** value that you use to access the machines in your cluster.

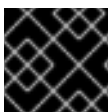


NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, you must provide an SSH key that your **ssh-agent** process uses to the installation program.

1.3.5. Deploy the cluster

You can install OpenShift Container Platform on a compatible cloud.



IMPORTANT

You can run the installation program only once, during initial installation.

Prerequisites

- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level debug 2
```

- 1** For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.
- 2** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.

Optionally, include the **--log-level debug** option to view installation details.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

Provide values at the prompts:

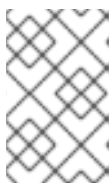
- a. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, you must provide an SSH key that your **ssh-agent** process uses to the installation program.

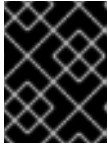
- b. Select AWS as the platform to target.
- c. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.
- d. Select the AWS region to deploy the cluster to.
- e. Select the base domain for the Route53 service that you configured for your cluster.
- f. Enter a descriptive name for your cluster.
- g. Paste the pull secret that you obtained from the [OpenShift Infrastructure Providers](#) page.



NOTE

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

- Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

1.3.6. Installation command options

The installation program uses [Semantic Versioning](#) for its user-facing API to ensure that commands are stable across versions. While most of the **openshift-install** command options comply with semantic versioning practices, some do not.

Commands that comply with semantic versioning

The following commands are covered by the versioning:

openshift-install [options] create install-config

Creates the **install-config.yaml** file in the asset directory. The version of the generated **install-config** might change.

openshift-install [options] create ignition-configs

Creates the **bootstrap.ign**, **master.ign**, and **worker.ign** files in the asset directory,. The content of the generated files might change between versions.

openshift-install [options] create cluster

Launches a new cluster.

openshift-install [options] destroy bootstrap

Destroys any bootstrap resources created for the cluster.

openshift-install [options] destroy cluster

Destroys the cluster resources.

openshift-install [options] help

Shows help for the command. The available options and unstable commands might change between versions

openshift-install [options] version

Displays sufficient version information for maintainers to identify the installation program version. The format and content of its output might change between versions.

The install-config format itself

New versions of this format might be released, but within a minor version series, the **openshift-install** command can read previous versions.

Commands that do not comply with semantic versioning

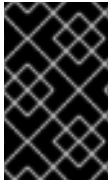
The following commands are not covered by the versioning:

openshift-install [options] graph, **openshift-install [options] create manifest-templates**, **openshift-install [options] create manifests**

If you must alter, add, or remove Kubernetes objects, you must perform those actions after you install the cluster..

1.3.7. Installing the OpenShift Command-line Interface

You can download and install the OpenShift Command-line Interface (CLI), commonly known as **oc**.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.1. You must download and install the new version of **oc**.

Procedure

1. From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**
2. From the site that is displayed, download the compressed file for your operating system.



NOTE

You can install **oc** on Linux, Windows, or macOS.

3. Extract the compressed file and place it in a directory that is on your PATH.

1.3.8. Logging into the cluster

You can log into your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
$ oc whoami
system:admin
```

1

For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of telemetry](#).

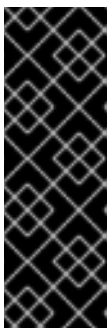
1.4. INSTALLING A CLUSTER ON AWS WITH NETWORK CUSTOMIZATIONS

In OpenShift Container Platform version 4.1, you can install a cluster on Amazon Web Services (AWS) with customized network configuration options. 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.

You must set most of the network configuration parameters during installation, and you can modify only **kubeProxy** configuration parameters in a running cluster.

Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure an AWS account](#) to host the cluster.



IMPORTANT

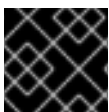
If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use key-based, long-lived credentials. To generate appropriate keys, see [Managing Access Keys for IAM Users](#) in the AWS documentation. You can supply the keys when you run the installation program.

- If you use a firewall, you must [configure it to access Red Hat Insights](#).

1.4.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.1, Telemetry is the component that provides metrics about cluster health and the success of updates. To perform subscription management, including legally entitling your purchase from Red Hat, you must use the Telemetry service and access the [OpenShift Infrastructure Providers](#) page.

Because there is no disconnected subscription management, you cannot both opt out of sending data back to Red Hat and entitle your purchase. Support for disconnected subscription management might be added in future releases of OpenShift Container Platform



IMPORTANT

Your machines must have direct internet access to install the cluster.

You must have internet access to:

- Access the [OpenShift Infrastructure Providers](#) page to download the installation program
- Access [quay.io](#) to obtain the packages that are required to install your cluster

- Obtain the packages that are required to perform cluster updates
- Access [Red Hat's software as a service page](#) to perform subscription management

1.4.2. Generating an SSH private key and adding it to the agent

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, you must provide an SSH key that your **ssh-agent** process uses to the installer.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t rsa -b 4096 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the SSH key.

Running this command generates an SSH key that does not require a password in the location that you specified.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
Agent pid 31874
```

3. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

When you install OpenShift Container Platform, provide the SSH public key to the installer. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

1.4.3. Obtaining the installation program

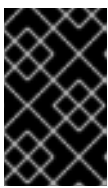
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 300 MB of local disk space to download the installation program.

Procedure

1. Access the [OpenShift Infrastructure Providers](#) page. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Download the installation program for your operating system and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [OpenShift Infrastructure Providers](#) page, download your installation pull secret. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

1.4.4. Creating the installation configuration file

You can customize your installation of OpenShift Container Platform on a compatible cloud.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

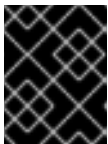
- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, you must provide an SSH key that your **ssh-agent** process uses to the installation program.

- ii. Select **AWS** as the platform to target.
 - iii. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.
 - iv. Select the AWS region to deploy the cluster to.
 - v. Select the base domain for the Route53 service that you configured for your cluster.
 - vi. Enter a descriptive name for your cluster.
 - vii. Paste the pull secret that you obtained from the [OpenShift Infrastructure Providers](#) page.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section and in the [Go documentation](#).
3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

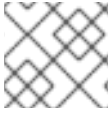


IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

1.4.4.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your Amazon Web Services (AWS) account and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



**NOTE**


You cannot modify these parameters after installation.

Table 1.3. Required parameters

Parameter	Description	Values
baseDomain	The base domain of your cloud provider. This value is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
controlPlane.platform	The cloud provider to host the control plane machines. This parameter value must match the compute.platform parameter value.	aws
compute.platform	The cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws
metadata.name	The name of your cluster.	A string that contains uppercase or lowercase letters, such as dev .
platform.aws.region	The region to deploy your cluster in.	A valid AWS region, such as us-east-1 .
pullSecret	The pull secret that you obtained from the OpenShift Infrastructure Providers page. You use this pull secret to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>

Table 1.4. Optional AWS platform parameters

Parameter	Description	Values
sshKey	<p>The SSH key to use to access your cluster machines.</p> <div>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, you must provide an SSH key that your ssh-agent process uses to the installation program.</p> </div>	A valid, local public SSH key that you added to the ssh-agent process.
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div>	Enabled or Disabled
compute.platform.aws.rootVolume.iops	The Input/Output Operations Per Second (IOPS) that is reserved for the root volume.	Integer, for example 4000 .
compute.platform.aws.rootVolume.size	The size in GiB of the root volume.	Integer, for example 500 .
compute.platform.aws.rootVolume.type	The instance type of the root volume.	Valid AWS EBS instance type , such as io1 .
compute.platform.aws.type	The EC2 instance type for the compute machines.	Valid AWS instance type , such as c5.9xlarge .

Parameter	Description	Values
compute.platform.aws.zones	The availability zones where the installation program creates machines for the compute MachinePool.	A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence .
compute.aws.region	The AWS region that the installation program creates compute resources in.	Valid AWS region , such as us-east-1 .
compute.replicas	The number of compute, or worker, machines to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane.hypert hreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div>	Enabled or Disabled
controlPlane.platform.aws.type	The EC2 instance type for the control plane machines.	Valid AWS instance type , such as c5.9xlarge .
controlPlane.platform.aws.zones	The availability zones where the installation program creates machines for the control plane MachinePool.	A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence .
controlPlane.aws.region	The AWS region that the installation program creates control plane resources in.	Valid AWS region , such as us-east-1 .
controlPlane.replicas	The number of control plane machines to provision.	A positive integer greater than or equal to 3 . The default value is 3 .
platform.aws.userTags	A map of keys and values that the installation program adds as tags to all resources that it creates.	Any valid YAML map, such as key value pairs in the <key>: <value> format. For more information about AWS tags, see Tagging Your Amazon EC2 Resources in the AWS documentation.

1.4.4.2. Network configuration parameters

You can modify your cluster network configuration parameters in the **install-config.yaml** configuration file. The following table describes the parameters.



NOTE

You cannot modify these parameters after installation.

Table 1.5. Required network parameters

Parameter	Description	Values
networking.net workType	The network plug-in to deploy. OpenShiftSDN is the only plug-in supported in OpenShift Container Platform 4.1.	OpenShiftSDN
networking.clus terNetwork.cidr	A block of IP addresses from which Pod IP addresses are allocated. The OpenShiftSDN network plug-in supports multiple cluster networks. The address blocks for multiple cluster networks must not overlap. Select address pools large enough to fit your anticipated workload.	An IP address allocation in CIDR format. The default value is 10.128.0.0/14 .
networking.clus terNetwork.host Prefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 , then each node is assigned a /23 subnet out of the given cidr , allowing for 510 ($2^{(32 - 23)} - 2$) Pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serv iceNetwork	A block of IP addresses for services. OpenShiftSDN allows only one serviceNetwork block. The address block must not overlap with any other network block.	An IP address allocation in CIDR format. The default value is 172.30.0.0/16 .
networking.mac hineCIDR	A block of IP addresses used by the OpenShift Container Platform installation program while installing the cluster. The address block must not overlap with any other network block.	An IP address allocation in CIDR format. The default value is 10.0.0.0/16 .

1.4.4.3. Sample customized install-config.yaml file for AWS

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.



IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it.

apiVersion: v1

baseDomain: example.com **1**

```

controlPlane: 2
  hyperthreading: Enabled 3 4
  name: master
  platform:
    aws:
      zones:
        - us-west-2a
        - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1
        type: m5.xlarge 5
      replicas: 3
compute: 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 8
        type: c5.4xlarge
      zones:
        - us-west-2c
      replicas: 3
  metadata:
    name: test-cluster 9
  networking:
  networking: 10
    clusterNetwork:
      - cidr: 10.128.0.0/14
      hostPrefix: 23
    machineCIDR: 10.0.0.0/16
    networkType: OpenShiftSDN
    serviceNetwork:
      - 172.30.0.0/16
  platform:
    aws:
      region: us-west-2 11
      userTags:
        adminContact: jdoe
        costCenter: 7536
  pullSecret: '{"auths": ...}' 12
  sshKey: ssh-ed25519 AAAA... 13

```

1 9 11 12 Required. The installation program prompts you for this value.

2 6 10 If you do not provide these parameters and values, the installation program provides the default value.

3 7 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both

sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

- 4 5** Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger instance types, such as **m4.2xlarge** or **m5.2xlarge**, for your machines if you disable simultaneous multithreading.

- 8** To configure faster storage for etcd, especially for larger clusters, set the storage type as **io1** and set **iops** to **2000**.
- 13** You can optionally provide the **sshKey** value that you use to access the machines in your cluster.

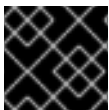


NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, you must provide an SSH key that your **ssh-agent** process uses to the installation program.

1.4.5. Modifying advanced network configuration parameters

You can modify the advanced network configuration parameters only before you install the cluster. Advanced configuration customization lets you integrate your cluster into your existing network environment by specifying an MTU or VXLAN port, by allowing customization of **kube-proxy** settings, and by specifying a different **mode** for the **openshiftSDNConfig** parameter.



IMPORTANT

Modifying the OpenShift Container Platform manifest files directly is not supported.

Prerequisites

- Generate the **install-config.yaml** file and complete any modifications to it.

Procedure

1. Use the following command to create manifests:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the name of the directory that contains the **install-config.yaml** file for your cluster.

2. Create a file that is named **cluster-network-03-config.yml** in the **<installation_directory>/manifests/** directory:

```
$ touch <installation_directory>/manifests/cluster-network-03-config.yml 1
```

- 1** For **<installation_directory>**, specify the directory name that contains the **manifests/** directory for your cluster.

After creating the file, three network configuration files are in the **manifests/** directory, as shown:

```
$ ls <installation_directory>/manifests/cluster-network-*
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-network-03-config.yml
```

3. Open the **cluster-network-03-config.yml** file in an editor and enter a CR that describes the Operator configuration you want:

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec: 1
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
```

- 1** The parameters for the **spec** field are only an example. Specify your configuration for the Network Operator in the CR.

The Network Operator provides default values for the parameters in the CR, so you must specify only the parameters that you want to change in the **Network.operator.openshift.io** CR.

4. Save the **cluster-network-03-config.yml** file and quit the text editor.
5. Optional: Back up the **manifests/cluster-network-03-config.yml** file. The installation program deletes the **manifests/** directory when creating the cluster.

1.4.6. Cluster Network Operator custom resource (CR)

The cluster network configuration in the **Network.operator.openshift.io** custom resource (CR) stores the configuration settings for the Cluster Network Operator (CNO).

The following CR displays the default configuration for the CNO and explains both the parameters you can configure and valid parameter values:

Cluster Network Operator CR

```
apiVersion: operator.openshift.io/v1
```

```

kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork: ❶
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork: ❷
  - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN ❸
    openshiftSDNConfig: ❹
      mode: NetworkPolicy ❺
      mtu: 1450 ❻
      vxlanPort: 4789 ❼
  kubeProxyConfig: ❽
    iptablesSyncPeriod: 30s ❾
    proxyArguments:
      iptables-min-sync-period: ❿
      - 30s

```

❶ ❷ ❸ Specified in the **install-config.yaml** file.

❹ Specify only if you want to override part of the OpenShift Container Platform SDN configuration.

❺ Configures the isolation mode for **OpenShiftSDN**. The allowed values are **Multitenant**, **Subnet**, or **NetworkPolicy**. The default value is **NetworkPolicy**.

❻ MTU for the VXLAN overlay network. This value is normally configured automatically, but if the nodes in your cluster do not all use the same MTU, then you must set this explicitly to 50 less than the smallest node MTU value.

❼ The port to use for all VXLAN packets. The default value is **4789**. If you are running in a virtualized environment with existing nodes that are part of another VXLAN network then you might be required to change this.

On Amazon Web Services (AWS), you can select an alternate port for the VXLAN between port **9000** and port **9999**.

❽ The parameters for this object specify the **kube-proxy** configuration. If you do not specify the parameter values, the Network Operator applies the displayed default parameter values.

❾ The refresh period for **iptables** rules. The default value is **30s**. Valid suffixes include **s**, **m**, and **h** and are described in the [Go time package](#) documentation.

❿ The minimum duration before refreshing **iptables** rules. This parameter ensures that the refresh does not happen too frequently. Valid suffixes include **s**, **m**, and **h** and are described in the [Go time package](#)

1.4.7. Deploy the cluster

You can install OpenShift Container Platform on a compatible cloud.



IMPORTANT

You can run the installation program only once, during initial installation.

Prerequisites

- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

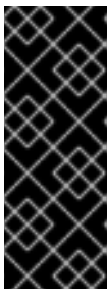
Procedure

1. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level debug 2
```

- 1 For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.
- 2 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.

Optionally, include the **--log-level debug** option to view installation details.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

Provide values at the prompts:

- a. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, you must provide an SSH key that your **ssh-agent** process uses to the installation program.

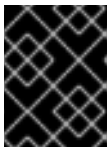
- b. Select AWS as the platform to target.
- c. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.
- d. Select the AWS region to deploy the cluster to.

- e. Select the base domain for the Route53 service that you configured for your cluster.
- f. Enter a descriptive name for your cluster.
- g. Paste the pull secret that you obtained from the [OpenShift Infrastructure Providers](#) page.

**NOTE**

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.

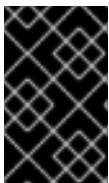
**IMPORTANT**

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

2. Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

1.4.8. Installing the OpenShift Command-line Interface

You can download and install the OpenShift Command-line Interface (CLI), commonly known as **oc**.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.1. You must download and install the new version of **oc**.

Procedure

1. From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**
2. From the site that is displayed, download the compressed file for your operating system.

**NOTE**

You can install **oc** on Linux, Windows, or macOS.

3. Extract the compressed file and place it in a directory that is on your PATH.

1.4.9. Logging into the cluster

You can log into your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1  
  
$ oc whoami  
system:admin
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of telemetry](#).

1.5. UNINSTALLING A CLUSTER ON AWS

You can remove a cluster that you deployed to Amazon Web Services (AWS).

1.5.1. Removing a cluster from AWS

You can remove a cluster that you installed on Amazon Web Services (AWS).

Prerequisites

- Have a copy of the installation program that you used to deploy the cluster.

Procedure

1. Optional: From the computer that you used to install the cluster, run the following command and record the UUID that it outputs:

```
$ oc get clusterversion -o jsonpath='{.spec.clusterID}{"\n"}' version
```

If not all of the cluster resources are removed from AWS, you can use this UUID to locate them and remove them.

2. From the computer that you used to install the cluster, run the following command:

```
$ ./openshift-install destroy cluster \  
--dir=<installation_directory> --log-level=debug 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

**NOTE**

You must specify the directory that contains the cluster definition files for your cluster. The installation program requires the **metadata.json** file in this directory to delete the cluster.

3. Optional: Delete the **<installation_directory>** directory and the OpenShift Container Platform installation program.

CHAPTER 2. INSTALLING ON USER-PROVISIONED AWS

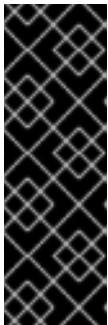
2.1. INSTALLING A CLUSTER ON AWS USING CLOUDFORMATION TEMPLATES

In OpenShift Container Platform version 4.1, you can install a cluster on Amazon Web Services (AWS) using infrastructure that you provide.

One way to create this infrastructure is to use the provided CloudFormation templates. You can modify the templates to customize your infrastructure or use the information that they contain to create AWS objects according to your company's policies.

Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure an AWS account](#) to host the cluster.



IMPORTANT

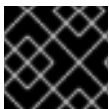
If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use key-based, long-lived credentials. To generate appropriate keys, see [Managing Access Keys for IAM Users](#) in the AWS documentation. You can supply the keys when you run the installation program.

- Download the AWS CLI and install it on your computer. See [Install the AWS CLI Using the Bundled Installer \(Linux, macOS, or Unix\)](#) in the AWS documentation.
- If you use a firewall, you must [configure it to access Red Hat Insights](#) .

2.1.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.1, Telemetry is the component that provides metrics about cluster health and the success of updates. To perform subscription management, including legally entitling your purchase from Red Hat, you must use the Telemetry service and access the [OpenShift Infrastructure Providers](#) page.

Because there is no disconnected subscription management, you cannot both opt out of sending data back to Red Hat and entitle your purchase. Support for disconnected subscription management might be added in future releases of OpenShift Container Platform



IMPORTANT

Your machines must have direct internet access to install the cluster.

You must have internet access to:

- Access the [OpenShift Infrastructure Providers](#) page to download the installation program
- Access [quay.io](#) to obtain the packages that are required to install your cluster

- Obtain the packages that are required to perform cluster updates
- Access [Red Hat's software as a service page](#) to perform subscription management

2.1.2. Required AWS infrastructure components

To install OpenShift Container Platform on user-provisioned infrastructure in Amazon Web Services (AWS), you must manually create both the machines and their supporting infrastructure.

For more information about the integration testing for different platforms, see the [OpenShift Container Platform 4.x Tested Integrations](#) page.

You can use the provided CloudFormation templates to create this infrastructure, you can manually create the components, or you can reuse existing infrastructure that meets the cluster requirements. Review the CloudFormation templates for more details about how the components interrelate.

2.1.2.1. Cluster machines

You need **AWS::EC2::Instance** objects for the following machines:

- A bootstrap machine. This machine is required during installation, but you can remove it after your cluster deploys.
- At least three control plane machines. The control plane machines are not governed by a MachineSet.
- Compute machines. You must create at least two compute, or worker, machines during installation. These machines are not governed by a MachineSet.

You can use the following instance types for the cluster machines:



VALID INSTANCE TYPES FOR MACHINES

If **m4** instance types are not available in your region, such as with **eu-west-3**, use **m5** types instead.

Instance type	Bootstrap	Control plane	Compute
i3.large	x		
m4.large or m5.large			x
m4.xlarge or m5.xlarge		x	x
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.8xlarge		x	x

Instance type	Bootstrap	Control plane	Compute
m4.10xlarge		x	x
m4.16xlarge		x	x
c4.large			x
c4.xlarge			x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
r4.large			x
r4.xlarge		x	x
r4.2xlarge		x	x
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x

2.1.2.2. Other infrastructure components

- A VPC
- DNS entries
- Load balancers and listeners
- A Route53 zone
- Security groups
- IAM roles
- S3 buckets

Required VPC components

You must provide a suitable VPC and subnets that allow communication to your machines.

Component	AWS type	Description	
VPC	<ul style="list-style-type: none">● AWS::EC2::VPC● AWS::EC2::VPCEndpoint	You must provide a public VPC for the cluster to use. The VPC requires an endpoint that references the route tables for each subnet.	
Public subnets	<ul style="list-style-type: none">● AWS::EC2::Subnet● AWS::EC2::SubnetNetworkAclAssociation	Your VPC must have public subnets for between 1 and 3 availability zones and associate them with appropriate Ingress rules.	
Internet gateway	<ul style="list-style-type: none">● AWS::EC2::InternetGateway● AWS::EC2::VPCGatewayAttachment● AWS::EC2::RouteTable● AWS::EC2::Route● PublicSubnetRouteTableAssociation● AWS::EC2::NatGateway● AWS::EC2::EIP	You must have a public internet gateway, with public routes, attached to the VPC. Each public subnet must also be attached to the route and have a NAT gateway and EIP address.	
Network access control	<ul style="list-style-type: none">● AWS::EC2::NetworkAcl● AWS::EC2::NetworkAclEntry	You must allow the VPC to access the following ports:	
		Port	Reason
		80	Inbound HTTP traffic
		443	Inbound HTTPS traffic
		22	Inbound SSH traffic
		1024 - 65535	Inbound ephemeral traffic
		0 - 65535	Outbound ephemeral traffic

Component	AWS type	Description
Private subnets	<ul style="list-style-type: none"> ● AWS::EC2::Subnet ● AWS::EC2::RouteTable ● AWS::EC2::SubnetRouteTableAssociation 	Your VPC can have a private subnets. The provided CloudFormation templates can create private subnets for between 1 and 3 availability zones. If you use private subnets, you must provide appropriate routes and tables for them.

Required DNS and load balancing components

Your DNS and load balancer configuration needs to use a public hosted zone and can use a private hosted zone similar to the one that the installation program uses if it provisions the cluster's infrastructure. You must create a DNS entry that resolves to your load balancer. An entry for **api.<cluster_name>.<domain>** must point to the external load balancer, and an entry for **api-int.<cluster_name>.<domain>** must point to the internal load balancer.

The cluster also requires load balancers and listeners for port 6443, which are required for the Kubernetes API and its extensions, and port 22623, which are required for the Ignition config files for new machines. The targets will be the master nodes. Port 6443 must be accessible to both clients external to the cluster and nodes within the cluster. Port 22623 must be accessible to nodes within the cluster.

Component	AWS type	Description
DNS	AWS::Route53::HostedZone	The hosted zone for your internal DNS.
etcd record sets	AWS::Route53::RecordSet	The registration records for etcd for your control plane machines.
Public load balancer	AWS::ElasticLoadBalancingV2::LoadBalancer	The load balancer for your public subnets.
External API server record	AWS::Route53::RecordSetGroup	Alias records for the external API server.
External listener	AWS::ElasticLoadBalancingV2::Listener	A listener on port 6443 for the external load balancer.

Component	AWS type	Description
External target group	AWS::ElasticLoadBalancingV2::TargetGroup	The target group for the external load balancer.
Private load balancer	AWS::ElasticLoadBalancingV2::LoadBalancer	The load balancer for your private subnets.
Internal API server record	AWS::Route53::RecordSetGroup	Alias records for the internal API server.
Internal listener	AWS::ElasticLoadBalancingV2::Listener	A listener on port 26443 for the internal load balancer.
Internal target group	AWS::ElasticLoadBalancingV2::TargetGroup	The target group for the Internal load balancer.
Internal listener	AWS::ElasticLoadBalancingV2::Listener	A listener on port 26443 for the internal load balancer.
Internal target group	AWS::ElasticLoadBalancingV2::TargetGroup	The target group for the Internal load balancer.

Security groups

The control plane and worker machines require access to the following ports:

Group	Type	IP Protocol	Port range
MasterSecurityGroup	AWS::EC2::SecurityGroup	icmp	0
		tcp	22
		tcp	6443
		tcp	22623

Group	Type	IP Protocol	Port range
WorkerSecurityGroup	AWS::EC2::Security Group	icmp	0
		tcp	22
BootstrapSecurityGroup	AWS::EC2::Security Group	tcp	22
		tcp	19531

Control plane Ingress

The control plane machines require the following Ingress groups. Each Ingress group is a **AWS::EC2::SecurityGroupIngress** resource.

Ingress group	Description	IP protocol	Port range
MasterIngress Etcd	etcd	tcp	2379- 2380
MasterIngress Vxlan	Vxlan packets	udp	4789
MasterIngress WorkerVxlan	Vxlan packets	udp	4789
MasterIngress Internal	Internal cluster communication	tcp	9000 - 9999
MasterIngress WorkerInternal	Internal cluster communication	tcp	9000 - 9999
MasterIngress Kube	Kubernetes kubelet, scheduler and controller manager	tcp	10250 - 10259
MasterIngress WorkerKube	Kubernetes kubelet, scheduler and controller manager	tcp	10250 - 10259
MasterIngress IngressServices	Kubernetes Ingress services	tcp	30000 - 32767
MasterIngress WorkerIngressServices	Kubernetes Ingress services	tcp	30000 - 32767

Worker Ingress

The worker machines require the following Ingress groups. Each Ingress group is a **AWS::EC2::SecurityGroupIngress** resource.

Ingress group	Description	IP protocol	Port range
WorkerIngress Vxlan	Vxlan packets	udp	4789
WorkerIngress WorkerVxlan	Vxlan packets	udp	4789
WorkerIngress Internal	Internal cluster communication	tcp	9000 - 9999
WorkerIngress WorkerInternal	Internal cluster communication	tcp	9000 - 9999
WorkerIngress Kube	Kubernetes kubelet, scheduler and controller manager	tcp	10250
WorkerIngress WorkerKube	Kubernetes kubelet, scheduler and controller manager	tcp	10250
WorkerIngress IngressServices	Kubernetes Ingress services	tcp	30000 - 32767
WorkerIngress WorkerIngressServices	Kubernetes Ingress services	tcp	30000 - 32767

Roles and instance profiles

You must grant the machines permissions in AWS. The provided CloudFormation templates grant the machines permission the following **AWS::IAM::Role** objects and provide a **AWS::IAM::InstanceProfile** for each set of roles. If you do not use the templates, you can grant the machines the following broad permissions or the following individual permissions.

Role	Effect	Action	Resource
Master	Allow	ec2:*	*
	Allow	elasticloadbalancing:*	*
	Allow	iam:PassRole	*
	Allow	s3:GetObject	*

Role	Effect	Action	Resource
Worker	Allow	ec2:Describe*	*
Bootstrap	Allow	ec2:Describe*	*
	Allow	ec2:AttachVolume	*
	Allow	ec2:DetachVolume	*

2.1.2.3. Required AWS permissions

When you attach the **AdministratorAccess** policy to the IAM user that you create, you grant that user all of the required permissions. To deploy an OpenShift Container Platform cluster, the IAM user requires the following permissions:

Required EC2 permissions for installation

- **ec2:AllocateAddress**
- **ec2:AssociateAddress**
- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**
- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2>CreateDhcpOptions**
- **ec2>CreateInternetGateway**
- **ec2>CreateNatGateway**
- **ec2>CreateRoute**
- **ec2>CreateRouteTable**
- **ec2>CreateSecurityGroup**
- **ec2>CreateSubnet**
- **ec2:CreateTags**
- **ec2>CreateVpc**
- **ec2>CreateVpcEndpoint**

- **ec2:CreateVolume**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInternetGateways**
- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**
- **ec2:DescribeNetworkAcls**
- **ec2:DescribePrefixLists**
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**
- **ec2:DescribeVpcAttribute**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcClassicLink**
- **ec2:DescribeVpcClassicLinkDnsSupport**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**
- **ec2:RevokeSecurityGroupEgress**

- **ec2:RunInstances**
- **ec2:TerminateInstances**
- **ec2:RevokeSecurityGroupIngress**
- **ec2:ReplaceRouteTableAssociation**
- **ec2:DescribeNetworkInterfaces**
- **ec2:ModifyNetworkInterfaceAttribute**

Required Elasticloadbalancing permissions for installation

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:CreateListener**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateLoadBalancerListeners**
- **elasticloadbalancing:CreateTargetGroup**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:DescribeInstanceHealth**
- **elasticloadbalancing:DescribeListeners**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeTags**
- **elasticloadbalancing:DescribeTargetGroupAttributes**
- **elasticloadbalancing:DescribeTargetHealth**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterTargets**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**

- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**

Required IAM permissions for installation

- **iam:AddRoleToInstanceProfile**
- **iam:CreateInstanceProfile**
- **iam:CreateRole**
- **iam:DeleteInstanceProfile**
- **iam>DeleteRole**
- **iam>DeleteRolePolicy**
- **iam:GetInstanceProfile**
- **iam:GetRole**
- **iam:GetRolePolicy**
- **iam:GetUser**
- **iam:ListInstanceProfilesForRole**
- **iam:ListRoles**
- **iam:ListUsers**
- **iam:PassRole**
- **iam:PutRolePolicy**
- **iam:RemoveRoleFromInstanceProfile**
- **iam:SimulatePrincipalPolicy**
- **iam:TagRole**

Required Route53 permissions for installation

- **route53:ChangeResourceRecordSets**
- **route53:ChangeTagsForResource**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:CreateHostedZone**
- **route53:ListHostedZones**
- **route53:ListHostedZonesByName**
- **route53:ListResourceRecordSets**

- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**

Required S3 permissions for installation

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketCors**
- **s3:GetBucketLocation**
- **s3:GetBucketLogging**
- **s3:GetBucketObjectLockConfiguration**
- **s3:GetBucketReplication**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketTagging**
- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **s3:GetEncryptionConfiguration**
- **s3:GetLifecycleConfiguration**
- **s3:GetReplicationConfiguration**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

S3 permissions that cluster Operators require

- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**

- **s3:GetObjectVersion**
- **s3:DeleteObject**

All additional permissions that are required to uninstall a cluster

- **autoscaling:DescribeAutoScalingGroups**
- **ec2:DeleteDhcpOptions**
- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteNetworkInterface**
- **ec2:DeleteRoute**
- **ec2:DeleteRouteTable**
- **ec2:DeleteSnapshot**
- **ec2:DeleteSecurityGroup**
- **ec2:DeleteSubnet**
- **ec2:DeleteVolume**
- **ec2:DeleteVpc**
- **ec2:DeleteVpcEndpoints**
- **ec2:DeregisterImage**
- **ec2:DetachInternetGateway**
- **ec2:DisassociateRouteTable**
- **ec2:ReleaseAddress**
- **elasticloadbalancing:DescribeTargetGroups**
- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DeleteLoadBalancer**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**
- **route53:DeleteHostedZone**
- **tag:GetResources**

2.1.3. Obtaining the installation program

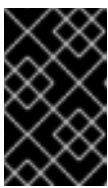
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 300 MB of local disk space to download the installation program.

Procedure

1. Access the [OpenShift Infrastructure Providers](#) page. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Download the installation program for your operating system and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [OpenShift Infrastructure Providers](#) page, download your installation pull secret. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

2.1.4. Generating an SSH private key and adding it to the agent

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, you must provide an SSH key that your **ssh-agent** process uses to the installer.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t rsa -b 4096 -N "" \
-f <path>/<file_name> 1
```

■

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the SSH key.

Running this command generates an SSH key that does not require a password in the location that you specified.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

```
Agent pid 31874
```

3. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

When you install OpenShift Container Platform, provide the SSH public key to the installer. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

2.1.5. Creating the installation files for AWS

To install OpenShift Container Platform on Amazon Web Services (AWS) using user-provisioned infrastructure, you must generate the files that the installation program needs to deploy your cluster and modify them so that the cluster creates only the machines that it will use. You generate and customize the **install_config.yaml** file, Kubernetes manifests, and Ignition config files.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours. You must complete your cluster installation and keep the cluster running for 24 hours in a non-degraded state to ensure that the first certificate rotation has finished.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Obtain the **install-config.yaml** file.
 - a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.



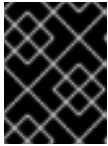
NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, you must provide an SSH key that your **ssh-agent** process uses to the installation program.

- ii. Select AWS as the platform to target.
 - iii. If you do not have an AWS profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.
 - iv. Select the AWS region to deploy the cluster to.
 - v. Select the base domain for the Route53 service that you configured for your cluster.
 - vi. Enter a descriptive name for your cluster.
 - vii. Paste the pull secret that you obtained from the [OpenShift Infrastructure Providers](#) page.
2. Edit the **install-config.yaml** file to set the number of compute, or worker, replicas to **0**, as shown in the following **compute** stanza:

```
compute:
- hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 0
```

3. Optional: Back up the **install-config.yaml** file.



IMPORTANT

The **install-config.yaml** file is consumed during the next step. If you want to reuse the file, back it up now.

4. Remove the Kubernetes manifest files for the control plane machines. By removing these files, you prevent the cluster from automatically generating control plane machines.

- a. Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

WARNING There are no compute nodes specified. The cluster will not fully initialize without compute nodes.

INFO Consuming "Install Config" from target directory

- 1** For **<installation_directory>**, specify the same installation directory.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

- b. Remove the files that define the control plane machines:

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml
```

5. Remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f openshift/99_openshift-cluster-api_worker-machineset-*
```

Because you create and manage the worker machines yourself, you do not need to initialize these machines.

6. Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the same installation directory.

The following files are generated in the directory:

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

2.1.6. Extracting the infrastructure name

The Ignition configs contain a unique cluster identifier that you can use to uniquely identify your cluster in Amazon Web Services (AWS) tags. The provided CloudFormation templates contain references to this tag, so you must extract it.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.
- Generate the Ignition config files for your cluster.
- Install the **jq** package.

Procedure

- To extract the infrastructure name from the Ignition config file metadata, run the following command:

```
$ jq -r .infraID /<installation_directory>/metadata.json 1
openshift-vw9j6 2
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2 The output of this command is your cluster name and a random string.

You need the output of this command to configure the provided CloudFormation templates and can use it in other AWS tags.

2.1.7. Creating a VPC in AWS

You must create a VPC in Amazon Web Services (AWS) for your OpenShift Container Platform cluster to use. You can customize the VPC to meet your requirements, including VPN and route tables. The easiest way to create the VPC is to modify the provided CloudFormation template.



NOTE

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.

Procedure

1. Create a JSON file that contains the parameter values that the template requires:

```
[
{
```

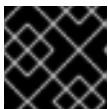
```

    "ParameterKey": "VpcCidr", ❶
    "ParameterValue": "10.0.0.0/16" ❷
  },
  {
    "ParameterKey": "AvailabilityZoneCount", ❸
    "ParameterValue": "1" ❹
  },
  {
    "ParameterKey": "SubnetBits", ❺
    "ParameterValue": "12" ❻
  }
]

```

- ❶ The CIDR block for the VPC.
- ❷ Specify a CIDR block in the format **x.x.x.x/16-24**.
- ❸ The number of availability zones to deploy the VPC in.
- ❹ Specify an integer between **1** and **3**.
- ❺ The size of each subnet in each availability zone.
- ❻ Specify an integer between **5** and **13**, where **5** is /27 and **13** is /19.

2. Copy the template from the **CloudFormation template for the VPC** section of this topic and save it as a YAML file on your computer. This template describes the VPC that your cluster requires.
3. Launch the template:



IMPORTANT

You must enter the command on a single line.

```

$ aws cloudformation create-stack --stack-name <name> ❶
  --template-body file://<template>.yaml ❷
  --parameters file://<parameters>.json ❸

```

- ❶ **<name>** is the name for the CloudFormation stack, such as **cluster-vpc**. You need the name of this stack if you remove the cluster.
- ❷ **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- ❸ **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

4. Confirm that the template components exist:

```

$ aws cloudformation describe-stacks --stack-name <name>

```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

VpcId	The ID of your VPC.
PublicSubnetIds	The IDs of the new public subnets.
PrivateSubnetIds	The IDs of the new private subnets.

2.1.7.1. CloudFormation template for the VPC

You can use the following CloudFormation template to deploy the VPC that you need for your OpenShift Container Platform cluster.

AWSTemplateFormatVersion: 2010-09-09

Description: Template for Best Practice VPC with 1-3 AZs

Parameters:

VpcCidr:

AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\V(1[6-9]|2[0-4]))\$

ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.

Default: 10.0.0.0/16

Description: CIDR block for VPC.

Type: String

AvailabilityZoneCount:

ConstraintDescription: "The number of availability zones. (Min: 1, Max: 3)"

MinValue: 1

MaxValue: 3

Default: 1

Description: "How many AZs to create VPC subnets for. (Min: 1, Max: 3)"

Type: Number

SubnetBits:

ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/19-27.

MinValue: 5

MaxValue: 13

Default: 12

Description: "Size of each subnet to create within the availability zones. (Min: 5 = /27, Max: 13 = /19)"

Type: Number

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:

default: "Network Configuration"

Parameters:

- VpcCidr

- SubnetBits

- Label:

default: "Availability Zones"

Parameters:

- AvailabilityZoneCount

ParameterLabels:

AvailabilityZoneCount:

default: "Availability Zone Count"

VpcCidr:

default: "VPC CIDR"

SubnetBits:

default: "Bits Per Subnet"

Conditions:

DoAz3: !Equals [3, !Ref AvailabilityZoneCount]

DoAz2: !Or [!Equals [2, !Ref AvailabilityZoneCount], Condition: DoAz3]

Resources:

VPC:

Type: "AWS::EC2::VPC"

Properties:

EnableDnsSupport: "true"

EnableDnsHostnames: "true"

CidrBlock: !Ref VpcCidr

PublicSubnet:

Type: "AWS::EC2::Subnet"

Properties:

VpcId: !Ref VPC

CidrBlock: !Select [0, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]

AvailabilityZone: !Select

- 0

- Fn::GetAZs: !Ref "AWS::Region"

PublicSubnet2:

Type: "AWS::EC2::Subnet"

Condition: DoAz2

Properties:

VpcId: !Ref VPC

CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]

AvailabilityZone: !Select

- 1

- Fn::GetAZs: !Ref "AWS::Region"

PublicSubnet3:

Type: "AWS::EC2::Subnet"

Condition: DoAz3

Properties:

VpcId: !Ref VPC

CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]

AvailabilityZone: !Select

- 2

- Fn::GetAZs: !Ref "AWS::Region"

InternetGateway:

Type: "AWS::EC2::InternetGateway"

GatewayToInternet:

Type: "AWS::EC2::VPCEGatewayAttachment"

Properties:

VpcId: !Ref VPC

InternetGatewayId: !Ref InternetGateway

PublicRouteTable:

```
Type: "AWS::EC2::RouteTable"
Properties:
  VpcId: !Ref VPC
PublicRoute:
  Type: "AWS::EC2::Route"
  DependsOn: GatewayToInternet
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway
PublicSubnetRouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PublicSubnet
    RouteTableId: !Ref PublicRouteTable
PublicSubnetRouteTableAssociation2:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz2
  Properties:
    SubnetId: !Ref PublicSubnet2
    RouteTableId: !Ref PublicRouteTable
PublicSubnetRouteTableAssociation3:
  Condition: DoAz3
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PublicSubnet3
    RouteTableId: !Ref PublicRouteTable
PublicNetworkAcl:
  Type: "AWS::EC2::NetworkAcl"
  Properties:
    VpcId: !Ref VPC
InboundHTTPPublicNetworkAclEntry:
  Type: "AWS::EC2::NetworkAclEntry"
  Properties:
    NetworkAclId: !Ref PublicNetworkAcl
    RuleNumber: "100"
    Protocol: "6"
    RuleAction: allow
    Egress: "false"
    CidrBlock: 0.0.0.0/0
    PortRange:
      From: "80"
      To: "80"
InboundHTTPSPublicNetworkAclEntry:
  Type: "AWS::EC2::NetworkAclEntry"
  Properties:
    NetworkAclId: !Ref PublicNetworkAcl
    RuleNumber: "101"
    Protocol: "6"
    RuleAction: allow
    Egress: "false"
    CidrBlock: 0.0.0.0/0
    PortRange:
      From: "443"
      To: "443"
InboundSSHPublicNetworkAclEntry:
```

Type: "AWS::EC2::NetworkAclEntry"

Properties:

NetworkAclId: !Ref PublicNetworkAcl

RuleNumber: "102"

Protocol: "6"

RuleAction: allow

Egress: "false"

CidrBlock: 0.0.0.0/0

PortRange:

From: "22"

To: "22"

InboundEphemeralPublicNetworkAclEntry:

Type: "AWS::EC2::NetworkAclEntry"

Properties:

NetworkAclId: !Ref PublicNetworkAcl

RuleNumber: "103"

Protocol: "6"

RuleAction: allow

Egress: "false"

CidrBlock: 0.0.0.0/0

PortRange:

From: "1024"

To: "65535"

OutboundPublicNetworkAclEntry:

Type: "AWS::EC2::NetworkAclEntry"

Properties:

NetworkAclId: !Ref PublicNetworkAcl

RuleNumber: "100"

Protocol: "6"

RuleAction: allow

Egress: "true"

CidrBlock: 0.0.0.0/0

PortRange:

From: "0"

To: "65535"

PublicSubnetNetworkAclAssociation:

Type: "AWS::EC2::SubnetNetworkAclAssociation"

Properties:

SubnetId: !Ref PublicSubnet

NetworkAclId: !Ref PublicNetworkAcl

PublicSubnetNetworkAclAssociation2:

Type: "AWS::EC2::SubnetNetworkAclAssociation"

Condition: DoAz2

Properties:

SubnetId: !Ref PublicSubnet2

NetworkAclId: !Ref PublicNetworkAcl

PublicSubnetNetworkAclAssociation3:

Type: "AWS::EC2::SubnetNetworkAclAssociation"

Condition: DoAz3

Properties:

SubnetId: !Ref PublicSubnet3

NetworkAclId: !Ref PublicNetworkAcl

PrivateSubnet:

Type: "AWS::EC2::Subnet"

Properties:

VpcId: !Ref VPC

```

    CidrBlock: !Select [3, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable:
  Type: "AWS::EC2::RouteTable"
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PrivateSubnet
    RouteTableId: !Ref PrivateRouteTable
NAT:
  DependsOn:
    - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Properties:
    AllocationId:
      "Fn::GetAtt":
        - EIP
        - AllocationId
    SubnetId: !Ref PublicSubnet
EIP:
  Type: "AWS::EC2::EIP"
  Properties:
    Domain: vpc
Route:
  Type: "AWS::EC2::Route"
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT
PrivateSubnet2:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [4, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable2:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation2:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz2
  Properties:
    SubnetId: !Ref PrivateSubnet2
    RouteTableId: !Ref PrivateRouteTable2
NAT2:

```

```

DependsOn:
- GatewayToInternet
Type: "AWS::EC2::NatGateway"
Condition: DoAz2
Properties:
  AllocationId:
    "Fn::GetAtt":
      - EIP2
      - AllocationId
  SubnetId: !Ref PublicSubnet2
EIP2:
  Type: "AWS::EC2::EIP"
  Condition: DoAz2
  Properties:
    Domain: vpc
Route2:
  Type: "AWS::EC2::Route"
  Condition: DoAz2
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT2
PrivateSubnet3:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [5, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable3:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation3:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz3
  Properties:
    SubnetId: !Ref PrivateSubnet3
    RouteTableId: !Ref PrivateRouteTable3
NAT3:
  DependsOn:
    - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz3
  Properties:
    AllocationId:
      "Fn::GetAtt":
        - EIP3
        - AllocationId
    SubnetId: !Ref PublicSubnet3
EIP3:

```

```

Type: "AWS::EC2::EIP"
Condition: DoAz3
Properties:
  Domain: vpc
Route3:
Type: "AWS::EC2::Route"
Condition: DoAz3
Properties:
  RouteTableId:
    Ref: PrivateRouteTable3
  DestinationCidrBlock: 0.0.0.0/0
  NatGatewayId:
    Ref: NAT3
S3Endpoint:
Type: AWS::EC2::VPCEndpoint
Properties:
  PolicyDocument:
    Version: 2012-10-17
    Statement:
      - Effect: Allow
        Principal: '*'
        Action:
          - '*'
        Resource:
          - '*'
  RouteTableIds:
    - !Ref PublicRouteTable
    - !Ref PrivateRouteTable
    - !If [DoAz2, !Ref PrivateRouteTable2, !Ref "AWS::NoValue"]
    - !If [DoAz3, !Ref PrivateRouteTable3, !Ref "AWS::NoValue"]
  ServiceName: !Join
    - "
    - - com.amazonaws.
      - !Ref 'AWS::Region'
      - .s3
  VpcId: !Ref VPC

Outputs:
VpcId:
  Description: ID of the new VPC.
  Value: !Ref VPC
PublicSubnetIds:
  Description: Subnet IDs of the public subnets.
  Value:
    !Join [
      " ",
      [!Ref PublicSubnet, !If [DoAz2, !Ref PublicSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PublicSubnet3, !Ref "AWS::NoValue"]]
    ]
PrivateSubnetIds:
  Description: Subnet IDs of the private subnets.
  Value:
    !Join [
      " ",

```

```
[!Ref PrivateSubnet, !If [DoAz2, !Ref PrivateSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PrivateSubnet3, !Ref "AWS::NoValue"]]]
]
```

2.1.8. Creating networking and load balancing components in AWS

You must configure networking and load balancing in Amazon Web Services (AWS) for your OpenShift Container Platform cluster to use. The easiest way to create these components is to modify the provided CloudFormation template, which also creates a hosted zone and subnet tags.

You can run the template multiple times within a single VPC.



NOTE

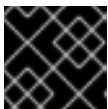
If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.

Procedure

1. Obtain the Hosted Zone ID for the the Route53 zone that you specified in the **install-config.yaml** file for your cluster. You can obtain this ID from the AWS console or by running the following command:



IMPORTANT

You must enter the command on a single line.

```
$ aws route53 list-hosted-zones-by-name |
jq --arg name "<route53_domain>." \ 1
-r '.HostedZones | .[] | select(.Name=="\($name)") | .Id'
```

- 1** For the **<route53_domain>**, specify the Route53 base domain that you used when you generated the **install-config.yaml** file for the cluster.

2. Create a JSON file that contains the parameter values that the template requires:

```
[
{
  "ParameterKey": "ClusterName", 1
  "ParameterValue": "mycluster" 2
},
{
```

```

    "ParameterKey": "InfrastructureName", 3
    "ParameterValue": "mycluster-<random_string>" 4
  },
  {
    "ParameterKey": "HostedZoneId", 5
    "ParameterValue": "<random_string>" 6
  },
  {
    "ParameterKey": "HostedZoneName", 7
    "ParameterValue": "example.com" 8
  },
  {
    "ParameterKey": "PublicSubnets", 9
    "ParameterValue": "subnet-<random_string>" 10
  },
  {
    "ParameterKey": "PrivateSubnets", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
    "ParameterKey": "VpcId", 13
    "ParameterValue": "vpc-<random_string>" 14
  }
]

```

- 1 A short, representative cluster name to use for host names, etc.
- 2 Specify the cluster name that you used when you generated the **install-config.yaml** file for the cluster.
- 3 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- 4 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
- 5 The Route53 public zone ID to register the targets with.
- 6 Specify the Route53 public zone ID, which as a format similar to **Z21IXYZABCZ2A4**. You can obtain this value from the AWS console.
- 7 The Route53 zone to register the targets with.
- 8 Specify the Route53 base domain that you used when you generated the **install-config.yaml** file for the cluster. Do not include the trailing period (.) that is displayed in the AWS console.
- 9 The public subnets that you created for your VPC.
- 10 Specify the **PublicSubnetIds** value from the output of the CloudFormation template for the VPC.
- 11 The private subnets that you created for your VPC.
- 12 Specify the **PrivateSubnetIds** value from the output of the CloudFormation template for the VPC.

- the VPC.
- 13 The VPC that you created for the cluster.
 - 14 Specify the **VpcId** value from the output of the CloudFormation template for the VPC.
3. Copy the template from the **CloudFormation template for the network and load balancers** section of this topic and save it as a YAML file on your computer. This template describes the networking and load balancing objects that your cluster requires.
 4. Launch the template:



IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> 1
  --template-body file://<template>.yaml 2
  --parameters file://<parameters>.json 3
  --capabilities CAPABILITY_NAMED_IAM
```

- 1 **<name>** is the name for the CloudFormation stack, such as **cluster-dns**. You need the name of this stack if you remove the cluster.
- 2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- 3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

5. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

PrivateHostedZoneId	Hosted zone ID for the private DNS.
ExternalApiLoadBalancerName	Full name of the external API load balancer.
InternalApiLoadBalancerName	Full name of the internal API load balancer.

ApiServer DnsName	Full host name of the API server.
RegisterN lbIpTarget sLambda	Lambda ARN useful to help register/deregister IP targets for these load balancers.
ExternalA piTargetG roupArn	ARN of external API target group.
InternalAp iTargetGr oupArn	ARN of internal API target group.
InternalSe rviceTarg etGroupA rn	ARN of internal service target group.

2.1.8.1. CloudFormation template for the network and load balancers

You can use the following CloudFormation template to deploy the networking objects and load balancers that you need for your OpenShift Container Platform cluster.

AWSTemplateFormatVersion: [2010-09-09](#)

Description: Template for OpenShift Cluster Network Elements (Route53 & LBs)

Parameters:

ClusterName:

AllowedPattern: `^[a-zA-Z][a-zA-Z0-9-]{0,26}$`

MaxLength: [27](#)

MinLength: [1](#)

ConstraintDescription: Cluster name must be alphanumeric, start with a letter, and have a maximum of [27](#) characters.

Description: A short, representative cluster name to use for host names and other identifying names.

Type: String

InfrastructureName:

AllowedPattern: `^[a-zA-Z][a-zA-Z0-9-]{0,26}$`

MaxLength: [27](#)

MinLength: [1](#)

ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of [27](#) characters.

Description: A short, unique cluster ID used to tag cloud resources and identify items owned or used by the cluster.

Type: String

HostedZoneId:

Description: The Route53 public zone ID to register the targets with, such as Z21IXYZABCZ2A4.

Type: String

HostedZoneName:

Description: The Route53 zone to register the targets with, such as example.com. Omit the trailing period.

```

Type: String
Default: "example.com"
PublicSubnets:
  Description: The internet-facing subnets.
  Type: List<AWS::EC2::Subnet::Id>
PrivateSubnets:
  Description: The internal subnets.
  Type: List<AWS::EC2::Subnet::Id>
VpcId:
  Description: The VPC-scoped resources will belong to this VPC.
  Type: AWS::EC2::VPC::Id

Metadata:
AWS::CloudFormation::Interface:
  ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
        - ClusterName
        - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
        - VpcId
        - PublicSubnets
        - PrivateSubnets
    - Label:
        default: "DNS"
      Parameters:
        - HostedZoneName
        - HostedZoneId
  ParameterLabels:
    ClusterName:
      default: "Cluster Name"
    InfrastructureName:
      default: "Infrastructure Name"
    VpcId:
      default: "VPC ID"
    PublicSubnets:
      default: "Public Subnets"
    PrivateSubnets:
      default: "Private Subnets"
    HostedZoneName:
      default: "Public Hosted Zone Name"
    HostedZoneId:
      default: "Public Hosted Zone ID"

Resources:
ExtApiElb:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Name: !Join ["-", [!Ref InfrastructureName, "ext"]]
    IpAddressType: ipv4
    Subnets: !Ref PublicSubnets
    Type: network

```

```

IntApiElb:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Name: !Join ["-", [!Ref InfrastructureName, "int"]]
    Scheme: internal
    IpAddressType: ipv4
    Subnets: !Ref PrivateSubnets
    Type: network

IntDns:
  Type: "AWS::Route53::HostedZone"
  Properties:
    HostedZoneConfig:
      Comment: "Managed by CloudFormation"
    Name: !Join [".", [!Ref ClusterName, !Ref HostedZoneName]]
    HostedZoneTags:
      - Key: Name
        Value: !Join ["-", [!Ref InfrastructureName, "int"]]
      - Key: !Join [""], ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "owned"
    VPCs:
      - VPCId: !Ref VpcId
        VPCRegion: !Ref "AWS::Region"

ExternalApiServerRecord:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    Comment: Alias record for the API server
    HostedZoneId: !Ref HostedZoneId
    RecordSets:
      - Name:
          !Join [
            ".",
            ["api", !Ref ClusterName, !Join [""], [!Ref HostedZoneName, "."]]],
          Type: A
          AliasTarget:
            HostedZoneId: !GetAtt ExtApiElb.CanonicalHostedZoneID
            DNSName: !GetAtt ExtApiElb.DNSName

InternalApiServerRecord:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    Comment: Alias record for the API server
    HostedZoneId: !Ref IntDns
    RecordSets:
      - Name:
          !Join [
            ".",
            ["api", !Ref ClusterName, !Join [""], [!Ref HostedZoneName, "."]]],
          Type: A
          AliasTarget:
            HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
            DNSName: !GetAtt IntApiElb.DNSName
      - Name:

```

```

!Join [
  ".",
  ["api-int", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
]
Type: A
AliasTarget:
  HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneId
  DNSName: !GetAtt IntApiElb.DNSName

```

ExternalApiListener:

```

Type: AWS::ElasticLoadBalancingV2::Listener
Properties:
  DefaultActions:
    - Type: forward
  TargetGroupArn:
    Ref: ExternalApiTargetGroup
  LoadBalancerArn:
    Ref: ExtApiElb
  Port: 6443
  Protocol: TCP

```

ExternalApiTargetGroup:

```

Type: AWS::ElasticLoadBalancingV2::TargetGroup
Properties:
  Port: 6443
  Protocol: TCP
  TargetType: ip
  VpcId:
    Ref: VpcId
  TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

```

InternalApiListener:

```

Type: AWS::ElasticLoadBalancingV2::Listener
Properties:
  DefaultActions:
    - Type: forward
  TargetGroupArn:
    Ref: InternalApiTargetGroup
  LoadBalancerArn:
    Ref: IntApiElb
  Port: 6443
  Protocol: TCP

```

InternalApiTargetGroup:

```

Type: AWS::ElasticLoadBalancingV2::TargetGroup
Properties:
  Port: 6443
  Protocol: TCP
  TargetType: ip
  VpcId:
    Ref: VpcId
  TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

```

InternalServiceInternalListener:

Type: AWS::ElasticLoadBalancingV2::Listener

Properties:

DefaultActions:

- Type: forward

TargetGroupArn:

Ref: InternalServiceTargetGroup

LoadBalancerArn:

Ref: IntApiElb

Port: 22623

Protocol: TCP

InternalServiceTargetGroup:

Type: AWS::ElasticLoadBalancingV2::TargetGroup

Properties:

Port: 22623

Protocol: TCP

TargetType: ip

VpcId:

Ref: VpcId

TargetGroupAttributes:

- Key: deregistration_delay.timeout_seconds

Value: 60

RegisterTargetLambdalamRole:

Type: AWS::IAM::Role

Properties:

RoleName: !Join ["-", [!Ref InfrastructureName, "nlb", "lambda", "role"]]

AssumeRolePolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Principal:

Service:

- "lambda.amazonaws.com"

Action:

- "sts:AssumeRole"

Path: "/"

Policies:

- PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]

PolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Action:

```
[
  "elasticloadbalancing:RegisterTargets",
  "elasticloadbalancing:DeregisterTargets",
]
```

Resource: !Ref InternalApiTargetGroup

- Effect: "Allow"

Action:

```
[
  "elasticloadbalancing:RegisterTargets",
  "elasticloadbalancing:DeregisterTargets",
]
```

```

    ]
    Resource: !Ref InternalServiceTargetGroup
  - Effect: "Allow"
    Action:
      [
        "elasticloadbalancing:RegisterTargets",
        "elasticloadbalancing:DeregisterTargets",
      ]
    Resource: !Ref ExternalApiTargetGroup

```

RegisterNlbIpTargets:

```

Type: "AWS::Lambda::Function"
Properties:
  Handler: "index.handler"
  Role:
    Fn::GetAtt:
      - "RegisterTargetLambdalaRole"
      - "Arn"
  Code:
    ZipFile: |
      import json
      import boto3
      import cfnresponse
      def handler(event, context):
        elb = boto3.client('elbv2')
        if event['RequestType'] == 'Delete':
          elb.deregister_targets(TargetGroupArn=event['ResourceProperties']['TargetArn'],Targets=
[{'Id': event['ResourceProperties']['TargetIp']})
          elif event['RequestType'] == 'Create':
            elb.register_targets(TargetGroupArn=event['ResourceProperties']['TargetArn'],Targets=[{'Id':
event['ResourceProperties']['TargetIp']})
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['TargetArn']+event['ResourceProperties']['TargetIp'])
  Runtime: "python3.7"
  Timeout: 120

```

RegisterSubnetTagsLambdalaRole:

```

Type: AWS::IAM::Role
Properties:
  RoleName: !Join ["-", [!Ref InfrastructureName, "subnet-tags-lambda-role"]]
  AssumeRolePolicyDocument:
    Version: "2012-10-17"
    Statement:
      - Effect: "Allow"
        Principal:
          Service:
            - "lambda.amazonaws.com"
        Action:
          - "sts:AssumeRole"
  Path: "/"
  Policies:
    - PolicyName: !Join ["-", [!Ref InfrastructureName, "subnet-tagging-policy"]]
      PolicyDocument:
        Version: "2012-10-17"
        Statement:

```

```

- Effect: "Allow"
  Action:
    [
      "ec2:DeleteTags",
      "ec2:CreateTags"
    ]
  Resource: "arn:aws:ec2:*:*:subnet/*"
- Effect: "Allow"
  Action:
    [
      "ec2:DescribeSubnets",
      "ec2:DescribeTags"
    ]
  Resource: "*"

```

RegisterSubnetTags:

Type: "AWS::Lambda::Function"

Properties:

Handler: "index.handler"

Role:

Fn::GetAtt:

- "RegisterSubnetTagsLambdalamRole"

- "Arn"

Code:

ZipFile: |

```

import json
import boto3
import cfnresponse
def handler(event, context):
    ec2_client = boto3.client('ec2')
    if event['RequestType'] == 'Delete':
        for subnet_id in event['ResourceProperties']['Subnets']:
            ec2_client.delete_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName']}]);
    elif event['RequestType'] == 'Create':
        for subnet_id in event['ResourceProperties']['Subnets']:
            ec2_client.create_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName'], 'Value': 'shared'}]);
    responseData = {}
    cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['InfrastructureName']+event['ResourceProperties']['Subnets'][0])
Runtime: "python3.7"
Timeout: 120

```

RegisterPublicSubnetTags:

Type: Custom::SubnetRegister

Properties:

ServiceToken: !GetAtt RegisterSubnetTags.Arn

InfrastructureName: !Ref InfrastructureName

Subnets: !Ref PublicSubnets

RegisterPrivateSubnetTags:

Type: Custom::SubnetRegister

Properties:

ServiceToken: !GetAtt RegisterSubnetTags.Arn

InfrastructureName: !Ref InfrastructureName

Subnets: !Ref PrivateSubnets

Outputs:

PrivateHostedZoneId:

Description: Hosted zone ID for the private DNS, which is required for private records.

Value: !Ref IntDns

ExternalApiLoadBalancerName:

Description: Full name of the External API load balancer created.

Value: !GetAtt ExtApiElb.LoadBalancerFullName

InternalApiLoadBalancerName:

Description: Full name of the Internal API load balancer created.

Value: !GetAtt IntApiElb.LoadBalancerFullName

ApiServerDnsName:

Description: Full hostname of the API server, which is required for the Ignition config files.

Value: !Join [".", ["api-int", !Ref ClusterName, !Ref HostedZoneName]]

RegisterNlbTargetsLambda:

Description: Lambda ARN useful to help register or deregister IP targets for these load balancers.

Value: !GetAtt RegisterNlbTargets.Arn

ExternalApiTargetGroupArn:

Description: ARN of External API target group.

Value: !Ref ExternalApiTargetGroup

InternalApiTargetGroupArn:

Description: ARN of Internal API target group.

Value: !Ref InternalApiTargetGroup

InternalServiceTargetGroupArn:

Description: ARN of internal service target group.

Value: !Ref InternalServiceTargetGroup

2.1.9. Creating security group and roles in AWS

You must create security groups and roles in Amazon Web Services (AWS) for your OpenShift Container Platform cluster to use. The easiest way to create these components is to modify the provided CloudFormation template.



NOTE

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.

Procedure

1. Create a JSON file that contains the parameter values that the template requires:

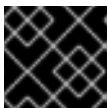
```
[
{
```

```

    "ParameterKey": "InfrastructureName", ❶
    "ParameterValue": "mycluster-<random_string>" ❷
  },
  {
    "ParameterKey": "VpcCidr", ❸
    "ParameterValue": "10.0.0.0/16" ❹
  },
  {
    "ParameterKey": "PrivateSubnets", ❺
    "ParameterValue": "subnet-<random_string>" ❻
  },
  {
    "ParameterKey": "VpcId", ❼
    "ParameterValue": "vpc-<random_string>" ❽
  }
]

```

- ❶ The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
 - ❷ Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
 - ❸ The CIDR block for the VPC.
 - ❹ Specify the CIDR block parameter that you used for the VPC that you defined in the form **x.x.x.x/16-24**.
 - ❺ The private subnets that you created for your VPC.
 - ❻ Specify the **PrivateSubnetIds** value from the output of the CloudFormation template for the VPC.
 - ❼ The VPC that you created for the cluster.
 - ❽ Specify the **VpcId** value from the output of the CloudFormation template for the VPC.
2. Copy the template from the **CloudFormation template for security objects** section of this topic and save it as a YAML file on your computer. This template describes the security groups and roles that your cluster requires.
 3. Launch the template:



IMPORTANT

You must enter the command on a single line.

```

$ aws cloudformation create-stack --stack-name <name> ❶
  --template-body file://<template>.yaml ❷
  --parameters file://<parameters>.json ❸
  --capabilities CAPABILITY_NAMED_IAM

```

- 1 **<name>** is the name for the CloudFormation stack, such as **cluster-sec**. You need the name of this stack if you remove the cluster.
- 2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- 3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

4. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

MasterSecurityGroupID	Master Security Group ID
WorkerSecurityGroupID	Worker Security Group ID
MasterInstanceProfile	Master IAM Instance Profile
WorkerInstanceProfile	Worker IAM Instance Profile

2.1.9.1. CloudFormation template for security objects

You can use the following CloudFormation template to deploy the security objects that you need for your OpenShift Container Platform cluster.

AWSTemplateFormatVersion: 2010-09-09

Description: Template for OpenShift Cluster Security Elements (Security Groups & IAM)

Parameters:

InfrastructureName:

AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-\]{0,26})\$

MaxLength: 27

MinLength: 1

ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of 27 characters.

Description: A short, unique cluster ID used to tag cloud resources and identify items owned or used by the cluster.

Type: String

VpcCidr:

```

AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4]
[0-9]|25[0-5])(\.(1[6-9]|2[0-4]))$
ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
Default: 10.0.0.0/16
Description: CIDR block for VPC.
Type: String
VpcId:
Description: The VPC-scoped resources will belong to this VPC.
Type: AWS::EC2::VPC::Id
PrivateSubnets:
Description: The internal subnets.
Type: List<AWS::EC2::Subnet::Id>

Metadata:
AWS::CloudFormation::Interface:
ParameterGroups:
- Label:
  default: "Cluster Information"
Parameters:
- InfrastructureName
- Label:
  default: "Network Configuration"
Parameters:
- VpcId
- VpcCidr
- PrivateSubnets
ParameterLabels:
InfrastructureName:
  default: "Infrastructure Name"
VpcId:
  default: "VPC ID"
VpcCidr:
  default: "VPC CIDR"
PrivateSubnets:
  default: "Private Subnets"

Resources:
MasterSecurityGroup:
Type: AWS::EC2::SecurityGroup
Properties:
GroupDescription: Cluster Master Security Group
SecurityGroupIngress:
- IpProtocol: icmp
  FromPort: 0
  ToPort: 0
  CidrIp: !Ref VpcCidr
- IpProtocol: tcp
  FromPort: 22
  ToPort: 22
  CidrIp: !Ref VpcCidr
- IpProtocol: tcp
  ToPort: 6443
  FromPort: 6443
  CidrIp: !Ref VpcCidr
- IpProtocol: tcp
  FromPort: 22623

```

ToPort: 22623
 CidrIp: !Ref VpcCidr
 VpcId: !Ref VpcId

WorkerSecurityGroup:

Type: AWS::EC2::SecurityGroup
 Properties:
 GroupDescription: Cluster Worker Security Group
 SecurityGroupIngress:
 - IpProtocol: icmp
 FromPort: 0
 ToPort: 0
 CidrIp: !Ref VpcCidr
 - IpProtocol: tcp
 FromPort: 22
 ToPort: 22
 CidrIp: !Ref VpcCidr
 VpcId: !Ref VpcId

MasterIngressEtcd:

Type: AWS::EC2::SecurityGroupIngress
 Properties:
 GroupId: !GetAtt MasterSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
 Description: etcd
 FromPort: 2379
 ToPort: 2380
 IpProtocol: tcp

MasterIngressVxlan:

Type: AWS::EC2::SecurityGroupIngress
 Properties:
 GroupId: !GetAtt MasterSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
 Description: Vxlan packets
 FromPort: 4789
 ToPort: 4789
 IpProtocol: udp

MasterIngressWorkerVxlan:

Type: AWS::EC2::SecurityGroupIngress
 Properties:
 GroupId: !GetAtt MasterSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
 Description: Vxlan packets
 FromPort: 4789
 ToPort: 4789
 IpProtocol: udp

MasterIngressInternal:

Type: AWS::EC2::SecurityGroupIngress
 Properties:
 GroupId: !GetAtt MasterSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
 Description: Internal cluster communication
 FromPort: 9000

ToPort: 9999

IpProtocol: tcp

MasterIngressWorkerInternal:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Internal cluster communication

FromPort: 9000

ToPort: 9999

IpProtocol: tcp

MasterIngressKube:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Kubernetes kubelet, scheduler and controller manager

FromPort: 10250

ToPort: 10259

IpProtocol: tcp

MasterIngressWorkerKube:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Kubernetes kubelet, scheduler and controller manager

FromPort: 10250

ToPort: 10259

IpProtocol: tcp

MasterIngressIngressServices:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Kubernetes ingress services

FromPort: 30000

ToPort: 32767

IpProtocol: tcp

MasterIngressWorkerIngressServices:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Kubernetes ingress services

FromPort: 30000

ToPort: 32767

IpProtocol: tcp

WorkerIngressVxlan:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
 Description: Vxlan packets
 FromPort: 4789
 ToPort: 4789
 IpProtocol: udp

WorkerIngressWorkerVxlan:

Type: AWS::EC2::SecurityGroupIngress
 Properties:
 GroupId: !GetAtt WorkerSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
 Description: Vxlan packets
 FromPort: 4789
 ToPort: 4789
 IpProtocol: udp

WorkerIngressInternal:

Type: AWS::EC2::SecurityGroupIngress
 Properties:
 GroupId: !GetAtt WorkerSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
 Description: Internal cluster communication
 FromPort: 9000
 ToPort: 9999
 IpProtocol: tcp

WorkerIngressWorkerInternal:

Type: AWS::EC2::SecurityGroupIngress
 Properties:
 GroupId: !GetAtt WorkerSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
 Description: Internal cluster communication
 FromPort: 9000
 ToPort: 9999
 IpProtocol: tcp

WorkerIngressKube:

Type: AWS::EC2::SecurityGroupIngress
 Properties:
 GroupId: !GetAtt WorkerSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
 Description: Kubernetes secure kubelet port
 FromPort: 10250
 ToPort: 10250
 IpProtocol: tcp

WorkerIngressWorkerKube:

Type: AWS::EC2::SecurityGroupIngress
 Properties:
 GroupId: !GetAtt WorkerSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
 Description: Internal Kubernetes communication
 FromPort: 10250
 ToPort: 10250
 IpProtocol: tcp

WorkerIngressIngressServices:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Kubernetes ingress services

FromPort: 30000

ToPort: 32767

IProtocol: tcp

WorkerIngressWorkerIngressServices:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Kubernetes ingress services

FromPort: 30000

ToPort: 32767

IProtocol: tcp

MasterIamRole:

Type: AWS::IAM::Role

Properties:**AssumeRolePolicyDocument:**

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Principal:

Service:

- "ec2.amazonaws.com"

Action:

- "sts:AssumeRole"

Policies:

- PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]

PolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Action: "ec2:*"

Resource: "*"

- Effect: "Allow"

Action: "elasticloadbalancing:*"

Resource: "*"

- Effect: "Allow"

Action: "iam:PassRole"

Resource: "*"

- Effect: "Allow"

Action: "s3:GetObject"

Resource: "*"

MasterInstanceProfile:

Type: "AWS::IAM::InstanceProfile"

Properties:**Roles:**

- Ref: "MasterIamRole"


```

WorkerIamRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: "Allow"
          Principal:
            Service:
              - "ec2.amazonaws.com"
          Action:
            - "sts:AssumeRole"
    Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "worker", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
            - Effect: "Allow"
              Action: "ec2:Describe*"
              Resource: "*"

```

```

WorkerInstanceProfile:
  Type: "AWS::IAM::InstanceProfile"
  Properties:
    Roles:
      - Ref: "WorkerIamRole"

```

```

Outputs:
  MasterSecurityGroupId:
    Description: Master Security Group ID
    Value: !GetAtt MasterSecurityGroup.GroupId

```

```

WorkerSecurityGroupId:
  Description: Worker Security Group ID
  Value: !GetAtt WorkerSecurityGroup.GroupId

```

```

MasterInstanceProfile:
  Description: Master IAM Instance Profile
  Value: !Ref MasterInstanceProfile

```

```

WorkerInstanceProfile:
  Description: Worker IAM Instance Profile
  Value: !Ref WorkerInstanceProfile

```

2.1.10. RHCOS AMIs for the AWS infrastructure

You must use a valid Red Hat Enterprise Linux CoreOS (RHCOS) AMI for your Amazon Web Services (AWS) zone for your OpenShift Container Platform nodes.

Table 2.1. RHCOS AMIs

AWS zone	AWS AMI
ap-northeast-1	ami-0c63b39219b8123e5
ap-northeast-2	ami-073cba0913d2250a4
ap-south-1	ami-0270be11430101040
ap-southeast-1	ami-06eb9d35ede4f08a3
ap-southeast-2	ami-0d980796ce258b5d5
ca-central-1	ami-0f907257d1686e3f7
eu-central-1	ami-02fdd627029c0055b
eu-west-1	ami-0d4839574724ed3fa
eu-west-2	ami-053073b95aa285347
eu-west-3	ami-09deb5deb6567bcd5
sa-east-1	ami-068a2000546e1889d
us-east-1	ami-046fe691f52a953f9
us-east-2	ami-0649fd5d42859bdfc
us-west-1	ami-0c1d2b5606111ac8c
us-west-2	ami-00745fcbb14a863ed

2.1.11. Creating the bootstrap node in AWS

You must create the bootstrap node in Amazon Web Services (AWS) to use during OpenShift Container Platform cluster initialization. The easiest way to create this node is to modify the provided CloudFormation template.



NOTE

If you do not use the provided CloudFormation template to create your bootstrap node, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.

- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.
- Create and configure DNS, load balancers, and listeners in AWS.
- Create control plane and compute roles.

Procedure

1. Provide a location to serve the **bootstrap.ign** Ignition config file to your cluster. This file is located in your installation directory. One way to do this is to create an S3 bucket in your cluster's region and upload the Ignition config file to it.



IMPORTANT

The provided CloudFormation Template assumes that the Ignition config files for your cluster are served from an S3 bucket. If you choose to serve the files from another location, you must modify the templates.



NOTE

The bootstrap Ignition config file does contain secrets, like X.509 keys. The following steps provide basic security for the S3 bucket. To provide additional security, you can enable an S3 bucket policy to allow only certain users, such as the OpenShift IAM user, to access objects that the bucket contains. You can avoid S3 entirely and serve your bootstrap Ignition config file from any address that the bootstrap machine can reach.

- a. Create the bucket:

```
$ aws s3 mb s3://<cluster-name>-infra 1
```

1 <cluster-name>-infra is the bucket name.

- b. Upload the **bootstrap.ign** Ignition config file to the bucket:

```
$ aws s3 cp bootstrap.ign s3://<cluster-name>-infra/bootstrap.ign
```

- c. Verify that the file uploaded:

```
$ aws s3 ls s3://<cluster-name>-infra/
2019-04-03 16:15:16    314878 bootstrap.ign
```

2. Create a JSON file that contains the parameter values that the template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
]
```

```

{
  "ParameterKey": "RhcosAmi", ❸
  "ParameterValue": "ami-<random_string>" ❹
},
{
  "ParameterKey": "AllowedBootstrapSshCidr", ❺
  "ParameterValue": "0.0.0.0/0" ❻
},
{
  "ParameterKey": "PublicSubnet", ❼
  "ParameterValue": "subnet-<random_string>" ❽
},
{
  "ParameterKey": "MasterSecurityGroupId", ❾
  "ParameterValue": "sg-<random_string>" ❿
},
{
  "ParameterKey": "VpcId", ⓫
  "ParameterValue": "vpc-<random_string>" ⓬
},
{
  "ParameterKey": "BootstrapIgnitionLocation", ⓭
  "ParameterValue": "s3://<bucket_name>/bootstrap.ign" ⓮
},
{
  "ParameterKey": "AutoRegisterELB", ⓯
  "ParameterValue": "yes" ⓰
},
{
  "ParameterKey": "RegisterNlbTargetsLambdaArn", ⓱
  "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbTargets-<random_string>" ⓲
},
{
  "ParameterKey": "ExternalApiTargetGroupArn", ⓳
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" ⓴
},
{
  "ParameterKey": "InternalApiTargetGroupArn", ⓵
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" ⓶
},
{
  "ParameterKey": "InternalServiceTargetGroupArn", ⓷
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" ⓸
}
}
]

```

- ❶ The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.

- 2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
 - 3 Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the bootstrap node.
 - 4 Specify a valid **AWS::EC2::Image::Id** value.
 - 5 CIDR block to allow SSH access to the bootstrap node.
 - 6 Specify a CIDR block in the format **x.x.x.x/16-24**.
 - 7 The public subnet that is associated with your VPC to launch the bootstrap node into.
 - 8 Specify the **PublicSubnetIds** value from the output of the CloudFormation template for the VPC.
 - 9 The master security group ID (for registering temporary rules)
 - 10 Specify the **MasterSecurityGroupId** value from the output of the CloudFormation template for the security group and roles.
 - 11 The VPC created resources will belong to.
 - 12 Specify the **VpcId** value from the output of the CloudFormation template for the VPC.
 - 13 Location to fetch bootstrap Ignition config file from.
 - 14 Specify the S3 bucket and file name in the form **s3://<bucket_name>/bootstrap.ign**.
 - 15 Whether or not to register a network load balancer (NLB).
 - 16 Specify **yes** or **no**. If you specify **yes**, you must provide a Lambda Amazon Resource Name (ARN) value.
 - 17 The ARN for NLB IP target registration lambda group.
 - 18 Specify the **RegisterNlbTargetsLambda** value from the output of the CloudFormation template for DNS and load balancing.
 - 19 The ARN for external API load balancer target group.
 - 20 Specify the **ExternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
 - 21 The ARN for internal API load balancer target group.
 - 22 Specify the **InternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
 - 23 The ARN for internal service load balancer target group.
 - 24 Specify the **InternalServiceTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
3. Copy the template from the **CloudFormation template for the bootstrap machines** section of this topic and save it as a YAML file on your computer. This template describes the bootstrap machine that your cluster requires.

4. Launch the template:

**IMPORTANT**

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> ❶
  --template-body file://<template>.yaml ❷
  --parameters file://<parameters>.json ❸
  --capabilities CAPABILITY_NAMED_IAM
```

- ❶ **<name>** is the name for the CloudFormation stack, such as **cluster-bootstrap**. You need the name of this stack if you remove the cluster.
- ❷ **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- ❸ **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

5. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

Bootstrap Instanceld	The bootstrap Instance ID.
Bootstrap PublicIp	The bootstrap node public IP address.
Bootstrap PrivateIp	The bootstrap node private IP address.

2.1.11.1. CloudFormation template for the bootstrap machine

You can use the following CloudFormation template to deploy the bootstrap machine that you need for your OpenShift Container Platform cluster.

AWSTemplateFormatVersion: 2010-09-09

Description: Template for OpenShift Cluster Bootstrap (EC2 Instance, Security Groups and IAM)

Parameters:

InfrastructureName:

AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-_]){0,26}\$

MaxLength: 27

MinLength: 1

ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a

maximum of 27 characters.

Description: A short, unique cluster ID used to tag cloud resources and identify items owned or used by the cluster.

Type: String

RhcosAmi:

Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.

Type: AWS::EC2::Image::Id

AllowedBootstrapSshCidr:

AllowedPattern: `^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\V([0-9]|1[0-9]|2[0-9]|3[0-2]))$`

ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/0-32.

Default: 0.0.0.0/0

Description: CIDR block to allow SSH access to the bootstrap node.

Type: String

PublicSubnet:

Description: The public subnet to launch the bootstrap node into.

Type: AWS::EC2::Subnet::Id

MasterSecurityGroupId:

Description: The master security group ID for registering temporary rules.

Type: AWS::EC2::SecurityGroup::Id

VpcId:

Description: The VPC-scoped resources will belong to this VPC.

Type: AWS::EC2::VPC::Id

BootstrapIgnitionLocation:

Default: s3://my-s3-bucket/bootstrap.ign

Description: Ignition config file location.

Type: String

AutoRegisterELB:

Default: "yes"

AllowedValues:

- "yes"
- "no"

Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?

Type: String

RegisterNlbIpTargetsLambdaArn:

Description: ARN for NLB IP target registration lambda.

Type: String

ExternalApiTargetGroupArn:

Description: ARN for external API load balancer target group.

Type: String

InternalApiTargetGroupArn:

Description: ARN for internal API load balancer target group.

Type: String

InternalServiceTargetGroupArn:

Description: ARN for internal service load balancer target group.

Type: String

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:
 - default: "Cluster Information"

Parameters:

- InfrastructureName

- Label:
 - default: "Host Information"

```

Parameters:
- RhcosAmi
- BootstrapIgnitionLocation
- MasterSecurityGroupId
- Label:
  default: "Network Configuration"
Parameters:
- VpcId
- AllowedBootstrapSshCidr
- PublicSubnet
- Label:
  default: "Load Balancer Automation"
Parameters:
- AutoRegisterELB
- RegisterNlbTargetsLambdaArn
- ExternalApiTargetGroupArn
- InternalApiTargetGroupArn
- InternalServiceTargetGroupArn
ParameterLabels:
InfrastructureName:
  default: "Infrastructure Name"
VpcId:
  default: "VPC ID"
AllowedBootstrapSshCidr:
  default: "Allowed SSH Source"
PublicSubnet:
  default: "Public Subnet"
RhcosAmi:
  default: "Red Hat Enterprise Linux CoreOS AMI ID"
BootstrapIgnitionLocation:
  default: "Bootstrap Ignition Source"
MasterSecurityGroupId:
  default: "Master Security Group ID"
AutoRegisterELB:
  default: "Use Provided ELB Automation"

Conditions:
DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]

Resources:
BootstrapIamRole:
Type: AWS::IAM::Role
Properties:
AssumeRolePolicyDocument:
Version: "2012-10-17"
Statement:
- Effect: "Allow"
Principal:
Service:
- "ec2.amazonaws.com"
Action:
- "sts:AssumeRole"
Path: "/"
Policies:
- PolicyName: !Join ["-", [!Ref InfrastructureName, "bootstrap", "policy"]]
PolicyDocument:

```



```

Version: "2012-10-17"
Statement:
- Effect: "Allow"
  Action: "ec2:Describe*"
  Resource: "*"
- Effect: "Allow"
  Action: "ec2:AttachVolume"
  Resource: "*"
- Effect: "Allow"
  Action: "ec2:DetachVolume"
  Resource: "*"
- Effect: "Allow"
  Action: "s3:GetObject"
  Resource: "*"

```

```

BootstrapInstanceProfile:
Type: "AWS::IAM::InstanceProfile"
Properties:
  Path: "/"
  Roles:
  - Ref: "BootstrapIamRole"

```

```

BootstrapSecurityGroup:
Type: AWS::EC2::SecurityGroup
Properties:
  GroupDescription: Cluster Bootstrap Security Group
  SecurityGroupIngress:
  - IpProtocol: tcp
    FromPort: 22
    ToPort: 22
    CidrIp: !Ref AllowedBootstrapSshCidr
  - IpProtocol: tcp
    ToPort: 19531
    FromPort: 19531
    CidrIp: 0.0.0.0/0
  VpcId: !Ref VpcId

```

```

BootstrapInstance:
Type: AWS::EC2::Instance
Properties:
  ImageId: !Ref RhcosAmi
  IamInstanceProfile: !Ref BootstrapInstanceProfile
  InstanceType: "i3.large"
  NetworkInterfaces:
  - AssociatePublicIpAddress: "true"
    DeviceIndex: "0"
    GroupSet:
    - !Ref "BootstrapSecurityGroup"
    - !Ref "MasterSecurityGroup"
    SubnetId: !Ref "PublicSubnet"
  UserData:
    Fn::Base64: !Sub
      - '{"ignition":{"config":{"replace":{"source":"${S3Loc}","verification":{}}},"timeouts":
        {}, "version":"2.1.0"},"networkd":{,"passwd":{,"storage":{,"systemd":{}}}'
      - {
        S3Loc: !Ref BootstrapIgnitionLocation

```

}

RegisterBootstrapApiTarget:

Condition: DoRegistration

Type: Custom::NLBRegister

Properties:

ServiceToken: !Ref RegisterNlbTargetsLambdaArn

TargetArn: !Ref ExternalApiTargetGroupArn

TargetIp: !GetAtt BootstrapInstance.PrivateIp

RegisterBootstrapInternalApiTarget:

Condition: DoRegistration

Type: Custom::NLBRegister

Properties:

ServiceToken: !Ref RegisterNlbTargetsLambdaArn

TargetArn: !Ref InternalApiTargetGroupArn

TargetIp: !GetAtt BootstrapInstance.PrivateIp

RegisterBootstrapInternalServiceTarget:

Condition: DoRegistration

Type: Custom::NLBRegister

Properties:

ServiceToken: !Ref RegisterNlbTargetsLambdaArn

TargetArn: !Ref InternalServiceTargetGroupArn

TargetIp: !GetAtt BootstrapInstance.PrivateIp

Outputs:

BootstrapInstanceId:

Description: Bootstrap Instance ID.

Value: !Ref BootstrapInstance

BootstrapPublicIp:

Description: The bootstrap node public IP address.

Value: !GetAtt BootstrapInstance.PublicIp

BootstrapPrivateIp:

Description: The bootstrap node private IP address.

Value: !GetAtt BootstrapInstance.PrivateIp

2.1.12. Creating the control plane machines in AWS

You must create the control plane machines in Amazon Web Services (AWS) for your cluster to use. The easiest way to create these nodes is to modify the provided CloudFormation template.



NOTE

If you do not use the provided CloudFormation template to create your control plane nodes, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.

- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.
- Create and configure DNS, load balancers, and listeners in AWS.
- Create control plane and compute roles.
- Create the bootstrap machine.

Procedure

1. Create a JSON file that contains the parameter values that the template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcOsAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AutoRegisterDNS", 5
    "ParameterValue": "yes" 6
  },
  {
    "ParameterKey": "PrivateHostedZoneId", 7
    "ParameterValue": "<random_string>" 8
  },
  {
    "ParameterKey": "PrivateHostedZoneName", 9
    "ParameterValue": "mycluster.example.com" 10
  },
  {
    "ParameterKey": "Master0Subnet", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
    "ParameterKey": "Master1Subnet", 13
    "ParameterValue": "subnet-<random_string>" 14
  },
  {
    "ParameterKey": "Master2Subnet", 15
    "ParameterValue": "subnet-<random_string>" 16
  },
  {
    "ParameterKey": "MasterSecurityGroupID", 17
    "ParameterValue": "sg-<random_string>" 18
  },
  {
    "ParameterKey": "IgnitionLocation", 19
  }
]
```

```

    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/master"
  },
  {
    "ParameterKey": "CertificateAuthorities",
    "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz=="
  },
  {
    "ParameterKey": "MasterInstanceProfileName",
    "ParameterValue": "<roles_stack>-MasterInstanceProfile-<random_string>"
  },
  {
    "ParameterKey": "MasterInstanceType",
    "ParameterValue": "m4.xlarge"
  },
  {
    "ParameterKey": "AutoRegisterELB",
    "ParameterValue": "yes"
  },
  {
    "ParameterKey": "RegisterNLbpTargetsLambdaArn",
    "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNLbpTargets-<random_string>"
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn",
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>"
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn",
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>"
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn",
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>"
  }
]

```

- 1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- 2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
- 3 Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the control plane machines.
- 4 Specify an **AWS::EC2::Image::Id** value.
- 5 Whether or not to perform DNS etcd registration.

- 6 Specify **yes** or **no**. If you specify **yes**, you must provide Hosted Zone information.
- 7 The Route53 private zone ID to register the etcd targets with.
- 8 Specify the **PrivateHostedZoneId** value from the output of the CloudFormation template for DNS and load balancing.
- 9 The Route53 zone to register the targets with.
- 10 Specify **<cluster_name>.<domain_name>** where **<domain_name>** is the Route53 base domain that you used when you generated **install-config.yaml** file for the cluster. Do not include the trailing period (.) that is displayed in the AWS console.
- 11 13 15 A subnet, preferably private, to launch the control plane machines on.
- 12 14 16 Specify a subnet from the **PrivateSubnets** value from the output of the CloudFormation template for DNS and load balancing.
- 17 The master security group ID to associate with master nodes.
- 18 Specify the **MasterSecurityGroupID** value from the output of the CloudFormation template for the security group and roles.
- 19 The location to fetch control plane Ignition config file from.
- 20 Specify the generated Ignition config file location, https://api-int.<cluster_name>.<domain_name>:22623/config/master.
- 21 The base64 encoded certificate authority string to use.
- 22 Specify the value from the **master.ign** file that is in the installation directory. This value is the long string with the format **data:text/plain;charset=utf-8;base64,ABC...xYz==**.
- 23 The IAM profile to associate with master nodes.
- 24 Specify the **MasterInstanceProfile** parameter value from the output of the CloudFormation template for the security group and roles.
- 25 The type of AWS instance to use for the control plane machines.
- 26 Allowed values:
 - **m4.xlarge**
 - **m4.2xlarge**
 - **m4.4xlarge**
 - **m4.8xlarge**
 - **m4.10xlarge**
 - **m4.16xlarge**
 - **c4.2xlarge**
 - **c4.4xlarge**

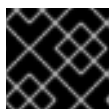
- **c4.8xlarge**
- **r4.xlarge**
- **r4.2xlarge**
- **r4.4xlarge**
- **r4.8xlarge**
- **r4.16xlarge**



IMPORTANT

If **m4** instance types are not available in your region, such as with **eu-west-3**, specify an **m5** type, such as **m5.xlarge**, instead.

27. Whether or not to register a network load balancer (NLB).
 28. Specify **yes** or **no**. If you specify **yes**, you must provide a Lambda Amazon Resource Name (ARN) value.
 29. The ARN for NLB IP target registration lambda group.
 30. Specify the **RegisterNlbTargetsLambda** value from the output of the CloudFormation template for DNS and load balancing.
 31. The ARN for external API load balancer target group.
 32. Specify the **ExternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
 33. The ARN for internal API load balancer target group.
 34. Specify the **InternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
 35. The ARN for internal service load balancer target group.
 36. Specify the **InternalServiceTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
2. Copy the template from the **CloudFormation template for control plane machines** section of this topic and save it as a YAML file on your computer. This template describes the control plane machines that your cluster requires.
 3. If you specified an **m5** instance type as the value for **MasterInstanceType**, add that instance type to the **MasterInstanceType.AllowedValues** parameter in the CloudFormation template.
 4. Launch the template:



IMPORTANT

You must enter the command on a single line.

■

```
$ aws cloudformation create-stack --stack-name <name> ❶
    --template-body file://<template>.yaml ❷
    --parameters file://<parameters>.json ❸
```

- ❶ **<name>** is the name for the CloudFormation stack, such as **cluster-control-plane**. You need the name of this stack if you remove the cluster.
- ❷ **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- ❸ **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

5. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

2.1.12.1. CloudFormation template for control plane machines

You can use the following CloudFormation template to deploy the control plane machines that you need for your OpenShift Container Platform cluster.

AWSTemplateFormatVersion: 2010-09-09

Description: Template for OpenShift Cluster Node Launch (EC2 master instances)

Parameters:

InfrastructureName:

AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-_]){0,26}\$

MaxLength: 27

MinLength: 1

ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of 27 characters.

Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.

Type: String

RhcosAmi:

Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.

Type: AWS::EC2::Image::Id

AutoRegisterDNS:

Default: "yes"

AllowedValues:

- "yes"

- "no"

Description: Do you want to invoke DNS etcd registration, which requires Hosted Zone information?

Type: String

PrivateHostedZoneId:

Description: The Route53 private zone ID to register the etcd targets with, such as Z21XYZABCZ2A4.

Type: String

PrivateHostedZoneName:

Description: The Route53 zone to register the targets with, such as cluster.example.com. Omit the trailing period.

Type: String

Master0Subnet:

Description: The subnets, recommend private, to launch the master nodes into.

Type: AWS::EC2::Subnet::Id

Master1Subnet:

Description: The subnets, recommend private, to launch the master nodes into.

Type: AWS::EC2::Subnet::Id

Master2Subnet:

Description: The subnets, recommend private, to launch the master nodes into.

Type: AWS::EC2::Subnet::Id

MasterSecurityGroupId:

Description: The master security group ID to associate with master nodes.

Type: AWS::EC2::SecurityGroup::Id

IgnitionLocation:

Default: https://api-int.\$CLUSTER_NAME.\$DOMAIN:22623/config/master

Description: Ignition config file location.

Type: String

CertificateAuthorities:

Default: data:text/plain;charset=utf-8;base64,ABC...xYz==

Description: Base64 encoded certificate authority string to use.

Type: String

MasterInstanceProfileName:

Description: IAM profile to associate with master nodes.

Type: String

MasterInstanceType:

Default: m4.xlarge

Type: String

AllowedValues:

- "m4.xlarge"
- "m4.2xlarge"
- "m4.4xlarge"
- "m4.8xlarge"
- "m4.10xlarge"
- "m4.16xlarge"
- "c4.2xlarge"
- "c4.4xlarge"
- "c4.8xlarge"
- "r4.xlarge"
- "r4.2xlarge"
- "r4.4xlarge"
- "r4.8xlarge"
- "r4.16xlarge"

AutoRegisterELB:

Default: "yes"

AllowedValues:

- "yes"
- "no"

Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?

Type: String

RegisterNLBpTargetsLambdaArn:

Description: ARN for NLB IP target registration lambda. Supply the value from the cluster infrastructure or select "no" for AutoRegisterELB.

Type: String

ExternalApiTargetGroupArn:

Description: ARN for external API load balancer target group. Supply the value from the cluster infrastructure or select "no" for AutoRegisterELB.

Type: String

InternalApiTargetGroupArn:

Description: ARN for internal API load balancer target group. Supply the value from the cluster infrastructure or select "no" for AutoRegisterELB.

Type: String

InternalServiceTargetGroupArn:

Description: ARN for internal service load balancer target group. Supply the value from the cluster infrastructure or select "no" for AutoRegisterELB.

Type: String

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:

default: "Cluster Information"

Parameters:

- InfrastructureName

- Label:

default: "Host Information"

Parameters:

- MasterInstanceType

- RhcosAmi

- IgnitionLocation

- CertificateAuthorities

- MasterSecurityGroupId

- MasterInstanceProfileName

- Label:

default: "Network Configuration"

Parameters:

- VpcId

- AllowedBootstrapSshCidr

- Master0Subnet

- Master1Subnet

- Master2Subnet

- Label:

default: "DNS"

Parameters:

- AutoRegisterDNS

- PrivateHostedZoneName

- PrivateHostedZoneId

- Label:

default: "Load Balancer Automation"

Parameters:

- AutoRegisterELB

- RegisterNlbPTargetsLambdaArn

- ExternalApiTargetGroupArn

- InternalApiTargetGroupArn

- InternalServiceTargetGroupArn

ParameterLabels:

InfrastructureName:

default: "Infrastructure Name"

VpcId:

default: "VPC ID"

Master0Subnet:

default: "Master-0 Subnet"

Master1Subnet:

default: "Master-1 Subnet"

Master2Subnet:

```

    default: "Master-2 Subnet"
MasterInstanceType:
    default: "Master Instance Type"
MasterInstanceProfileName:
    default: "Master Instance Profile Name"
RhcosAmi:
    default: "Red Hat Enterprise Linux CoreOS AMI ID"
BootstrapIgnitionLocation:
    default: "Master Ignition Source"
CertificateAuthorities:
    default: "Ignition CA String"
MasterSecurityGroupId:
    default: "Master Security Group ID"
AutoRegisterDNS:
    default: "Use Provided DNS Automation"
AutoRegisterELB:
    default: "Use Provided ELB Automation"
PrivateHostedZoneName:
    default: "Private Hosted Zone Name"
PrivateHostedZoneId:
    default: "Private Hosted Zone ID"

```

Conditions:

```

DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]
DoDns: !Equals ["yes", !Ref AutoRegisterDNS]

```

Resources:

```

Master0:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref RhcosAmi
    BlockDeviceMappings:
      - DeviceName: /dev/xvda
    Ebs:
      VolumeSize: "120"
      VolumeType: "gp2"
    IamInstanceProfile: !Ref MasterInstanceProfileName
    InstanceType: !Ref MasterInstanceType
    NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
          - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master0Subnet"
    UserData:
      Fn::Base64: !Sub
        - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}],"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]},"timeouts":
{},"version":"2.2.0"},"networkd":{},"passwd":{},"storage":{},"systemd":{}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
  Tags:
    - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
      Value: "shared"

```

```

RegisterMaster0:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref ExternalApiTargetGroupArn
    TargetIp: !GetAtt Master0.PrivateIp

```

```

RegisterMaster0InternalApiTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref InternalApiTargetGroupArn
    TargetIp: !GetAtt Master0.PrivateIp

```

```

RegisterMaster0InternalServiceTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref InternalServiceTargetGroupArn
    TargetIp: !GetAtt Master0.PrivateIp

```

```

Master1:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref RhcosAmi
    BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
    IamInstanceProfile: !Ref MasterInstanceProfileName
    InstanceType: !Ref MasterInstanceType
    NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
          - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master1Subnet"
    UserData:
      Fn::Base64: !Sub
        - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}],"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]},"timeouts":
{},"version":"2.2.0"},"networkd":{"passwd":{},"storage":{},"systemd":{}}'
          - {
              SOURCE: !Ref IgnitionLocation,
              CA_BUNDLE: !Ref CertificateAuthorities,
            }
    Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

```

```

RegisterMaster1:

```

Condition: DoRegistration
 Type: Custom::NLBRegister
 Properties:
 ServiceToken: !Ref RegisterNlbTargetsLambdaArn
 TargetArn: !Ref ExternalApiTargetGroupArn
 TargetIp: !GetAtt Master1.PrivateIp

RegisterMaster1InternalApiTarget:
 Condition: DoRegistration
 Type: Custom::NLBRegister
 Properties:
 ServiceToken: !Ref RegisterNlbTargetsLambdaArn
 TargetArn: !Ref InternalApiTargetGroupArn
 TargetIp: !GetAtt Master1.PrivateIp

RegisterMaster1InternalServiceTarget:
 Condition: DoRegistration
 Type: Custom::NLBRegister
 Properties:
 ServiceToken: !Ref RegisterNlbTargetsLambdaArn
 TargetArn: !Ref InternalServiceTargetGroupArn
 TargetIp: !GetAtt Master1.PrivateIp

Master2:
 Type: AWS::EC2::Instance
 Properties:
 ImageId: !Ref RhcosAmi
 BlockDeviceMappings:
 - DeviceName: /dev/xvda
 Ebs:
 VolumeSize: "120"
 VolumeType: "gp2"
 IamInstanceProfile: !Ref MasterInstanceProfileName
 InstanceType: !Ref MasterInstanceType
 NetworkInterfaces:
 - AssociatePublicIpAddress: "false"
 DeviceIndex: "0"
 GroupSet:
 - !Ref "MasterSecurityGroup"
 SubnetId: !Ref "Master2Subnet"
 UserData:
 Fn::Base64: !Sub
 - '{"ignition":{"config":{"append":[{"source":"\${SOURCE}","verification":{} }],"security":{"tls":{"certificateAuthorities":[{"source":"\${CA_BUNDLE}","verification":{} }]},"timeouts":{"version":"2.2.0"},"networkd":{},"passwd":{},"storage":{},"systemd":{}}'
 - {
 SOURCE: !Ref IgnitionLocation,
 CA_BUNDLE: !Ref CertificateAuthorities,
 }
 Tags:
 - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
 Value: "shared"

RegisterMaster2:
 Condition: DoRegistration
 Type: Custom::NLBRegister

Properties:

ServiceToken: !Ref RegisterNlbTargetsLambdaArn
 TargetArn: !Ref ExternalApiTargetGroupArn
 TargetIp: !GetAtt Master2.PrivateIp

RegisterMaster2InternalApiTarget:

Condition: DoRegistration
 Type: Custom::NLBRegister
 Properties:
 ServiceToken: !Ref RegisterNlbTargetsLambdaArn
 TargetArn: !Ref InternalApiTargetGroupArn
 TargetIp: !GetAtt Master2.PrivateIp

RegisterMaster2InternalServiceTarget:

Condition: DoRegistration
 Type: Custom::NLBRegister
 Properties:
 ServiceToken: !Ref RegisterNlbTargetsLambdaArn
 TargetArn: !Ref InternalServiceTargetGroupArn
 TargetIp: !GetAtt Master2.PrivateIp

EtcdSrvRecords:

Condition: DoDns
 Type: AWS::Route53::RecordSet
 Properties:
 HostedZoneId: !Ref PrivateHostedZoneId
 Name: !Join [".", ["_etcd-server-ssl._tcp", !Ref PrivateHostedZoneName]]
 ResourceRecords:
 - !Join [
 " ",
 ["0 10 2380", !Join [".", ["etcd-0", !Ref PrivateHostedZoneName]]],
]
 - !Join [
 " ",
 ["0 10 2380", !Join [".", ["etcd-1", !Ref PrivateHostedZoneName]]],
]
 - !Join [
 " ",
 ["0 10 2380", !Join [".", ["etcd-2", !Ref PrivateHostedZoneName]]],
]
 TTL: 60
 Type: SRV

Etcd0Record:

Condition: DoDns
 Type: AWS::Route53::RecordSet
 Properties:
 HostedZoneId: !Ref PrivateHostedZoneId
 Name: !Join [".", ["etcd-0", !Ref PrivateHostedZoneName]]
 ResourceRecords:
 - !GetAtt Master0.PrivateIp
 TTL: 60
 Type: A

Etcd1Record:

Condition: DoDns

```
Type: AWS::Route53::RecordSet
Properties:
  HostedZoneId: !Ref PrivateHostedZoneId
  Name: !Join [".", ["etcd-1", !Ref PrivateHostedZoneName]]
  ResourceRecords:
    - !GetAtt Master1.PrivateIp
  TTL: 60
  Type: A
```

```
Etcd2Record:
  Condition: DoDns
  Type: AWS::Route53::RecordSet
Properties:
  HostedZoneId: !Ref PrivateHostedZoneId
  Name: !Join [".", ["etcd-2", !Ref PrivateHostedZoneName]]
  ResourceRecords:
    - !GetAtt Master2.PrivateIp
  TTL: 60
  Type: A
```

```
Outputs:
PrivateIPs:
  Description: The control-plane node private IP addresses.
  Value:
    !Join [
      ",",
      [!GetAtt Master0.PrivateIp, !GetAtt Master1.PrivateIp, !GetAtt Master2.PrivateIp]
    ]
```

2.1.13. Initializing the bootstrap node on AWS with user-provisioned infrastructure

After you create all of the required infrastructure in Amazon Web Services (AWS), you can install the cluster.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.
- Create and configure DNS, load balancers, and listeners in AWS.
- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.
- If you plan to manually manage the worker machines, create the worker machines.

Procedure

1. Change to the directory that contains the installation program and run the following command:

■

```
$ ./openshift-install wait-for bootstrap-complete --dir=<installation_directory> \ 1
--log-level debug 2
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2** Optionally, include the **--log-level debug** option to view installation details.

If the command exits without a **FATAL** warning, your production control plane has initialized.

2.1.13.1. Creating the worker nodes in AWS

You can create worker nodes in Amazon Web Services (AWS) for your cluster to use. The easiest way to manually create these nodes is to modify the provided CloudFormation template.



IMPORTANT

The CloudFormation template creates a stack that represents one worker machine. You must create a stack for each worker machine.



NOTE

If you do not use the provided CloudFormation template to create your worker nodes, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.
- Create and configure DNS, load balancers, and listeners in AWS.
- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.

Procedure

1. Create a JSON file that contains the parameter values that the CloudFormation template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
```

```

    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "Subnet", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "WorkerSecurityGroupld", 7
    "ParameterValue": "sg-<random_string>" 8
  },
  {
    "ParameterKey": "IgnitionLocation", 9
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/worker"
10
  },
  {
    "ParameterKey": "CertificateAuthorities", 11
    "ParameterValue": "" 12
  },
  {
    "ParameterKey": "WorkerInstanceProfileName", 13
    "ParameterValue": "" 14
  },
  {
    "ParameterKey": "WorkerInstanceType", 15
    "ParameterValue": "m4.large" 16
  }
]

```

- 1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- 2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
- 3 Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the worker nodes.
- 4 Specify an **AWS::EC2::Image::Id** value.
- 5 A subnet, preferably private, to launch the worker nodes on.
- 6 Specify a subnet from the **PrivateSubnets** value from the output of the CloudFormation template for DNS and load balancing.
- 7 The worker security group ID to associate with worker nodes.
- 8 Specify the **WorkerSecurityGroupld** value from the output of the CloudFormation template for the security group and roles.
- 9 The location to fetch bootstrap Ignition config file from.
- 10 Specify the generated Ignition config location, https://api-int.<cluster_name>.<domain_name>:22623/config/worker.

- 11 Base64 encoded certificate authority string to use.
- 12 Specify the value from the **worker.ign** file that is in the installation directory. This value is the long string with the format **data:text/plain;charset=utf-8;base64,ABC...xYz==**.
- 13 The IAM profile to associate with worker nodes.
- 14 Specify the **WorkerInstanceProfile** parameter value from the output of the CloudFormation template for the security group and roles.
- 15 The type of AWS instance to use for the control plane machines.
- 16 Allowed values:
 - **m4.large**
 - **m4.xlarge**
 - **m4.2xlarge**
 - **m4.4xlarge**
 - **m4.8xlarge**
 - **m4.10xlarge**
 - **m4.16xlarge**
 - **c4.large**
 - **c4.xlarge**
 - **c4.2xlarge**
 - **c4.4xlarge**
 - **c4.8xlarge**
 - **r4.large**
 - **r4.xlarge**
 - **r4.2xlarge**
 - **r4.4xlarge**
 - **r4.8xlarge**
 - **r4.16xlarge**



IMPORTANT

If **m4** instance types are not available in your region, such as with **eu-west-3**, use **m5** types instead.

2. Copy the template from the **CloudFormation template for worker machines** section of this topic and save it as a YAML file on your computer. This template describes the networking objects and load balancers that your cluster requires.
3. If you specified an **m5** instance type as the value for **WorkerInstanceType**, add that instance type to the **WorkerInstanceType.AllowedValues** parameter in the CloudFormation template.
4. Create a worker stack.
 - a. Launch the template:

**IMPORTANT**

You must enter the command on a single line.

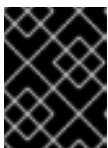
```
$ aws cloudformation create-stack --stack-name <name> 1
--template-body file://<template>.yaml \ 2
--parameters file://<parameters>.json 3
```

- 1** **<name>** is the name for the CloudFormation stack, such as **cluster-workers**. You need the name of this stack if you remove the cluster.
- 2** **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- 3** **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

- b. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

5. Continue to create worker stacks until you have created enough worker Machines for your cluster.

**IMPORTANT**

You must create at least two worker machines, so you must create at least two stacks that use this CloudFormation template.

2.1.13.1.1. CloudFormation template for worker machines

You can use the following CloudFormation template to deploy the worker machines that you need for your OpenShift Container Platform cluster.

AWSTemplateFormatVersion: [2010-09-09](#)

Description: Template for OpenShift Cluster Node Launch (EC2 worker instance)

Parameters:

InfrastructureName:

AllowedPattern: `^[a-zA-Z][a-zA-Z0-9\-\]{0,26}$`

MaxLength: [27](#)

MinLength: [1](#)

ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of 27 characters.

Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.

Type: String

RhcosAmi:

Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.

Type: AWS::EC2::Image::Id

Subnet:

Description: The subnets, recommend private, to launch the master nodes into.

Type: AWS::EC2::Subnet::Id

WorkerSecurityGroupId:

Description: The master security group ID to associate with master nodes.

Type: AWS::EC2::SecurityGroup::Id

IgnitionLocation:

Default: https://api-int.\$CLUSTER_NAME.\$DOMAIN:22623/config/worker

Description: Ignition config file location.

Type: String

CertificateAuthorities:

Default: data:text/plain;charset=utf-8;base64,ABC...xYZ==

Description: Base64 encoded certificate authority string to use.

Type: String

WorkerInstanceProfileName:

Description: IAM profile to associate with master nodes.

Type: String

WorkerInstanceType:

Default: m4.large

Type: String

AllowedValues:

- "m4.large"
- "m4.xlarge"
- "m4.2xlarge"
- "m4.4xlarge"
- "m4.8xlarge"
- "m4.10xlarge"
- "m4.16xlarge"
- "c4.large"
- "c4.xlarge"
- "c4.2xlarge"
- "c4.4xlarge"
- "c4.8xlarge"
- "r4.large"
- "r4.xlarge"
- "r4.2xlarge"
- "r4.4xlarge"
- "r4.8xlarge"
- "r4.16xlarge"

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:
 - default: "Cluster Information"

Parameters:

- InfrastructureName
- Label:
 - default: "Host Information"

```

Parameters:
- WorkerInstanceType
- RhcosAmi
- IgnitionLocation
- CertificateAuthorities
- WorkerSecurityGroupId
- WorkerInstanceProfileName
- Label:
  default: "Network Configuration"
Parameters:
- Subnet
ParameterLabels:
Subnet:
  default: "Subnet"
InfrastructureName:
  default: "Infrastructure Name"
WorkerInstanceType:
  default: "Worker Instance Type"
WorkerInstanceProfileName:
  default: "Worker Instance Profile Name"
RhcosAmi:
  default: "Red Hat Enterprise Linux CoreOS AMI ID"
IgnitionLocation:
  default: "Worker Ignition Source"
CertificateAuthorities:
  default: "Ignition CA String"
WorkerSecurityGroupId:
  default: "Worker Security Group ID"

Resources:
Worker0:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref RhcosAmi
    BlockDeviceMappings:
      - DeviceName: /dev/xvda
    Ebs:
      VolumeSize: "120"
      VolumeType: "gp2"
    IamInstanceProfile: !Ref WorkerInstanceProfileName
    InstanceType: !Ref WorkerInstanceType
    NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
          - !Ref "WorkerSecurityGroupId"
        SubnetId: !Ref "Subnet"
    UserData:
      Fn::Base64: !Sub
        - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}],"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]},"timeouts":
{},"version":"2.2.0"},"networkd":{},"passwd":{},"storage":{},"systemd":{}}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }

```

Tags:

- Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
Value: "shared"

Outputs:

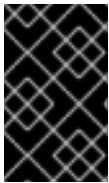
PrivateIP:

Description: The compute node private IP address.

Value: !GetAtt Worker0.PrivateIp

2.1.14. Installing the OpenShift Command-line Interface

You can download and install the OpenShift Command-line Interface (CLI), commonly known as **oc**.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.1. You must download and install the new version of **oc**.

Procedure

1. From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**.
2. From the site that is displayed, download the compressed file for your operating system.



NOTE

You can install **oc** on Linux, Windows, or macOS.

3. Extract the compressed file and place it in a directory that is on your PATH.

2.1.15. Logging into the cluster

You can log into your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

```
$ oc whoami  
system:admin
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2.1.16. Approving the CSRs for your machines

When you add machines to a cluster, two pending certificates signing request (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	63m	v1.13.4+b626c2fe1
master-1	Ready	master	63m	v1.13.4+b626c2fe1
master-2	Ready	master	64m	v1.13.4+b626c2fe1
worker-0	NotReady	worker	76s	v1.13.4+b626c2fe1
worker-1	NotReady	worker	70s	v1.13.4+b626c2fe1

The output lists all of the machines that you created.

2. Review the pending certificate signing requests (CSRs) and ensure that the you see a client and server request with **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

NAME	AGE	REQUESTOR	CONDITION
csr-8b2br	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrap	Pending 1
csr-8vnps	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrap	Pending
csr-bfd72	5m26s	system:node:ip-10-0-50-126.us-east-2.compute.internal	Pending 2
csr-c57lv	5m26s	system:node:ip-10-0-95-157.us-east-2.compute.internal	Pending
...			

- 1 A client request CSR.

- 2 A server request CSR.

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After you approve the initial CSRs, the subsequent node client CSRs are automatically approved by the cluster **kube-controller-manager**. You must implement a method of automatically approving the kubelet serving certificate requests.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- If all the CSRs are valid, approve them all by running the following command:

```
$ oc get csr -ojson | jq -r '.items[] | select(.status == {} ) | .metadata.name' | xargs oc adm certificate approve
```

2.1.17. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

- Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED
SINCE				
authentication	4.1.0	True	False	False 69s
cloud-credential	4.1.0	True	False	False 12m
cluster-autoscaler	4.1.0	True	False	False 11m
console	4.1.0	True	False	False 46s
dns	4.1.0	True	False	False 11m
image-registry	4.1.0	False	True	False 5m26s
ingress	4.1.0	True	False	False 5m36s
kube-apiserver	4.1.0	True	False	False 8m53s
kube-controller-manager	4.1.0	True	False	False 7m24s
kube-scheduler	4.1.0	True	False	False 12m
machine-api	4.1.0	True	False	False 12m
machine-config	4.1.0	True	False	False 7m36s

marketplace	4.1.0	True	False	False	7m54m
monitoring	4.1.0	True	False	False	7h54s
network	4.1.0	True	False	False	5m9s
node-tuning	4.1.0	True	False	False	11m
openshift-apiserver	4.1.0	True	False	False	11m
openshift-controller-manager	4.1.0	True	False	False	5m943s
openshift-samples	4.1.0	True	False	False	3m55s
operator-lifecycle-manager	4.1.0	True	False	False	11m
operator-lifecycle-manager-catalog	4.1.0	True	False	False	11m
service-ca	4.1.0	True	False	False	11m
service-catalog-apiserver	4.1.0	True	False	False	5m26s
service-catalog-controller-manager	4.1.0	True	False	False	5m25s
storage	4.1.0	True	False	False	5m30s

2. Configure the Operators that are not available.

2.1.17.1. Image registry storage configuration

If the **image-registry** Operator is not available, you must configure storage for it. Instructions for both configuring a PersistentVolume, which is required for production clusters, and for configuring an empty directory as the storage location, which is available for only non-production clusters, are shown.

2.1.17.1.1. Configuring registry storage for AWS with user-provisioned infrastructure

During installation, your cloud credentials are sufficient to create an S3 bucket and the Registry Operator will automatically configure storage.

If the Registry Operator cannot create an S3 bucket, and automatically configure storage, you can create a S3 bucket and configure storage with the following procedure.

Prerequisites

- A cluster on AWS with user-provisioned infrastructure.

Procedure

Use the following procedure if the Registry Operator cannot create an S3 bucket and automatically configure storage.

1. Set up a [Bucket Lifecycle Policy](#) to abort incomplete multipart uploads that are one day old.
2. Fill in the storage configuration in **configs.imageregistry.operator.openshift.io/cluster**:

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster

storage:
  s3:
    bucket: <bucket-name>
    region: <region-name>
```


**WARNING**

To secure your registry images in AWS, [block public access](#) to the S3 bucket.

2.1.17.1.2. Configuring storage for the image registry in non-production clusters

You must configure storage for the image registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

Procedure

- To set the image registry storage to an empty directory:

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec": {"storage":{"emptyDir":{}}}'
```

**WARNING**

Configure this option for only non-production clusters.

If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

Wait a few minutes and run the command again.

2.1.18. Completing an AWS installation on user-provisioned infrastructure

After you start the OpenShift Container Platform installation on Amazon Web Service (AWS) user-provisioned infrastructure, remove the bootstrap node, reconcile the default Machine and MachineSet definitions, and delete unused nodes

Prerequisites

- Deploy the bootstrap node for an OpenShift Container Platform cluster on user-provisioned AWS infrastructure.
- Install the **oc** CLI and log in.

Procedure

1. Delete the bootstrap resources. If you used the CloudFormation template, [delete its stack](#):

```
$ aws cloudformation delete-stack --stack-name <name> 1
```

1 **<name>** is the name of your bootstrap stack.

2. Complete the cluster installation:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of telemetry](#).

CHAPTER 3. INSTALLING ON BARE METAL

3.1. INSTALLING A CLUSTER ON BARE METAL

In OpenShift Container Platform version 4.1, 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.

Prerequisites

- Provision [persistent storage](#) for your cluster. To deploy a private image registry, your storage must provide ReadWriteMany access modes.
- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall, you must [configure it to access Red Hat Insights](#) .

3.1.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.1, Telemetry is the component that provides metrics about cluster health and the success of updates. To perform subscription management, including legally entitling your purchase from Red Hat, you must use the Telemetry service and access the [OpenShift Infrastructure Providers](#) page.

Because there is no disconnected subscription management, you cannot both opt out of sending data back to Red Hat and entitle your purchase. Support for disconnected subscription management might be added in future releases of OpenShift Container Platform



IMPORTANT

Your machines must have direct internet access to install the cluster.

You must have internet access to:

- Access the [OpenShift Infrastructure Providers](#) page to download the installation program
- Access [quay.io](#) to obtain the packages that are required to install your cluster
- Obtain the packages that are required to perform cluster updates
- Access [Red Hat's software as a service page](#) to perform subscription management

3.1.2. Machine requirements for a cluster with user-provisioned infrastructure

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

3.1.2.1. Required machines

The smallest OpenShift Container Platform clusters require the following hosts:

- One bootstrap machine
- Three control plane, or master, machines
- At least two compute, or worker, machines



NOTE

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.



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.

3.1.2.2. Network connectivity requirements

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config files from the Machine Config Server. During the initial boot, the machines require a DHCP server in order to establish a network connection to download their Ignition config files. After the initial boot, the machines can be configured to use static IP addresses.

3.1.2.3. Minimum resource requirements

Each cluster machine must meet the following minimum requirements:

Machine	Operating System	vCPU	RAM	Storage
Bootstrap	RHCOS	4	16 GB	120 GB
Control plane	RHCOS	4	16 GB	120 GB
Compute	RHCOS or RHEL 7.6	2	8 GB	120 GB

3.1.2.4. 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.

3.1.3. Creating the user-provisioned infrastructure

Before you deploy a OpenShift Container Platform cluster that uses user-provisioned infrastructure, you must create the underlying infrastructure.

Prerequisites

- Review the [OpenShift Container Platform 4.x Tested Integrations](#) page before you create the supporting infrastructure for your cluster.

Procedure

1. Configure DHCP.
2. Provision the required load balancers.
3. Configure the ports for your machines.
4. Configure DNS.
5. Ensure network connectivity.

3.1.3.1. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the Machine Config Server.

During the initial boot, the machines require a DHCP server in order to establish a network connection to download their Ignition config files.

It is recommended to use the DHCP server to manage the machines for the cluster long-term. Ensure that the DHCP server is configured to provide persistent IP addresses and host names to the cluster machines.

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 acceptable approach is to always refer to hosts by their fully-qualified domain names in both the node objects and all DNS requests.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 3.1. All machines to all machines

2379-2380	etcd server, peer, and metrics ports
6443	Kubernetes API
9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .

2379-2380 etcd server, peer, and metrics ports	
10249-10259	The default ports that Kubernetes reserves
10256	openshift-sdn
30000-32767	Kubernetes NodePort

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.



IMPORTANT

OpenShift Container Platform requires all nodes to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Load balancers

Before you install OpenShift Container Platform, you must provision two layer-4 load balancers.

Port	Machines	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.	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
443	The machines that run the Ingress router pods, compute, or worker, by default.	x	x	HTTPS traffic
80	The machines that run the Ingress router pods, compute, or worker by default.	x	x	HTTP traffic




NOTE

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

3.1.3.2. User-provisioned DNS requirements

The following DNS records are required for a OpenShift Container Platform cluster that uses user-provisioned infrastructure. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the cluster base domain that you specify in the **install-config.yaml** file.

Table 3.2. Required DNS records

Component	Record	Description
Kubernetes API	api.<cluster_name>.<base_domain>	This DNS record must point to the load balancer for the control plane machines. This record must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
	api-int.<cluster_name>.<base_domain>	<p>This DNS record must point to the load balancer for the control plane machines. This record must be resolvable from all the nodes within the cluster.</p> <div>  <p>IMPORTANT</p> <p>The API server must be able to resolve the worker nodes by the host names that are recorded in Kubernetes. If it cannot resolve the node names, proxied API calls can fail, and you cannot retrieve logs from Pods.</p> </div>
Routes	*.apps.<cluster_name>.<base_domain>	A wildcard DNS record that points to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. This record must be resolvable by both clients external to the cluster and from all the nodes within the cluster.

Component	Record	Description
etcd	etcd-<code><index></code>.<code><cluster_name></code>.<code><base_domain></code>	OpenShift Container Platform requires DNS records for each etcd instance to point to the control plane machines that host the instances. The etcd instances are differentiated by <code><index></code> values, which start with 0 and end with <code>n-1</code> , where n is the number of control plane machines in the cluster. The DNS record must resolve to an unicast IPV4 address for the control plane machine, and the records must be resolvable from all the nodes in the cluster.
	_etcd-server-ssl._tcp.<code><cluster_name></code>.<code><base_domain></code>	For each control plane machine, OpenShift Container Platform also requires a SRV DNS record for etcd server on that machine with priority 0 , weight 10 and port 2380 . A cluster that uses three control plane machines requires the following records: <pre> # _service._proto.name. TTL class SRV priority weight port target. _etcd-server-ssl._tcp. <cluster_name>. <base_domain> 86400 IN SRV 0 10 2380 etcd- 0.<cluster_name>. <base_domain>. _etcd-server-ssl._tcp. <cluster_name>. <base_domain> 86400 IN SRV 0 10 2380 etcd- 1.<cluster_name>. <base_domain>. _etcd-server-ssl._tcp. <cluster_name>. <base_domain> 86400 IN SRV 0 10 2380 etcd- 2.<cluster_name>. <base_domain>. </pre>

```

# _service._proto.name.                TTL  class SRV priority weight port target.
_etcd-server-ssl._tcp.<cluster_name>.<base_domain> 86400 IN  SRV 0    10  2380 etcd-0.
<cluster_name>.<base_domain>.

```



```
_etcd-server-ssl._tcp.<cluster_name>.<base_domain> 86400 IN SRV 0 10 2380 etcd-1.
<cluster_name>.<base_domain>.
_etcd-server-ssl._tcp.<cluster_name>.<base_domain> 86400 IN SRV 0 10 2380 etcd-2.
<cluster_name>.<base_domain>.
```

3.1.4. Generating an SSH private key and adding it to the agent

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, you must provide an SSH key that your **ssh-agent** process uses to the installer.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t rsa -b 4096 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the SSH key.

Running this command generates an SSH key that does not require a password in the location that you specified.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
Agent pid 31874
```

3. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

When you install OpenShift Container Platform, provide the SSH public key to the installer. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

3.1.5. Obtaining the installation program

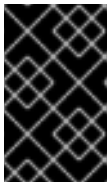
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 300 MB of local disk space to download the installation program.

Procedure

1. Access the [OpenShift Infrastructure Providers](#) page. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Download the installation program for your operating system and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.

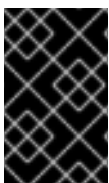
3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [OpenShift Infrastructure Providers](#) page, download your installation pull secret. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

3.1.6. Installing the OpenShift Command-line Interface

You can download and install the OpenShift Command-line Interface (CLI), commonly known as **oc**.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.1. You must download and install the new version of **oc**.

Procedure

1. From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**
2. From the site that is displayed, download the compressed file for your operating system.



NOTE

You can install **oc** on Linux, Windows, or macOS.

3. Extract the compressed file and place it in a directory that is on your PATH.

3.1.7. Manually creating the installation configuration file

For installations of OpenShift Container Platform that use user-provisioned infrastructure, you must manually generate your installation configuration file.

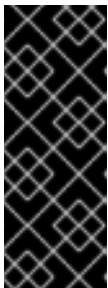
Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.

Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```



IMPORTANT

You must create a directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.



NOTE

You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

3.1.7.1. Sample install-config.yaml file for bare metal

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

```
apiVersion: v1
baseDomain: example.com 1
compute:
- hyperthreading: Enabled 2 3
  name: worker
  replicas: 0 4
```

```

controlPlane:
  hyperthreading: Enabled 5 6
  name: master 7
  replicas: 3 8
metadata:
  name: test 9
networking:
  clusterNetworks:
    - cidr: 10.128.0.0/14 10
      hostPrefix: 23 11
  networkType: OpenShiftSDN
  serviceNetwork: 12
    - 172.30.0.0/16
platform:
  none: {} 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15

```

- 1 The base domain of the cluster. All DNS records must be sub-domains of this base and include the cluster name.
- 2 5 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.
- 3 6 7 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.

- 4 You must set the value of the **replicas** parameter to **0**. This parameter controls the number of workers that the cluster creates and manages for you, which are functions that the cluster does not perform when you use user-provisioned infrastructure. You must manually deploy worker machines for the cluster to use before you finish installing OpenShift Container Platform.
- 8 The number of control plane machines that you add to the cluster. Because the cluster uses this values as the number of etcd endpoints in the cluster, the value must match the number of control plane machines that you deploy.
- 9 The cluster name that you specified in your DNS records.
- 10 A block of IP addresses from which Pod IP addresses are allocated. This block must not overlap with existing physical networks. These IP addresses are used for the Pod network, and if you need to access the Pods from an external network, configure load balancers and routers to manage the traffic.

- 11 The subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23**, then each node is assigned a **/23** subnet out of the given **cidr**, which allows for 510 ($2^{(32 - 23)} - 2$)
- 12 The IP address pool to use for service IP addresses. You can enter only one IP address pool. If you need to access the services from an external network, configure load balancers and routers to manage the traffic.
- 13 You must set the platform to **none**. You cannot provide additional platform configuration variables for bare metal infrastructure.
- 14 The pull secret that you obtained from the [OpenShift Infrastructure Providers](#) page. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.
- 15 The public portion of the default SSH key for the **core** user in Red Hat Enterprise Linux CoreOS (RHCOS).

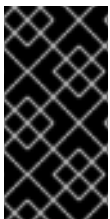


NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, you must provide an SSH key that your **ssh-agent** process uses to the installation program.

3.1.8. Creating the Ignition config files

Because you must manually start the cluster machines, you must generate the Ignition config files that the cluster needs to make its machines.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours. You must complete your cluster installation and keep the cluster running for 24 hours in a non-degraded state to ensure that the first certificate rotation has finished.

Prerequisites

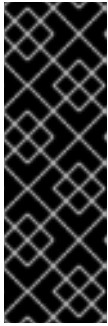
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

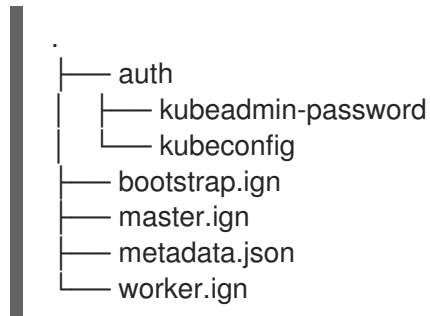
- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

If you created an **install-config.yaml** file, specify the directory that contains it. Otherwise, specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

The following files are generated in the directory:



3.1.9. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines

Before you install a cluster on bare metal infrastructure that you provision, you must create RHCOS machines for it to use. Follow either the steps to use an ISO image or network PXE booting to create the machines.

3.1.9.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image

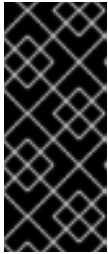
Before you install a cluster on bare metal infrastructure that you provision, you must create RHCOS machines for it to use. You can use an ISO image to create the machines.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Have access to an HTTP server that you can access from your computer and that the machines that you create can access.

Procedure

1. Upload the control plane, compute, and bootstrap Ignition config files that the installation program created to your HTTP server. Note the URLs of these files.
2. Obtain the RHCOS images that are required for your preferred method of installing operating system instances from the [Product Downloads](#) page on the Red Hat customer portal or the [RHCOS image mirror](#) page.



IMPORTANT

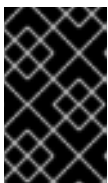
The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available.

You must download the ISO file and either the BIOS or UEFI file. Those file names resemble the following examples:

- ISO: **rhcos-`<version>`-`<architecture>`-installer.iso**
 - Compressed metal BIOS: **rhcos-`<version>`-`<architecture>`-metal-bios.raw.gz**
 - Compressed metal UEFI: **rhcos-`<version>`-`<architecture>`-metal-uefi.raw.gz**
3. Upload either the BIOS or UEFI RHCOS image file to your HTTP server and note its URL.
 4. Use the ISO to start the RHCOS installation. Use one of the following installation options:
 - Burn the ISO image to a disk and boot it directly.
 - Use ISO redirection via a LOM interface.
 5. After the instance boots, press the **TAB** or **E** key to edit the kernel command line.
 6. Add the parameters to the kernel command line:

```
coreos.inst=yes
coreos.inst.install_dev=sda 1
coreos.inst.image_url=<bare_metal_image_URL> 2
coreos.inst.ignition_url=http://example.com/config.ign 3
```

- 1** Specify the block device of the system to install to.
 - 2** Specify the URL of the UEFI or BIOS image that you uploaded to your server.
 - 3** Specify the URL of the Ignition config file for this machine type.
7. Press Enter to complete the installation. After RHCOS installs, the system reboots. After the system reboots, it applies the Ignition config file that you specified.
 8. Continue to create the machines for your cluster.



IMPORTANT

You must create the bootstrap and control plane machines at this time. Because some pods are deployed on compute machines by default, also create at least two compute machines before you install the cluster.

3.1.9.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE or iPXE booting

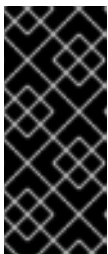
Before you install a cluster on bare metal infrastructure that you provision, you must create RHCOS machines for it to use. You can use PXE or iPXE booting to create the machines.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Configure suitable PXE or iPXE infrastructure.
- Have access to an HTTP server that you can access from your computer.

Procedure

1. Upload the master, worker, and bootstrap Ignition config files that the installation program created to your HTTP server. Note the URLs of these files.
2. Obtain the RHCOS ISO image, compressed metal BIOS, **kernel** and **initramfs** files from the [Product Downloads](#) page on the Red Hat customer portal or the [RHCOS image mirror](#) page.



IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available.

The file names contain the OpenShift Container Platform version number. They resemble the following examples:

- ISO: **rhcos-`<version>`-`<architecture>`-installer.iso**
 - Compressed metal BIOS: **rhcos-`<version>`-`<architecture>`-metal-bios.raw.gz**
 - **kernel**: **rhcos-`<version>`-`<architecture>`-installer-kernel**
 - **initramfs**: **rhcos-`<version>`-`<architecture>`-installer-initramfs.img**
3. Upload the compressed metal BIOS file and the **kernel** and **initramfs** files to your HTTP server.
 4. Configure the network boot infrastructure so that the machines boot from their local disks after RHCOS is installed on them.
 5. Configure PXE or iPXE installation for the RHCOS images.
Modify one of the following example menu entries for your environment and verify that the image and Ignition files are properly accessible:
 - For PXE:

```

DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
  KERNEL http://<HTTP_server>/rhcos-<version>-<architecture>-installer-kernel
  APPEND ip=dhcp rd.neednet=1 initrd=http://<HTTP_server>/rhcos-<version>-

```

1


```
<architecture>-installer-initramfs.img console=tty0 console=ttyS0 coreos.inst=yes
coreos.inst.install_dev=sda coreos.inst.image_url=http://<HTTP_server>/rhcos-
<version>-<architecture>-metal-bios.raw.gz
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 2 3
```

- 1 Specify the location of the **kernel** file that you uploaded to your HTTP server.
- 2 If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.
- 3 Specify locations of the RHCOS files that you uploaded to your HTTP server. The **initrd** parameter value is the location of the **initramfs** file, the **coreos.inst.image_url** parameter value is the location of the compressed metal BIOS file, and the **coreos.inst.ignition_url** parameter value is the location of the bootstrap Ignition config file.

- For iPXE:

```
kernel http://<HTTP_server>/rhcos-<version>-<architecture>-installer-kernel ip=dhcp
rd.neednet=1 initrd=http://<HTTP_server>/rhcos-<version>-<architecture>-installer-
initramfs.img console=tty0 console=ttyS0 coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal-
bios.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1 2
initrd http://<HTTP_server>/rhcos-<version>-<architecture>-installer-initramfs.img 3
boot
```

- 1 Specify locations of the RHCOS files that you uploaded to your HTTP server. The **kernel** parameter value is the location of the **kernel** file, the **initrd** parameter value is the location of the **initramfs** file, the **coreos.inst.image_url** parameter value is the location of the compressed metal BIOS file, and the **coreos.inst.ignition_url** parameter value is the location of the bootstrap Ignition config file.
- 2 If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.
- 3 Specify the location of the **initramfs** file that you uploaded to your HTTP server.

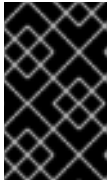
6. If you use UEFI, edit the included **grub.conf** file that is included in the ISO that you downloaded to include the following installation options:

```
menuentry 'Install Red Hat Enterprise Linux CoreOS' --class fedora --class gnu-linux --class
gnu --class os {
    linux /images/vmlinuz nomodeset rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal-
bios.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1
    initrd http://<HTTP_server>/rhcos-<version>-<architecture>-installer-initramfs.img 2
}
```

- 1 For the **coreos.inst.image_url** parameter value, specify the location of the compressed metal UEFI file that you uploaded to your HTTP server. For the **coreos.inst.ignition_url**, specify the location of the bootstrap Ignition config file that you uploaded to your HTTP server.

- 2 Specify the location of the **initramfs** file that you uploaded to your HTTP server.

7. Continue to create the machines for your cluster.



IMPORTANT

You must create the bootstrap and control plane machines at this time. Because some pods are deployed on compute machines by default, also create at least two compute machine before you install the cluster.

3.1.10. Creating the cluster

To create the OpenShift Container Platform cluster, you provision machines by using the Ignition config files that you generated with the installation program.

Prerequisites

- Create the required infrastructure for the cluster.
- You obtained the installation program and generated the Ignition config files for your cluster.
- You used the Ignition config files to create RHCOS machines for your cluster.
- Your machines have direct internet access.

Procedure

1. Start and monitor the installation process:

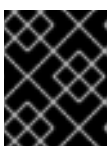
```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
--log-level debug 2
```

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.13.4+b626c2fe1 up
INFO Waiting up to 30m0s for the bootstrap-complete event...
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2 Optionally, include the **--log-level debug** option to view installation details.

The command succeeds when the Kubernetes API server signals that it has been bootstrapped on the control plane machines.

2. After bootstrap process is complete, remove the bootstrap machine from the load balancer.



IMPORTANT

You must remove the bootstrap machine from the load balancer at this point. You can also remove or reformat the machine itself.

3.1.11. Logging into the cluster

You can log into your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
$ oc whoami
system:admin
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

3.1.12. Approving the CSRs for your machines

When you add machines to a cluster, two pending certificates signing request (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes

NAME      STATUS    ROLES    AGE   VERSION
master-0  Ready     master   63m   v1.13.4+b626c2fe1
master-1  Ready     master   63m   v1.13.4+b626c2fe1
master-2  Ready     master   64m   v1.13.4+b626c2fe1
worker-0  NotReady  worker   76s   v1.13.4+b626c2fe1
worker-1  NotReady  worker   70s   v1.13.4+b626c2fe1
```

The output lists all of the machines that you created.

2. Review the pending certificate signing requests (CSRs) and ensure that the you see a client and server request with **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

NAME	AGE	REQUESTOR	CONDITION
csr-8b2br	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending 1
csr-8vnps	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
csr-bfd72	5m26s	system:node:ip-10-0-50-126.us-east-2.compute.internal	Pending 2
csr-c57lv	5m26s	system:node:ip-10-0-95-157.us-east-2.compute.internal	Pending
...			

1 A client request CSR.

2 A server request CSR.

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After you approve the initial CSRs, the subsequent node client CSRs are automatically approved by the cluster **kube-controller-manager**. You must implement a method of automatically approving the kubelet serving certificate requests.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** is the name of a CSR from the list of current CSRs.

- If all the CSRs are valid, approve them all by running the following command:

```
$ oc get csr -ojson | jq -r '.items[] | select(.status == {} ) | .metadata.name' | xargs oc adm certificate approve
```

3.1.13. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED
SINCE				
authentication	4.1.0	True	False	False 69s
cloud-credential	4.1.0	True	False	False 12m
cluster-autoscaler	4.1.0	True	False	False 11m
console	4.1.0	True	False	False 46s
dns	4.1.0	True	False	False 11m
image-registry	4.1.0	False	True	False 5m26s
ingress	4.1.0	True	False	False 5m36s
kube-apiserver	4.1.0	True	False	False 8m53s
kube-controller-manager	4.1.0	True	False	False 7m24s
kube-scheduler	4.1.0	True	False	False 12m
machine-api	4.1.0	True	False	False 12m
machine-config	4.1.0	True	False	False 7m36s
marketplace	4.1.0	True	False	False 7m54m
monitoring	4.1.0	True	False	False 7h54s
network	4.1.0	True	False	False 5m9s
node-tuning	4.1.0	True	False	False 11m
openshift-apiserver	4.1.0	True	False	False 11m
openshift-controller-manager	4.1.0	True	False	False 5m943s
openshift-samples	4.1.0	True	False	False 3m55s
operator-lifecycle-manager	4.1.0	True	False	False 11m
operator-lifecycle-manager-catalog	4.1.0	True	False	False 11m
service-ca	4.1.0	True	False	False 11m
service-catalog-apiserver	4.1.0	True	False	False 5m26s
service-catalog-controller-manager	4.1.0	True	False	False 5m25s
storage	4.1.0	True	False	False 5m30s

2. Configure the Operators that are not available.

3.1.13.1. Image registry storage configuration

If the **image-registry** Operator is not available, you must configure storage for it. Instructions for both configuring a PersistentVolume, which is required for production clusters, and for configuring an empty directory as the storage location, which is available for only non-production clusters, are shown.

3.1.13.1.1. Configuring registry storage for bare metal

As a cluster administrator, following installation you must configure your registry to use storage.

Prerequisites

- Cluster administrator permissions.
- A cluster on bare metal.
- A provisioned persistent volume (PV) with **ReadWriteMany** access mode, such as **NFS**.
- Must have "100Gi" capacity.

Procedure

1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.
2. Verify you do not have a registry pod:

```
$ oc get pod -n openshift-image-registry
```

NOTE

If the storage type is **emptyDIR**, the replica number can not be greater than **1**. If the storage type is **NFS**, and you want to scale up the registry Pod by setting **replica>1** you must enable the **no_wdelay** mount option. For example:

```
# cat /etc/exports
/mnt/data *(rw,sync,no_wdelay,no_root_squash,insecure,fsid=0)
sh-4.3# exportfs -rv
exporting */mnt/data
```

1. Check the registry configuration:

```
$ oc edit configs.imageregistry.operator.openshift.io

storage:
  pvc:
    claim:
```

Leave the **claim** field blank to allow the automatic creation of an **image-registry-storage** PVC.

2. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

3.1.13.1.2. Configuring storage for the image registry in non-production clusters

You must configure storage for the image registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

Procedure

- To set the image registry storage to an empty directory:

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec": {"storage":{"emptyDir":{}}}}'
```

**WARNING**

Configure this option for only non-production clusters.

If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

Wait a few minutes and run the command again.

3.1.14. Completing installation on user-provisioned infrastructure

After you complete the Operator configuration, you can finish installing the cluster on infrastructure that you provide.

Prerequisites

- Your control plane has initialized.
- You have completed the initial Operator configuration.

Procedure

1. Confirm that all the cluster components are online:

```
$ watch -n5 oc get clusteroperators
```

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED
SINCE				
authentication	4.1.0	True	False	False 10m
cloud-credential	4.1.0	True	False	False 22m
cluster-autoscaler	4.1.0	True	False	False 21m
console	4.1.0	True	False	False 10m
dns	4.1.0	True	False	False 21m
image-registry	4.1.0	True	False	False 16m
ingress	4.1.0	True	False	False 16m
kube-apiserver	4.1.0	True	False	False 19m
kube-controller-manager	4.1.0	True	False	False 18m
kube-scheduler	4.1.0	True	False	False 22m
machine-api	4.1.0	True	False	False 22m
machine-config	4.1.0	True	False	False 18m
marketplace	4.1.0	True	False	False 18m
monitoring	4.1.0	True	False	False 18m
network	4.1.0	True	False	False 16m
node-tuning	4.1.0	True	False	False 21m
openshift-apiserver	4.1.0	True	False	False 21m
openshift-controller-manager	4.1.0	True	False	False 17m
openshift-samples	4.1.0	True	False	False 14m
operator-lifecycle-manager	4.1.0	True	False	False 21m

operator-lifecycle-manager-catalog	4.1.0	True	False	False	21m
service-ca	4.1.0	True	False	False	21m
service-catalog-apiserver	4.1.0	True	False	False	16m
service-catalog-controller-manager	4.1.0	True	False	False	16m
storage	4.1.0	True	False	False	16m

When all of the cluster Operators are **AVAILABLE**, you can complete the installation.

2. Monitor for cluster completion:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete ❶
INFO Waiting up to 30m0s for the cluster to initialize...
```

- ❶ For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

The command succeeds when the Cluster Version Operator finishes deploying the OpenShift Container Platform cluster from Kubernetes API server.

3. Confirm that the Kubernetes API server is communicating with the Pods.

- a. To view a list of all Pods, use the following command:

```
$ oc get pods --all-namespaces
```

NAMESPACE	NAME	READY	STATUS
RESTARTS AGE			
openshift-apiserver-operator	openshift-apiserver-operator-85cb746d55-zqhs8	1/1	Running
1 9m			
openshift-apiserver	apiserver-67b9g	1/1	Running
3m			
openshift-apiserver	apiserver-ljcmx	1/1	Running
1m			
openshift-apiserver	apiserver-z25h4	1/1	Running
2m			
openshift-authentication-operator	authentication-operator-69d5d8bf84-vh2n8	1/1	Running
0 5m			
...			

- b. View the logs for a Pod that is listed in the output of the previous command by using the following command:

```
$ oc logs <pod_name> -n <namespace> ❶
```

- ❶ Specify the Pod name and namespace, as shown in the output of the previous command.

If the Pod logs display, the Kubernetes API server can communicate with the cluster machines.

Next steps

- [Customize your cluster.](#)

- If necessary, you can [opt out of telemetry](#).

CHAPTER 4. INSTALLING ON VSPHERE

4.1. INSTALLING A CLUSTER ON VSPHERE

In OpenShift Container Platform version 4.1, you can install a cluster on VMware vSphere infrastructure that you provision.

Prerequisites

- Provision [persistent storage](#) for your cluster. To deploy a private image registry, your storage must provide ReadWriteMany access modes.
- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall, you must [configure it to access Red Hat Insights](#) .

4.1.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.1, Telemetry is the component that provides metrics about cluster health and the success of updates. To perform subscription management, including legally entitling your purchase from Red Hat, you must use the Telemetry service and access the [OpenShift Infrastructure Providers](#) page.

Because there is no disconnected subscription management, you cannot both opt out of sending data back to Red Hat and entitle your purchase. Support for disconnected subscription management might be added in future releases of OpenShift Container Platform



IMPORTANT

Your machines must have direct internet access to install the cluster.

You must have internet access to:

- Access the [OpenShift Infrastructure Providers](#) page to download the installation program
- Access [quay.io](#) to obtain the packages that are required to install your cluster
- Obtain the packages that are required to perform cluster updates
- Access [Red Hat's software as a service page](#) to perform subscription management

4.1.2. VMware vSphere infrastructure requirements

You must install the OpenShift Container Platform cluster on a VMware vSphere version 6.5 or 6.7U2 or later instance.

VMware recommends using vSphere Version 6.7 U2 or later with your OpenShift Container Platform cluster. vSphere 6.7U2 includes:

- Support for VMware NSX-T
- Support for vSAN, VMFS and NFS, using the in-tree VCP

While vSphere 6.5 with Hardware version 11 is supported, OpenShift Container Platform clusters are subject to the following restrictions:

- NSX-T SDN is not supported.
- You must use another SDN or storage provider that OpenShift Container Platform supports.

If you use a vSphere version 6.5 instance, consider upgrading to 6.7U2 before you install OpenShift Container Platform.

4.1.3. Machine requirements for a cluster with user-provisioned infrastructure

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

4.1.3.1. Required machines

The smallest OpenShift Container Platform clusters require the following hosts:

- One bootstrap machine
- Three control plane, or master, machines
- At least two compute, or worker, machines



NOTE

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.



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.

4.1.3.2. Network connectivity requirements

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config files from the Machine Config Server. During the initial boot, the machines require a DHCP server in order to establish a network connection to download their Ignition config files. After the initial boot, the machines can be configured to use static IP addresses.

4.1.3.3. Minimum resource requirements

Each cluster machine must meet the following minimum requirements:

Machine	Operating System	vCPU	RAM	Storage
Bootstrap	RHCOS	4	16 GB	120 GB

Machine	Operating System	vCPU	RAM	Storage
Control plane	RHCOS	4	16 GB	120 GB
Compute	RHCOS or RHEL 7.6	2	8 GB	120 GB

4.1.3.4. 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.

4.1.4. Creating the user-provisioned infrastructure

Before you deploy a OpenShift Container Platform cluster that uses user-provisioned infrastructure, you must create the underlying infrastructure.

Prerequisites

- Review the [OpenShift Container Platform 4.x Tested Integrations](#) page before you create the supporting infrastructure for your cluster.

Procedure

1. Configure DHCP.
2. Provision the required load balancers.
3. Configure the ports for your machines.
4. Configure DNS.
5. Ensure network connectivity.

4.1.4.1. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the Machine Config Server.

During the initial boot, the machines require a DHCP server in order to establish a network connection to download their Ignition config files.

It is recommended to use the DHCP server to manage the machines for the cluster long-term. Ensure that the DHCP server is configured to provide persistent IP addresses and host names to the cluster machines.

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 acceptable approach is to always refer to hosts by their fully-qualified domain names in both the node objects and all DNS requests.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 4.1. All machines to all machines

2379-2380 etcd server, peer, and metrics ports	
6443	Kubernetes API
9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .
10249-10259	The default ports that Kubernetes reserves
10256	openshift-sdn
30000-32767	Kubernetes NodePort

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.



IMPORTANT

OpenShift Container Platform requires all nodes to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Load balancers

Before you install OpenShift Container Platform, you must provision two layer-4 load balancers.

Port	Machines	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.	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
443	The machines that run the Ingress router pods, compute, or worker, by default.	x	x	HTTPS traffic

Port	Machines	Internal	External	Description
80	The machines that run the Ingress router pods, compute, or worker by default.	x	x	HTTP traffic




NOTE

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

4.1.4.2. User-provisioned DNS requirements

The following DNS records are required for a OpenShift Container Platform cluster that uses user-provisioned infrastructure. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the cluster base domain that you specify in the **install-config.yaml** file.

Table 4.2. Required DNS records

Component	Record	Description
Kubernetes API	api.<cluster_name>.<base_domain>	This DNS record must point to the load balancer for the control plane machines. This record must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
	api-int.<cluster_name>.<base_domain>	<div>This DNS record must point to the load balancer for the control plane machines. This record must be resolvable from all the nodes within the cluster.</div> <div>IMPORTANT The API server must be able to resolve the worker nodes by the host names that are recorded in Kubernetes. If it cannot resolve the node names, proxied API calls can fail, and you cannot retrieve logs from Pods.</div>

Component	Record	Description
Routes	*.apps.<cluster_name>.<base_domain>	A wildcard DNS record that points to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. This record must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
etcd	etcd-<index>.<cluster_name>.<base_domain>	OpenShift Container Platform requires DNS records for each etcd instance to point to the control plane machines that host the instances. The etcd instances are differentiated by <index> values, which start with 0 and end with n-1 , where n is the number of control plane machines in the cluster. The DNS record must resolve to an unicast IPV4 address for the control plane machine, and the records must be resolvable from all the nodes in the cluster.

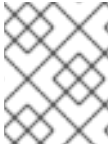
Component	Record	Description
	<code>_etcd-server-ssl._tcp.<cluster_name>.<base_domain></code>	<p>For each control plane machine, OpenShift Container Platform also requires a SRV DNS record for etcd server on that machine with priority 0, weight 10 and port 2380. A cluster that uses three control plane machines requires the following records:</p> <pre># _service._proto.name. TTL class SRV priority weight port target. _etcd-server-ssl._tcp. <cluster_name>. <base_domain> 86400 IN SRV 0 10 2380 etcd- 0.<cluster_name>. <base_domain>. _etcd-server-ssl._tcp. <cluster_name>. <base_domain> 86400 IN SRV 0 10 2380 etcd- 1.<cluster_name>. <base_domain>. _etcd-server-ssl._tcp. <cluster_name>. <base_domain> 86400 IN SRV 0 10 2380 etcd- 2.<cluster_name>. <base_domain>.</pre>

```
# _service._proto.name.          TTL  class SRV priority weight port target.
_etcd-server-ssl._tcp.<cluster_name>.<base_domain> 86400 IN  SRV 0    10  2380 etcd-0.
<cluster_name>.<base_domain>.
_etcd-server-ssl._tcp.<cluster_name>.<base_domain> 86400 IN  SRV 0    10  2380 etcd-1.
<cluster_name>.<base_domain>.
_etcd-server-ssl._tcp.<cluster_name>.<base_domain> 86400 IN  SRV 0    10  2380 etcd-2.
<cluster_name>.<base_domain>.
```

4.1.5. Generating an SSH private key and adding it to the agent

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, you must provide an SSH key that your **ssh-agent** process uses to the installer.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t rsa -b 4096 -N "" \
  -f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the SSH key.

Running this command generates an SSH key that does not require a password in the location that you specified.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"

Agent pid 31874
```

3. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1

Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

When you install OpenShift Container Platform, provide the SSH public key to the installer. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

4.1.6. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

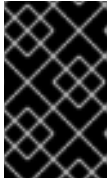
Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 300 MB of local disk space to download the installation program.

Procedure

1. Access the [OpenShift Infrastructure Providers](#) page. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

2. Download the installation program for your operating system and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [OpenShift Infrastructure Providers](#) page, download your installation pull secret. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

4.1.7. Manually creating the installation configuration file

For installations of OpenShift Container Platform that use user-provisioned infrastructure, you must manually generate your installation configuration file.

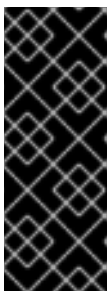
Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.

Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```



IMPORTANT

You must create a directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

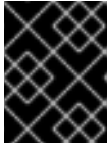
2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.



NOTE

You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

4.1.7.1. Sample install-config.yaml file for VMware vSphere

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

```
apiVersion: v1
baseDomain: example.com 1
compute:
- hyperthreading: Enabled 2 3
  name: worker
  replicas: 0 4
controlPlane:
  hyperthreading: Enabled 5 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
platform:
  vsphere:
    vcenter: your.vcenter.server 9
    username: username 10
    password: password 11
    datacenter: datacenter 12
    defaultDatastore: datastore 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
```

- 1 The base domain of the cluster. All DNS records must be sub-domains of this base and include the cluster name.
- 2 5 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.
- 3 6 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Your machines must use at least 8 CPUs and 32 GB of RAM if you disable simultaneous multithreading.

- 4 You must set the value of the **replicas** parameter to **0**. This parameter controls the number of workers that the cluster creates and manages for you, which are functions that the cluster does not
- 7 The number of control plane machines that you add to the cluster. Because the cluster uses this values as the number of etcd endpoints in the cluster, the value must match the number of control plane machines that you deploy.
- 8 The cluster name that you specified in your DNS records.
- 9 The fully-qualified host name or IP address of the vCenter server.
- 10 The name of the user for accessing the server. This user must have at least the roles and privileges that are required for [dynamic persistent volume provisioning](#) in vSphere.
- 11 The password associated with the vSphere user.
- 12 The vSphere datacenter.
- 13 The default vSphere datastore to use.
- 14 The pull secret that you obtained from the [OpenShift Infrastructure Providers](#) page. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.
- 15 The public portion of the default SSH key for the **core** user in Red Hat Enterprise Linux CoreOS (RHCOS).



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, you must provide an SSH key that your **ssh-agent** process uses to the installer.

4.1.8. Creating the Ignition config files

Because you must manually start the cluster machines, you must generate the Ignition config files that the cluster needs to make its machines.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours. You must complete your cluster installation and keep the cluster running for 24 hours in a non-degraded state to ensure that the first certificate rotation has finished.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

If you created an **install-config.yaml** file, specify the directory that contains it. Otherwise, specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

The following files are generated in the directory:

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

4.1.9. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines in vSphere

Before you install a cluster that contains user-provisioned infrastructure on VMware vSphere, you must create RHCOS machines on vSphere hosts for it to use.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Have access to an HTTP server that you can access from your computer and that the machines that you create can access.
- Create a [vSphere cluster](#).

Procedure

1. Upload the bootstrap Ignition config file, which is named **<installation_directory>/bootstrap.ign**, that the installation program created to your HTTP server. Note the URL of this file.
You must host the bootstrap Ignition config file because it is too large to fit in a vApp property.
2. Save the following secondary Ignition config file for your bootstrap node to your computer as **<installation_directory>/append-bootstrap.ign**.

```
{
  "ignition": {
```

```

"config": {
  "append": [
    {
      "source": "<bootstrap_ignition_config_url>", 1
      "verification": {}
    }
  ],
},
"timeouts": {},
"version": "2.1.0"
},
"networkd": {},
"passwd": {},
"storage": {},
"systemd": {}
}

```

- 1 Specify the URL of the bootstrap Ignition config file that you hosted.

When you create the Virtual Machine (VM) for the bootstrap machine, you use this Ignition config file.

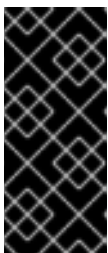
3. Convert the master, worker, and secondary bootstrap Ignition config files to Base64 encoding. For example, if you use a Linux operating system, you can use the **base64** command to encode the files.

```

$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
$ base64 -w0 <installation_directory>/append-bootstrap.ign >
<installation_directory>/append-bootstrap.64

```

4. Obtain the RHCOS OVA image from the [Product Downloads](#) page on the Red Hat customer portal or the [RHCOS image mirror](#) page.



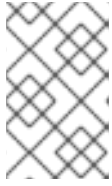
IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download an image with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image version that matches your OpenShift Container Platform version if it is available.

The file name contains the OpenShift Container Platform version number in the format **rhcos-
<version>-<architecture>-vmware.ova**.

5. In the vSphere Client, create a folder in your datacenter to store your VMs.
 - a. Click the **VMs and Templates** view.
 - b. Right-click the name of your datacenter.
 - c. Click **New Folder → New VM and Template Folder**.

- d. In the window that is displayed, enter the folder name. The folder name must match the cluster name that you specified in the **install-config.yaml** file.
6. In the vSphere Client, create a template for the OVA image.



NOTE

In the following steps, you use the same template for all of your cluster machines and provide the location for the Ignition config file for that machine type when you provision the VMs.

- a. From the **Hosts and Clusters** tab, right-click your cluster's name and click **Deploy OVF Template**.
- b. On the **Select an OVF** tab, specify the name of the RHCOS OVA file that you downloaded.
- c. On the **Select a name and folder** tab, set a **Virtual machine name**, such as RHCOS, click the name of your vSphere cluster, and select the folder you created in the previous step.
- d. On the **Select a compute resource** tab, click the name of your vSphere cluster.
- e. On the **Select storage** tab, configure the storage options for your VM.
 - Select **Thin Provision**.
 - Select the datastore that you specified in your **install-config.yaml** file.
- f. On the **Select network** tab, specify the network that you configured for the cluster, if available.
- g. If you plan to use the same template for all cluster machine types, do not specify values on the **Customize template** tab.
7. After the template deploys, deploy a VM for a machine in the cluster.
 - a. Right-click the template's name and click **Clone → Clone to Virtual Machine**
 - b. On the **Select a name and folder** tab, specify a name for the VM. You might include the machine type in the name, such as **control-plane-0** or **compute-1**.
 - c. On the **Select a name and folder** tab, select the name of the folder that you created for the cluster.
 - d. On the **Select a compute resource** tab, select the name of a host in your datacenter.
 - e. Optional: On the **Select storage** tab, customize the storage options.
 - f. On the **Select clone options**, select **Customize this virtual machine's hardware**
 - g. On the **Customize hardware** tab, click **VM Options → Advanced**.
 - From the **Latency Sensitivity** list, select **High**.
 - Click **Edit Configuration**, and on the **Configuration Parameters** window, click **Add Configuration Params**. Define the following parameter names and values:

- **guestinfo.ignition.config.data**: Paste the contents of the base64-encoded Ignition config file for this machine type.
 - **guestinfo.ignition.config.data.encoding**: Specify **base64**.
 - **disk.EnableUUID**: Specify **TRUE**.
- h. In the **Virtual Hardware** panel of the **Customize hardware** tab, modify the specified values as required. Ensure that the amount of RAM, CPU, and disk storage meets the minimum requirements for the machine type.
 - i. Complete the configuration and power on the VM.
8. Create the rest of the machines for your cluster by following the preceding steps for each machine.

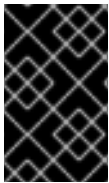


IMPORTANT

You must create the bootstrap and control plane machines at this time. Because some pods are deployed on compute machines by default, also create at least two compute machine before you install the cluster.

4.1.10. Installing the OpenShift Command-line Interface

You can download and install the OpenShift Command-line Interface (CLI), commonly known as **oc**.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.1. You must download and install the new version of **oc**.

Procedure

1. From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**
2. From the site that is displayed, download the compressed file for your operating system.



NOTE

You can install **oc** on Linux, Windows, or macOS.

3. Extract the compressed file and place it in a directory that is on your PATH.

4.1.11. Creating the cluster

To create the OpenShift Container Platform cluster, you provision machines by using the Ignition config files that you generated with the installation program.

Prerequisites

- Create the required infrastructure for the cluster.
- You obtained the installation program and generated the Ignition config files for your cluster.

- You used the Ignition config files to create RHCOS machines for your cluster.
- Your machines have direct internet access.

Procedure

1. Start and monitor the installation process:

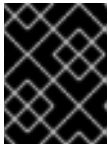
```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
--log-level debug 2

INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.13.4+b626c2fe1 up
INFO Waiting up to 30m0s for the bootstrap-complete event...
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2 Optionally, include the **--log-level debug** option to view installation details.

The command succeeds when the Kubernetes API server signals that it has been bootstrapped on the control plane machines.

2. After bootstrap process is complete, remove the bootstrap machine from the load balancer.



IMPORTANT

You must remove the bootstrap machine from the load balancer at this point. You can also remove or reformat the machine itself.

4.1.12. Logging into the cluster

You can log into your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
$ oc whoami
system:admin
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

4.1.13. Approving the CSRs for your machines

When you add machines to a cluster, two pending certificates signing request (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	63m	v1.13.4+b626c2fe1
master-1	Ready	master	63m	v1.13.4+b626c2fe1
master-2	Ready	master	64m	v1.13.4+b626c2fe1
worker-0	NotReady	worker	76s	v1.13.4+b626c2fe1
worker-1	NotReady	worker	70s	v1.13.4+b626c2fe1

The output lists all of the machines that you created.

2. Review the pending certificate signing requests (CSRs) and ensure that the you see a client and server request with **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

NAME	AGE	REQUESTOR	CONDITION
csr-8b2br	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrap	Pending 1
csr-8vnps	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrap	Pending
csr-bfd72	5m26s	system:node:ip-10-0-50-126.us-east-2.compute.internal	Pending 2
csr-c57lv	5m26s	system:node:ip-10-0-95-157.us-east-2.compute.internal	Pending
...			

- 1 A client request CSR.

- 2 A server request CSR.

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After you approve the initial CSRs, the subsequent node client CSRs are automatically approved by the cluster **kube-controller-manager**. You must implement a method of automatically approving the kubelet serving certificate requests.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- If all the CSRs are valid, approve them all by running the following command:

```
$ oc get csr -ojson | jq -r '.items[] | select(.status == {} ) | .metadata.name' | xargs oc adm certificate approve
```

4.1.14. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

- Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED
SINCE				
authentication	4.1.0	True	False	False 69s
cloud-credential	4.1.0	True	False	False 12m
cluster-autoscaler	4.1.0	True	False	False 11m
console	4.1.0	True	False	False 46s
dns	4.1.0	True	False	False 11m
image-registry	4.1.0	False	True	False 5m26s
ingress	4.1.0	True	False	False 5m36s
kube-apiserver	4.1.0	True	False	False 8m53s
kube-controller-manager	4.1.0	True	False	False 7m24s
kube-scheduler	4.1.0	True	False	False 12m
machine-api	4.1.0	True	False	False 12m
machine-config	4.1.0	True	False	False 7m36s

marketplace	4.1.0	True	False	False	7m54m
monitoring	4.1.0	True	False	False	7h54s
network	4.1.0	True	False	False	5m9s
node-tuning	4.1.0	True	False	False	11m
openshift-apiserver	4.1.0	True	False	False	11m
openshift-controller-manager	4.1.0	True	False	False	5m943s
openshift-samples	4.1.0	True	False	False	3m55s
operator-lifecycle-manager	4.1.0	True	False	False	11m
operator-lifecycle-manager-catalog	4.1.0	True	False	False	11m
service-ca	4.1.0	True	False	False	11m
service-catalog-apiserver	4.1.0	True	False	False	5m26s
service-catalog-controller-manager	4.1.0	True	False	False	5m25s
storage	4.1.0	True	False	False	5m30s

2. Configure the Operators that are not available.

4.1.14.1. Image registry storage configuration

If the **image-registry** Operator is not available, you must configure storage for it. Instructions for both configuring a PersistentVolume, which is required for production clusters, and for configuring an empty directory as the storage location, which is available for only non-production clusters, are shown.

4.1.14.1.1. Configuring registry storage for VMware vSphere

As a cluster administrator, following installation you must configure your registry to use storage.

Prerequisites

- Cluster administrator permissions.
- A cluster on VMware vSphere.
- A provisioned persistent volume (PV) with **ReadWriteMany** access mode, such as **NFS**.
- Must have "100Gi" capacity.

Procedure

1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.
2. Verify you do not have a registry pod:

```
$ oc get pod -n openshift-image-registry
```



NOTE

If the storage type is **emptyDir**, the replica number can not be greater than **1**. If the storage type is **NFS**, and you want to scale up the registry Pod by setting **replica>1** you must enable the **no_wdelay** mount option. For example:

```
# cat /etc/exports
/mnt/data *(rw, sync, no_wdelay, no_root_squash, insecure, fsid=0)
sh-4.3# exportfs -rv
exporting */mnt/data
```

1. Check the registry configuration:

```
$ oc edit configs.imageregistry.operator.openshift.io

storage:
pvc:
claim:
```

Leave the **claim** field blank to allow the automatic creation of an **image-registry-storage** PVC.

2. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

4.1.14.1.2. Configuring storage for the image registry in non-production clusters

You must configure storage for the image registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

Procedure

- To set the image registry storage to an empty directory:

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}'
```



WARNING

Configure this option for only non-production clusters.

If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

Wait a few minutes and run the command again.

4.1.15. Completing installation on user-provisioned infrastructure

After you complete the Operator configuration, you can finish installing the cluster on infrastructure that you provide.

Prerequisites

- Your control plane has initialized.
- You have completed the initial Operator configuration.

Procedure

1. Confirm that all the cluster components are online:

```
$ watch -n5 oc get clusteroperators
```

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED
SINCE				
authentication	4.1.0	True	False	False 10m
cloud-credential	4.1.0	True	False	False 22m
cluster-autoscaler	4.1.0	True	False	False 21m
console	4.1.0	True	False	False 10m
dns	4.1.0	True	False	False 21m
image-registry	4.1.0	True	False	False 16m
ingress	4.1.0	True	False	False 16m
kube-apiserver	4.1.0	True	False	False 19m
kube-controller-manager	4.1.0	True	False	False 18m
kube-scheduler	4.1.0	True	False	False 22m
machine-api	4.1.0	True	False	False 22m
machine-config	4.1.0	True	False	False 18m
marketplace	4.1.0	True	False	False 18m
monitoring	4.1.0	True	False	False 18m
network	4.1.0	True	False	False 16m
node-tuning	4.1.0	True	False	False 21m
openshift-apiserver	4.1.0	True	False	False 21m
openshift-controller-manager	4.1.0	True	False	False 17m
openshift-samples	4.1.0	True	False	False 14m
operator-lifecycle-manager	4.1.0	True	False	False 21m
operator-lifecycle-manager-catalog	4.1.0	True	False	False 21m
service-ca	4.1.0	True	False	False 21m
service-catalog-apiserver	4.1.0	True	False	False 16m
service-catalog-controller-manager	4.1.0	True	False	False 16m
storage	4.1.0	True	False	False 16m

When all of the cluster Operators are **AVAILABLE**, you can complete the installation.

2. Monitor for cluster completion:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
INFO Waiting up to 30m0s for the cluster to initialize...
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

The command succeeds when the Cluster Version Operator finishes deploying the OpenShift Container Platform cluster from Kubernetes API server.

3. Confirm that the Kubernetes API server is communicating with the Pods.

- a. To view a list of all Pods, use the following command:

```
$ oc get pods --all-namespaces
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
openshift-apiserver-operator	openshift-apiserver-operator-85cb746d55-zqhs8	1/1	Running	1	9m
openshift-apiserver	apiserver-67b9g	1/1	Running	0	3m
openshift-apiserver	apiserver-ljcmx	1/1	Running	0	1m
openshift-apiserver	apiserver-z25h4	1/1	Running	0	2m
openshift-authentication-operator	authentication-operator-69d5d8bf84-vh2n8	1/1	Running	0	5m
...					

- b. View the logs for a Pod that is listed in the output of the previous command by using the following command:

```
$ oc logs <pod_name> -n <namespace> ❶
```

- ❶ Specify the Pod name and namespace, as shown in the output of the previous command.

If the Pod logs display, the Kubernetes API server can communicate with the cluster machines.

Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of telemetry](#).

CHAPTER 5. GATHERING INSTALLATION LOGS

To assist in troubleshooting a failed OpenShift Container Platform installation, you can gather logs from the bootstrap and control plane, or master, machines.

Prerequisites

- You attempted to install a OpenShift Container Platform cluster, and installation failed.
- You provided an SSH key to the installation program, and that key is in your running **ssh-agent** process.

5.1. GATHERING LOGS FROM A FAILED INSTALLATION

If you gave an SSH key to your installation program, you can gather data about your failed installation.



NOTE

You use a different command to gather logs about an unsuccessful installation than to gather logs from a running cluster. If you must gather logs from a running cluster, use the **oc adm must-gather** command.

Prerequisites

- Your OpenShift Container Platform installation failed before the bootstrap process finished. The bootstrap node must be running and accessible through SSH.
- The **ssh-agent** process is active on your computer, and you provided both the **ssh-agent** process and the installation program the same SSH key.
- If you tried to install a cluster on infrastructure that you provisioned, you must have the fully-qualified domain names of the control plane, or master, machines.

Procedure

1. Generate the commands that are required to obtain the installation logs from the bootstrap and control plane machines:

- If you used installer-provisioned infrastructure, run the following command:

```
$ ./openshift-install gather bootstrap --dir=<directory> 1
```

- 1 **installation_directory** is the directory you stored the OpenShift Container Platform definition files that the installation program creates.

For installer-provisioned infrastructure, the installation program stores information about the cluster, so you do not specify the host names or IP addresses

- If you used infrastructure that you provisioned yourself, run the following command:

```
$ ./openshift-install gather bootstrap --dir=<directory> \ 1
--bootstrap <bootstrap_address> \ 2
--master "<master_address> <master_address> <master_address>" 3
```


- 1 **installation_directory** is the directory you stored the OpenShift Container Platform definition files that the installation program creates.
- 2 **<bootstrap_address>** is the fully-qualified domain name or IP address of the cluster's bootstrap machine.
- 3 **<master_address>** is the fully-qualified domain name or IP address of a control plane, or master, machine in your cluster. Specify a space-delimited list that contains all the control plane machines in your cluster.

The command output resembles the following example:

```
INFO Use the following commands to gather logs from the cluster
INFO ssh -A core@<bootstrap_address> '/usr/local/bin/installer-gather.sh <master_address>
<master_address> <master_address>'
INFO scp core@<bootstrap_address>:~/log-bundle.tar.gz .
```

You use both commands that are displayed to gather and download the logs.

2. Gather logs from the bootstrap and master machines:

```
$ ssh -A core@<bootstrap_address> '/usr/local/bin/installer-gather.sh <master_address>
<master_address> <master_address>'
```

You SSH into the bootstrap machine and run the gather tool, which is designed to collect as much data as possible from the bootstrap and control plane machines in your cluster and compress all of the gathered files.



NOTE

It is normal to see errors in the command output. If the command output displays the instructions to download the compressed log files, **log-bundle.tar.gz**, then the command succeeded.

3. Download the compressed file that contains the logs:

```
$ scp core@<bootstrap_address>:~/log-bundle.tar.gz . 1
```

- 1 **<bootstrap_address>** is the fully-qualified domain name or IP address of the bootstrap machine.

The command to download the log files is included at the end of the gather command output.

If you open a Red Hat support case about your installation failure, include the compressed logs in the case.

CHAPTER 6. INSTALLATION CONFIGURATION

6.1. AVAILABLE CLUSTER CUSTOMIZATIONS

You complete most of the cluster configuration and customization after you deploy your OpenShift Container Platform cluster. A number of *configuration resources* are available.

You modify the configuration resources to configure the major features of the cluster, such as the image registry, networking configuration, image build behavior, and the identity provider.

For current documentation of settings these resources expose, use the **oc explain** command, for example **oc explain builds --api-version=config.openshift.io/v1**

6.1.1. Cluster configuration resources

All cluster configuration resources are globally scoped (not namespaced) and named **cluster**.

Resource name	Description
apiserver.config.openshift.io	Provides api-server configuration such as certificates and certificate authorities .
authentication.config.openshift.io	Controls the identity provider and authentication configuration for the cluster.
build.config.openshift.io	Controls default and enforced configuration for all builds on the cluster.
console.config.openshift.io	Configures the behavior of the web console interface, including the logout behavior .
featuregate.config.openshift.io	Enables FeatureGates so that you can use Tech Preview features.
image.config.openshift.io	Configures how specific image registries should be treated (allowed, disallowed, insecure, CA details).
ingress.config.openshift.io	Configuration details related to routing such as the default domain for routes.
oauth.config.openshift.io	Configures identity providers and other behavior related to internal OAuth server flows.
project.config.openshift.io	Configures how projects are created including the project template.
proxy.config.openshift.io	Defines proxies to be used by components needing external network access. Note: not all components currently consume this value.

Resource name	Description
scheduler.config.openshift.io	Configures scheduler behavior such as policies and default node selectors.

6.1.2. Operator configuration resources

These configuration resources are cluster-scoped instances, named **cluster**, which control the behavior of a specific component as owned by a particular operator.

Resource name	Description
console.operator.openshift.io	Controls console appearance such as branding customizations
config.imageregistry.operator.openshift.io	Configures internal image registry settings such as public routing, log levels, proxy settings, resource constraints, replica counts, and storage type.
config.samples.operator.openshift.io	Configures the Samples Operator to control which example image streams and templates are installed on the cluster.

6.1.3. Additional configuration resources

These configuration resources represent a single instance of a particular component, in some cases multiple instances can be requested by creating multiple instances of the resource. In other cases only a specific resource instance name in a specific namespace will be consumed by the operator. Reference the component-specific documentation for details on how and when additional resource instances can be created.

Resource name	Instance name	Namespace	Description
alertmanager.monitoring.coreos.com	main	openshift-monitoring	Controls the alertmanager deployment parameters.
ingresscontroller.operator.openshift.io	default	openshift-ingress-operator	Configures Ingress Operator behavior such as domain, number of replicas, certificates, and controller placement.

6.1.4. Informational Resources

You use these resources to retrieve information about the cluster. You should not edit these resources directly.

Resource name	Instance name	Description
clusterversion.config.openshift.io	version	In OpenShift Container Platform 4.1, you must not customize the ClusterVersion resource for production clusters. Instead, follow the process to update a cluster .
dns.config.openshift.io	cluster	You cannot modify the DNS settings for your cluster. You can view the DNS Operator status .
infrastructure.config.openshift.io	cluster	Configuration details allowing the cluster to interact with its cloud provider.
network.config.openshift.io	cluster	You cannot modify your cluster networking after installation. To customize your network, follow the process to customize networking during installation .

6.2. CONFIGURING YOUR FIREWALL

If you use a firewall, you must configure it so that OpenShift Container Platform can access Red Hat Insights. This access is required for OpenShift Container Platform to access the Telemetry service, which is required to receive updates.

6.2.1. Configuring your firewall for OpenShift Container Platform

Before you install OpenShift Container Platform, you must configure your firewall to access Red Hat Insights.

Procedure

- Set your firewall to allow the following host names and ports on the outgoing network firewall:

```
cert-api.access.redhat.com:443
api.access.redhat.com:443
infogw.api.openshift.com:443
```