



OpenShift Container Platform 4.2

Installing

Installing OpenShift Container Platform 4.2 clusters

OpenShift Container Platform 4.2 Installing

Installing OpenShift Container Platform 4.2 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 4.2 clusters on all supported infrastructure types.

Table of Contents

CHAPTER 1. INSTALLING ON AWS	10
1.1. CONFIGURING AN AWS ACCOUNT	10
1.1.1. Configuring Route53	10
1.1.2. AWS account limits	10
1.1.3. Required AWS permissions	12
1.1.4. Creating an IAM user	18
1.1.5. Supported AWS regions	19
1.2. INSTALLING A CLUSTER QUICKLY ON AWS	20
1.2.1. Internet and Telemetry access for OpenShift Container Platform	20
1.2.2. Generating an SSH private key and adding it to the agent	20
1.2.3. Obtaining the installation program	21
1.2.4. Deploy the cluster	22
1.2.5. Installing the CLI	24
1.2.6. Logging in to the cluster	24
1.3. INSTALLING A CLUSTER ON AWS WITH CUSTOMIZATIONS	25
1.3.1. Internet and Telemetry access for OpenShift Container Platform	25
1.3.2. Generating an SSH private key and adding it to the agent	26
1.3.3. Obtaining the installation program	27
1.3.4. Creating the installation configuration file	27
1.3.4.1. Installation configuration parameters	28
1.3.4.2. Sample customized install-config.yaml file for AWS	32
1.3.5. Deploy the cluster	34
1.3.6. Installing the CLI	36
1.3.7. Logging in to the cluster	36
1.4. INSTALLING A CLUSTER ON AWS WITH NETWORK CUSTOMIZATIONS	37
1.4.1. Internet and Telemetry access for OpenShift Container Platform	38
1.4.2. Generating an SSH private key and adding it to the agent	38
1.4.3. Obtaining the installation program	39
1.4.4. Creating the installation configuration file	40
1.4.4.1. Installation configuration parameters	41
1.4.4.2. Network configuration parameters	45
1.4.4.3. Sample customized install-config.yaml file for AWS	46
1.4.5. Modifying advanced network configuration parameters	47
1.4.6. Cluster Network Operator custom resource (CR)	48
1.4.6.1. Configuration parameters for OpenShift SDN	49
1.4.6.2. Configuration parameters for Open Virtual Network (OVN) SDN	50
1.4.6.3. Cluster Network Operator example CR	50
1.4.7. Deploy the cluster	50
1.4.8. Installing the CLI	52
1.4.9. Logging in to the cluster	53
1.5. UNINSTALLING A CLUSTER ON AWS	53
1.5.1. Removing a cluster that uses installer-provisioned infrastructure	53
CHAPTER 2. INSTALLING ON AZURE	55
2.1. CONFIGURING AN AZURE ACCOUNT	55
2.1.1. Azure account limits	55
2.1.2. Configuring a public DNS zone in Azure	57
2.1.3. Increasing Azure account limits	58
2.1.4. Required Azure roles	59
2.1.5. Creating a service principal	59
2.1.6. Supported Azure regions	61

2.2. INSTALLING A CLUSTER QUICKLY ON AZURE	62
2.2.1. Internet and Telemetry access for OpenShift Container Platform	63
2.2.2. Generating an SSH private key and adding it to the agent	63
2.2.3. Obtaining the installation program	64
2.2.4. Deploy the cluster	65
2.2.5. Installing the CLI	67
2.2.6. Logging in to the cluster	67
2.3. INSTALLING A CLUSTER ON AZURE WITH CUSTOMIZATIONS	68
2.3.1. Internet and Telemetry access for OpenShift Container Platform	68
2.3.2. Generating an SSH private key and adding it to the agent	69
2.3.3. Obtaining the installation program	70
2.3.4. Creating the installation configuration file	70
2.3.4.1. Installation configuration parameters	72
2.3.4.2. Sample customized install-config.yaml file for Azure	75
2.3.5. Deploy the cluster	77
2.3.6. Installing the CLI	78
2.3.7. Logging in to the cluster	78
2.4. INSTALLING A CLUSTER ON AZURE WITH NETWORK CUSTOMIZATIONS	79
2.4.1. Internet and Telemetry access for OpenShift Container Platform	79
2.4.2. Generating an SSH private key and adding it to the agent	80
2.4.3. Obtaining the installation program	81
2.4.4. Creating the installation configuration file	81
2.4.4.1. Installation configuration parameters	82
2.4.4.2. Network configuration parameters	86
2.4.4.3. Sample customized install-config.yaml file for Azure	87
2.4.5. Modifying advanced network configuration parameters	88
2.4.6. Cluster Network Operator custom resource (CR)	89
2.4.6.1. Configuration parameters for OpenShift SDN	90
2.4.6.2. Configuration parameters for Open Virtual Network (OVN) SDN	91
2.4.6.3. Cluster Network Operator example CR	91
2.4.7. Deploy the cluster	91
2.4.8. Installing the CLI	92
2.4.9. Logging in to the cluster	93
2.5. UNINSTALLING A CLUSTER ON AZURE	94
2.5.1. Removing a cluster that uses installer-provisioned infrastructure	94
CHAPTER 3. INSTALLING ON GCP	95
3.1. CONFIGURING A GCP PROJECT	95
3.1.1. Creating a GCP project	95
3.1.2. Enabling API services in GCP	95
3.1.3. Configuring DNS for GCP	96
3.1.4. GCP account limits	96
3.1.5. Creating a service account in GCP	97
3.1.5.1. Required GCP permissions	98
3.1.6. Supported GCP regions	99
3.2. INSTALLING A CLUSTER QUICKLY ON GCP	99
3.2.1. Internet and Telemetry access for OpenShift Container Platform	100
3.2.2. Generating an SSH private key and adding it to the agent	100
3.2.3. Obtaining the installation program	101
3.2.4. Deploy the cluster	102
3.2.5. Installing the CLI	103
3.2.6. Logging in to the cluster	104
3.3. INSTALLING A CLUSTER ON GCP WITH CUSTOMIZATIONS	105

3.3.1. Internet and Telemetry access for OpenShift Container Platform	105
3.3.2. Generating an SSH private key and adding it to the agent	105
3.3.3. Obtaining the installation program	106
3.3.4. Creating the installation configuration file	107
3.3.4.1. Installation configuration parameters	108
3.3.4.2. Sample customized install-config.yaml file for GCP	112
3.3.5. Deploy the cluster	113
3.3.6. Installing the CLI	114
3.3.7. Logging in to the cluster	115
3.4. UNINSTALLING A CLUSTER ON GCP	115
3.4.1. Removing a cluster that uses installer-provisioned infrastructure	115
CHAPTER 4. INSTALLING ON USER-PROVISIONED AWS	117
4.1. INSTALLING A CLUSTER ON USER-PROVISIONED INFRASTRUCTURE IN AWS BY USING CLOUDFORMATION TEMPLATES	117
4.1.1. Internet and Telemetry access for OpenShift Container Platform	117
4.1.2. Required AWS infrastructure components	118
4.1.2.1. Cluster machines	118
4.1.2.2. Other infrastructure components	119
4.1.2.3. Required AWS permissions	125
4.1.3. Obtaining the installation program	131
4.1.4. Generating an SSH private key and adding it to the agent	131
4.1.5. Creating the installation files for AWS	132
4.1.5.1. Creating the installation configuration file	132
4.1.5.2. Configuring the cluster-wide proxy during installation	134
4.1.5.3. Creating the Kubernetes manifest and Ignition config files	135
4.1.6. Extracting the infrastructure name	137
4.1.7. Creating a VPC in AWS	137
4.1.7.1. CloudFormation template for the VPC	139
4.1.8. Creating networking and load balancing components in AWS	144
4.1.8.1. CloudFormation template for the network and load balancers	147
4.1.9. Creating security group and roles in AWS	154
4.1.9.1. CloudFormation template for security objects	157
4.1.10. RHCOS AMIs for the AWS infrastructure	163
4.1.11. Creating the bootstrap node in AWS	164
4.1.11.1. CloudFormation template for the bootstrap machine	168
4.1.12. Creating the control plane machines in AWS	172
4.1.12.1. CloudFormation template for control plane machines	176
4.1.13. Initializing the bootstrap node on AWS with user-provisioned infrastructure	183
4.1.13.1. Creating the worker nodes in AWS	184
4.1.13.1.1. CloudFormation template for worker machines	188
4.1.14. Installing the CLI	190
4.1.15. Logging in to the cluster	191
4.1.16. Approving the CSRs for your machines	191
4.1.17. Initial Operator configuration	193
4.1.17.1. Image registry storage configuration	193
4.1.17.1.1. Configuring registry storage for AWS with user-provisioned infrastructure	194
4.1.17.1.2. Configuring storage for the image registry in non-production clusters	194
4.1.18. Completing an AWS installation on user-provisioned infrastructure	195
CHAPTER 5. INSTALLING ON USER-PROVISIONED GCP	197
5.1. INSTALLING A CLUSTER ON GCP USING DEPLOYMENT MANAGER TEMPLATES	197
5.1.1. Configuring your GCP project	197

5.1.1.1. Creating a GCP project	197
5.1.1.2. Enabling API services in GCP	197
5.1.1.3. Configuring DNS for GCP	198
5.1.1.4. GCP account limits	199
5.1.1.5. Creating a service account in GCP	199
5.1.1.5.1. Required GCP permissions	200
5.1.1.6. Supported GCP regions	201
5.1.1.7. Installing and configuring CLI tools for GCP	201
5.1.2. Creating the installation files for GCP	202
5.1.2.1. Creating the installation configuration file	202
5.1.2.2. Configuring the cluster-wide proxy during installation	203
5.1.2.3. Creating the Kubernetes manifest and Ignition config files	205
5.1.3. Exporting common variables	207
5.1.3.1. Extracting the infrastructure name	207
5.1.3.2. Exporting common variables for Deployment Manager templates	207
5.1.4. Creating a VPC in GCP	208
5.1.4.1. Deployment Manager template for the VPC	209
5.1.5. Creating networking and load balancing components in GCP	211
5.1.5.1. Deployment Manager template for the network and load balancers	212
5.1.6. Creating firewall rules and IAM roles in GCP	214
5.1.6.1. Deployment Manager template for firewall rules and IAM roles	215
5.1.7. Creating the RHCOS cluster image for the GCP infrastructure	218
5.1.8. Creating the bootstrap machine in GCP	219
5.1.8.1. Deployment Manager template for the bootstrap machine	221
5.1.9. Creating the control plane machines in GCP	222
5.1.9.1. Deployment Manager template for control plane machines	224
5.1.10. Wait for bootstrap completion and remove bootstrap resources in GCP	226
5.1.11. Creating additional worker machines in GCP	227
5.1.11.1. Deployment Manager template for worker machines	229
5.1.12. Installing the CLI	230
5.1.13. Logging in to the cluster	230
5.1.14. Approving the CSRs for your machines	231
5.1.15. Optional: Adding the ingress DNS records	232
5.1.16. Completing a GCP installation on user-provisioned infrastructure	234
CHAPTER 6. INSTALLING ON BARE METAL	237
6.1. INSTALLING A CLUSTER ON BARE METAL	237
6.1.1. Internet and Telemetry access for OpenShift Container Platform	237
6.1.2. Machine requirements for a cluster with user-provisioned infrastructure	238
6.1.2.1. Required machines	238
6.1.2.2. Network connectivity requirements	238
6.1.2.3. Minimum resource requirements	238
6.1.2.4. Certificate signing requests management	239
6.1.3. Creating the user-provisioned infrastructure	239
6.1.3.1. Networking requirements for user-provisioned infrastructure	239
Network topology requirements	240
6.1.3.2. User-provisioned DNS requirements	241
6.1.4. Generating an SSH private key and adding it to the agent	244
6.1.5. Obtaining the installation program	245
6.1.6. Installing the CLI	245
6.1.7. Manually creating the installation configuration file	246
6.1.7.1. Sample install-config.yaml file for bare metal	246
6.1.7.2. Configuring the cluster-wide proxy during installation	248

6.1.8. Creating the Kubernetes manifest and Ignition config files	250
6.1.9. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines	251
6.1.9.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image	251
6.1.9.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE or iPXE booting	252
6.1.10. Creating the cluster	254
6.1.11. Logging in to the cluster	255
6.1.12. Approving the CSRs for your machines	256
6.1.13. Initial Operator configuration	257
6.1.13.1. Image registry storage configuration	258
6.1.13.1.1. Configuring registry storage for bare metal	258
6.1.13.1.2. Configuring storage for the image registry in non-production clusters	259
6.1.13.14. Completing installation on user-provisioned infrastructure	260
CHAPTER 7. INSTALLING ON OPENSTACK	262
7.1. INSTALLING A CLUSTER ON OPENSTACK WITH CUSTOMIZATIONS	262
7.1.1. Resource guidelines for installing OpenShift Container Platform on OpenStack	262
7.1.1.1. Control plane and compute machines	263
7.1.1.2. Bootstrap machine	263
7.1.2. Internet and Telemetry access for OpenShift Container Platform	264
7.1.3. Enabling Swift on OpenStack	264
7.1.4. Creating the Red Hat Enterprise Linux CoreOS (RHCOS) image	265
7.1.5. Verifying external network access	265
7.1.6. Defining parameters for the installation program	266
7.1.7. Obtaining the installation program	267
7.1.8. Creating the installation configuration file	268
7.1.9. Installation configuration parameters	269
7.1.9.1. Sample customized install-config.yaml file for OpenStack	273
7.1.10. Generating an SSH private key and adding it to the agent	274
7.1.11. Enabling access to the environment	275
7.1.11.1. Enabling access with floating IP addresses	275
7.1.11.2. Enabling access without floating IP addresses	276
7.1.12. Deploy the cluster	276
7.1.13. Verifying cluster status	277
7.1.14. Logging in to the cluster	278
7.1.15. Configuring application access with floating IP addresses	278
CHAPTER 8. INSTALLING ON VSphere	280
8.1. INSTALLING A CLUSTER ON VSphere	280
8.1.1. Internet and Telemetry access for OpenShift Container Platform	280
8.1.2. VMware vSphere infrastructure requirements	280
8.1.3. Machine requirements for a cluster with user-provisioned infrastructure	281
8.1.3.1. Required machines	281
8.1.3.2. Network connectivity requirements	281
8.1.3.3. Minimum resource requirements	281
8.1.3.4. Certificate signing requests management	282
8.1.4. Creating the user-provisioned infrastructure	282
8.1.4.1. Networking requirements for user-provisioned infrastructure	282
Network topology requirements	283
Ethernet adaptor hardware address requirements	284
8.1.4.2. User-provisioned DNS requirements	284
8.1.5. Generating an SSH private key and adding it to the agent	287
8.1.6. Obtaining the installation program	288
8.1.7. Manually creating the installation configuration file	288

8.1.7.1. Sample install-config.yaml file for VMware vSphere	289
8.1.7.2. Configuring the cluster-wide proxy during installation	290
8.1.8. Creating the Kubernetes manifest and Ignition config files	292
8.1.9. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines in vSphere	293
8.1.10. Installing the CLI	296
8.1.11. Creating the cluster	297
8.1.12. Logging in to the cluster	297
8.1.13. Approving the CSRs for your machines	298
8.1.14. Initial Operator configuration	299
8.1.14.1. Image registry storage configuration	300
8.1.14.1.1. Configuring registry storage for VMware vSphere	300
8.1.14.1.2. Configuring storage for the image registry in non-production clusters	301
8.1.15. Completing installation on user-provisioned infrastructure	302
CHAPTER 9. INSTALLING IN RESTRICTED NETWORKS	305
9.1. CREATING A MIRROR REGISTRY FOR INSTALLATION IN A RESTRICTED NETWORK	305
9.1.1. About the mirror registry	305
9.1.2. Preparing the bastion host	305
9.1.2.1. Installing the CLI	305
9.1.3. Creating a mirror registry	306
9.1.4. Creating a pull secret for your mirror registry	308
9.1.5. Mirroring the OpenShift Container Platform image repository	309
9.1.6. Using sample imagestreams in a restricted network installation	311
9.2. INSTALLING A CLUSTER ON AWS THAT USES MIRRORED INSTALLATION CONTENT	312
9.2.1. About installations in restricted networks	313
9.2.1.1. Additional limits	313
9.2.2. Internet and Telemetry access for OpenShift Container Platform	313
9.2.3. Required AWS infrastructure components	314
9.2.3.1. Cluster machines	314
9.2.3.2. Other infrastructure components	315
9.2.3.3. Required AWS permissions	321
9.2.4. Generating an SSH private key and adding it to the agent	327
9.2.5. Creating the installation files for AWS	328
9.2.5.1. Creating the installation configuration file	328
9.2.5.2. Configuring the cluster-wide proxy during installation	330
9.2.5.3. Creating the Kubernetes manifest and Ignition config files	331
9.2.6. Extracting the infrastructure name	333
9.2.7. Creating a VPC in AWS	333
9.2.7.1. CloudFormation template for the VPC	335
9.2.8. Creating networking and load balancing components in AWS	340
9.2.8.1. CloudFormation template for the network and load balancers	343
9.2.9. Creating security group and roles in AWS	351
9.2.9.1. CloudFormation template for security objects	353
9.2.10. RHCOS AMIs for the AWS infrastructure	359
9.2.11. Creating the bootstrap node in AWS	360
9.2.11.1. CloudFormation template for the bootstrap machine	364
9.2.12. Creating the control plane machines in AWS	368
9.2.12.1. CloudFormation template for control plane machines	372
9.2.13. Initializing the bootstrap node on AWS with user-provisioned infrastructure	380
9.2.13.1. Creating the worker nodes in AWS	380
9.2.13.1.1. CloudFormation template for worker machines	384
9.2.14. Logging in to the cluster	386
9.2.15. Approving the CSRs for your machines	387

9.2.16. Initial Operator configuration	388
9.2.16.1. Image registry storage configuration	389
9.2.16.1.1. Configuring registry storage for AWS with user-provisioned infrastructure	389
9.2.16.1.2. Configuring storage for the image registry in non-production clusters	390
9.2.17. Completing an AWS installation on user-provisioned infrastructure	390
9.3. INSTALLING A CLUSTER ON BARE METAL IN A RESTRICTED NETWORK	391
9.3.1. About installations in restricted networks	392
9.3.1.1. Additional limits	393
9.3.2. Internet and Telemetry access for OpenShift Container Platform	393
9.3.3. Machine requirements for a cluster with user-provisioned infrastructure	393
9.3.3.1. Required machines	393
9.3.3.2. Network connectivity requirements	394
9.3.3.3. Minimum resource requirements	394
9.3.3.4. Certificate signing requests management	394
9.3.4. Creating the user-provisioned infrastructure	394
9.3.4.1. Networking requirements for user-provisioned infrastructure	395
Network topology requirements	396
9.3.4.2. User-provisioned DNS requirements	396
9.3.5. Generating an SSH private key and adding it to the agent	399
9.3.6. Manually creating the installation configuration file	400
9.3.6.1. Sample install-config.yaml file for bare metal	400
9.3.6.2. Configuring the cluster-wide proxy during installation	402
9.3.7. Creating the Kubernetes manifest and Ignition config files	404
9.3.8. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines	405
9.3.8.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image	405
9.3.8.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE or iPXE booting	406
9.3.9. Creating the cluster	409
9.3.10. Logging in to the cluster	410
9.3.11. Approving the CSRs for your machines	410
9.3.12. Initial Operator configuration	411
9.3.12.1. Image registry storage configuration	412
9.3.12.1.1. Configuring registry storage for bare metal	412
9.3.12.1.2. Configuring storage for the image registry in non-production clusters	413
9.3.13. Completing installation on user-provisioned infrastructure	414
9.4. INSTALLING A CLUSTER ON VSphere IN A RESTRICTED NETWORK	416
9.4.1. About installations in restricted networks	416
9.4.1.1. Additional limits	417
9.4.2. Internet and Telemetry access for OpenShift Container Platform	417
9.4.3. VMware vSphere infrastructure requirements	417
9.4.4. Machine requirements for a cluster with user-provisioned infrastructure	418
9.4.4.1. Required machines	418
9.4.4.2. Network connectivity requirements	418
9.4.4.3. Minimum resource requirements	418
9.4.4.4. Certificate signing requests management	419
9.4.5. Creating the user-provisioned infrastructure	419
9.4.5.1. Networking requirements for user-provisioned infrastructure	419
Network topology requirements	420
9.4.5.2. User-provisioned DNS requirements	421
9.4.6. Generating an SSH private key and adding it to the agent	424
9.4.7. Manually creating the installation configuration file	425
9.4.7.1. Sample install-config.yaml file for VMware vSphere	425
9.4.7.2. Configuring the cluster-wide proxy during installation	427
9.4.8. Creating the Kubernetes manifest and Ignition config files	429

9.4.9. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines in vSphere	430
9.4.10. Creating the cluster	433
9.4.11. Logging in to the cluster	434
9.4.12. Approving the CSRs for your machines	434
9.4.13. Initial Operator configuration	436
9.4.13.1. Image registry storage configuration	436
9.4.13.1.1. Configuring registry storage for VMware vSphere	437
9.4.13.1.2. Configuring storage for the image registry in non-production clusters	438
9.4.13.14. Completing installation on user-provisioned infrastructure	438
CHAPTER 10. GATHERING INSTALLATION LOGS	441
10.1. GATHERING LOGS FROM A FAILED INSTALLATION	441
CHAPTER 11. INSTALLATION CONFIGURATION	443
11.1. AVAILABLE CLUSTER CUSTOMIZATIONS	443
11.1.1. Cluster configuration resources	443
11.1.2. Operator configuration resources	444
11.1.3. Additional configuration resources	444
11.1.4. Informational Resources	444
11.2. CONFIGURING YOUR FIREWALL	445
11.2.1. Configuring your firewall for OpenShift Container Platform	445

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 your 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 uses 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> <p> IMPORTANT</p> <p>To use the us-east-1 region, you must increase the EIP limit for your account.</p>
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	<p>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.</p> <p>Additional ENIs are created for additional machines and elastic load balancers that are created by cluster usage and deployed workloads.</p>
VPC Gateway	20	20 per account	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**
- **iamPassRole**
- **iamPutRolePolicy**
- **iamRemoveRoleFromInstanceProfile**
- **iamSimulatePrincipalPolicy**
- **iamTagRole**

Required Route53 permissions for installation

- **route53ChangeResourceRecordSets**
- **route53ChangeTagsForResource**
- **route53GetChange**
- **route53GetHostedZone**
- **route53CreateHostedZone**
- **route53ListHostedZones**
- **route53ListHostedZonesByName**
- **route53ListResourceRecordSets**
- **route53ListTagsForResource**
- **route53UpdateHostedZoneComment**

Required S3 permissions for installation

- **s3CreateBucket**
- **s3DeleteBucket**
- **s3GetAccelerateConfiguration**
- **s3GetBucketCors**
- **s3GetBucketLocation**
- **s3GetBucketLogging**
- **s3GetBucketObjectLockConfiguration**
- **s3GetBucketReplication**
- **s3GetBucketRequestPayment**
- **s3GetBucketTagging**
- **s3GetBucketVersioning**

- **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 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:
 - [Quickly install a cluster](#) with default options
 - [Install a cluster with cloud customizations](#)
 - [Install a cluster with network customizations](#)

- [Install a cluster on infrastructure that you provision](#)

1.2. INSTALLING A CLUSTER QUICKLY ON AWS

In OpenShift Container Platform version 4.2, 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.

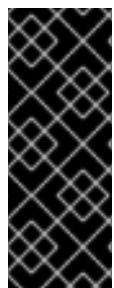
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.

1.2.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.2, you require access to the internet to install and entitle your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager](#). From there, you can allocate entitlements to your cluster.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management and entitlement. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster. If the Telemetry service cannot entitle your cluster, you must manually entitle it on the [Cluster registration](#) page.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

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

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.

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 installation program.

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 500 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 **create cluster** command of 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=info ②
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



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 on, specify an SSH key that your **ssh-agent** process uses.

- 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

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



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 CLI

You can install the CLI in order to interact with OpenShift Container Platform using a command-line interface.

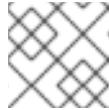


IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.2. Download and install the new version of **oc**.

Procedure

1. From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**.
2. Click the folder for your operating system and architecture and click the compressed file.



NOTE

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

3. Save the file to your file system.
4. Extract the compressed file.
5. Place it in a directory that is on your **PATH**.

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.2.6. Logging in to the cluster

You can log in to 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
```

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

2. Verify you can run **oc** commands successfully using the exported configuration:

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

Next steps

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

1.3. INSTALLING A CLUSTER ON AWS WITH CUSTOMIZATIONS

In OpenShift Container Platform version 4.2, you can install a customized cluster on infrastructure that the installation program provisions on Amazon Web Services (AWS). To customize the installation, you modify 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 allow the sites](#) that your cluster requires access to.

1.3.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.2, you require access to the internet to install and entitle your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager](#). From there, you can allocate entitlements to your cluster.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management and entitlement. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster. If the Telemetry service cannot entitle your cluster, you must manually entitle it on the [Cluster registration](#) page.

- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

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

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.

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 installation program.

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 500 MB of local disk space to download the installation program.

Procedure

- 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.
- 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.

- 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
```

- 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 Amazon Web Services (AWS).

Prerequisites

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

Procedure

- Create the **install-config.yaml** file.
 - 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.

- At the prompts, provide the configuration details for your cloud:
 - Optional: Select an SSH key to use to access your cluster machines.
 - Select **AWS** as the platform to target.
 - 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.
 - Select the AWS region to deploy the cluster to.
 - Select the base domain for the Route53 service that you configured for your cluster.
 - Enter a descriptive name for your cluster.
 - Paste the pull secret that you obtained from the [OpenShift Infrastructure Providers](#) page.
- Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
- 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 account on the cloud platform that hosts your cluster 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 in the **install-config.yaml** file after installation.

Table 1.1. 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, azure, gcp, openstack, or {}
compute.platform	The cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, or {}
metadata.name	The name of your cluster.	A string that contains uppercase or lowercase letters, such as dev .
platform.<platform>.region	The region to deploy your cluster in.	A valid region for your cloud, such as us-east-1 for AWS, centralus for Azure, or region1 for Red Hat OpenStack Platform (RHOSP).

Parameter	Description	Values
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 parameters

Parameter	Description	Values
sshKey	The SSH key to use to access your cluster machines.	A valid, local public SSH key that you added to the ssh-agent process.

NOTE



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

Parameter	Description	Values
compute.hyperthreading	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.	Enabled or Disabled
	<p> IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p>	
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.hyperthreading	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.	Enabled or Disabled
	<p> IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p>	
controlPlane.replicas	The number of control plane machines to provision.	A positive integer greater than or equal to 3 . The default value is 3 .

Table 1.3. Optional AWS parameters

Parameter	Description	Values
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 .
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 .
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 .
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 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:
  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 10
    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 on, specify an SSH key that your **ssh-agent** process uses.

1.3.5. Deploy the cluster

You can install OpenShift Container Platform on a compatible cloud.



IMPORTANT

You can run the **create cluster** command of 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

- Run the installation program:

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

- 1 For <installation_directory>, specify the location of your customized **./install-config.yaml** file. directory name to store the files that the installation program creates.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



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 on, specify an SSH key that your **ssh-agent** process uses.

- 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**

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

**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.

1.3.6. Installing the CLI

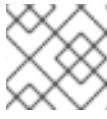
You can install the CLI in order to interact with OpenShift Container Platform using a command-line interface.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.2. Download and install the new version of **oc**.

Procedure

1. From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**.
2. Click the folder for your operating system and architecture and click the compressed file.
3. Save the file to your file system.
4. Extract the compressed file.
5. Place it in a directory that is on your **PATH**.

**NOTE**

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

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.3.7. Logging in to the cluster

You can log in to 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
```

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

2. Verify you can run **oc** commands successfully using the exported configuration:

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

Next steps

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

1.4. INSTALLING A CLUSTER ON AWS WITH NETWORK CUSTOMIZATIONS

In OpenShift Container Platform version 4.2, 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 allow the sites](#) that your cluster requires access to.

1.4.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.2, you require access to the internet to install and entitle your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager](#). From there, you can allocate entitlements to your cluster.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management and entitlement. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster. If the Telemetry service cannot entitle your cluster, you must manually entitle it on the [Cluster registration](#) page.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

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

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.

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.

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

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

```
Agent pid 31874
```

- 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 installation program.

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 500 MB of local disk space to download the installation program.

Procedure

- 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.
- 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.

- 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
```

- 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 Amazon Web Services (AWS).

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 on, specify an SSH key that your **ssh-agent** process uses.

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

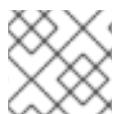


IMPORTANT

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 account on the cloud platform that hosts your cluster 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 in the **install-config.yaml** file after installation.

Table 1.4. 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 <code><metadata.name>.<baseDomain></code> 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, azure, gcp, openstack , or {}
compute.platform	The cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack , or {}
metadata.name	The name of your cluster.	A string that contains uppercase or lowercase letters, such as dev .

Parameter	Description	Values
platform.<platform>.region	The region to deploy your cluster in.	A valid region for your cloud, such as us-east-1 for AWS, centralus for Azure, or region1 for Red Hat OpenStack Platform (RHOSP).
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 1.5. Optional parameters

Parameter	Description	Values
sshKey	The SSH key to use to access your cluster machines.	A valid, local public SSH key that you added to the ssh-agent process.

NOTE

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

Parameter	Description	Values
compute.hyperthreading	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.	Enabled or Disabled
	 IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	
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.hyperthreading	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.	Enabled or Disabled
	 IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	
controlPlane.replicas	The number of control plane machines to provision.	A positive integer greater than or equal to 3 . The default value is 3 .

Table 1.6. Optional AWS parameters

Parameter	Description	Values
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 .
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 .
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 .
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.



IMPORTANT

The Open Virtual Networking (OVN) Kubernetes network plug-in is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of the OVN Technology Preview, see <https://access.redhat.com/articles/4380121>.

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 in the **install-config.yaml** file after installation.

Table 1.7. Required network parameters

Parameter	Description	Value
networking.networkType	The network plug-in to deploy. The OpenShiftSDN plug-in is the only plug-in supported in OpenShift Container Platform 4.2. The OVNKubernetes plug-in is available as Technology Preview in OpenShift Container Platform 4.2.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork.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.clusterNetwork.hostPrefix	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.serviceNetwork	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.machineCIDR	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: 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 on, specify an SSH key that your **ssh-agent** process uses.

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, several network configuration files are in the **manifests** directory, as shown:

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

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** parameter are only an example. Specify your configuration for the Cluster Network Operator in the CR.

The CNO provides default values for the parameters in the CR, so you must specify only the parameters that you want to change.

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 Operator manages the cluster network.

You can specify the cluster network configuration for your OpenShift Container Platform cluster by setting the parameters for the **defaultNetwork** parameter in the CNO CR. 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: ①
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  serviceNetwork: ②
    - 172.30.0.0/16
  defaultNetwork: ③
...
  ...
  kubeProxyConfig: ④
    iptablesSyncPeriod: 30s ⑤
  proxyArguments:
    iptables-min-sync-period: ⑥
      - 30s
```

① ② Specified in the **install-config.yaml** file.

③ Configures the software-defined networking (SDN) for the cluster network.

④ 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.

⑤ 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.

⑥ 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](#)

1.4.6.1. Configuration parameters for OpenShift SDN

The following YAML object describes the configuration parameters for OpenShift SDN:

```
defaultNetwork:
  type: OpenShiftSDN ①
  openshiftSDNConfig: ②
  mode: NetworkPolicy ③
  mtu: 1450 ④
  vxlanPort: 4789 ⑤
```

- 1 Specified in the **install-config.yaml** file.
- 2 Specify only if you want to override part of the OpenShift SDN configuration.
- 3 Configures the network isolation mode for **OpenShiftSDN**. The allowed values are **Multitenant**, **Subnet**, or **NetworkPolicy**. The default value is **NetworkPolicy**.
- 4 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.
- 5 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**.

1.4.6.2. Configuration parameters for Open Virtual Network (OVN) SDN

The OVN SDN does not have any configuration parameters in OpenShift Container Platform 4.2.

1.4.6.3. Cluster Network Operator example CR

A complete CR for the CNO is displayed in the following example:

Cluster Network Operator example CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  serviceNetwork:
    - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
  kubeProxyConfig:
    iptablesSyncPeriod: 30s
    proxyArguments:
      iptables-min-sync-period:
        - 30s
```

1.4.7. Deploy the cluster

You can install OpenShift Container Platform on a compatible cloud.



IMPORTANT

You can run the **create cluster** command of 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=info ②
```

- 1 For **<installation_directory>**, specify the location of your customized **.install-config.yaml** file. directory name to store the files that the installation program creates.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



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 on, specify an SSH key that your **ssh-agent** process uses.

- 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**

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

**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 CLI

You can install the CLI in order to interact with OpenShift Container Platform using a command-line interface.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.2. Download and install the new version of **oc**.

Procedure

1. From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**.
2. Click the folder for your operating system and architecture and click the compressed file.

**NOTE**

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

3. Save the file to your file system.
4. Extract the compressed file.
5. Place it in a directory that is on your **PATH**.

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.4.9. Logging in to the cluster

You can log in to 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
```

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

2. Verify you can run **oc** commands successfully using the exported configuration:

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

Next steps

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

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 that uses installer-provisioned infrastructure

You can remove a cluster that uses installer-provisioned infrastructure from your cloud.

Prerequisites

- Have a copy of the installation program that you used to deploy the cluster.
- Have the files that the installation program generated when you created your cluster.

Procedure

- 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
```

- 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.

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

CHAPTER 2. INSTALLING ON AZURE

2.1. CONFIGURING AN AZURE ACCOUNT

Before you can install OpenShift Container Platform, you must configure a Microsoft Azure account.



IMPORTANT

All Azure resources that are available through public endpoints are subject to resource name restrictions, and you cannot create resources that use certain terms. For a list of terms that Azure restricts, see [Resolve reserved resource name errors](#) in the Azure documentation.

2.1.1. Azure account limits

The OpenShift Container Platform cluster uses a number of Microsoft Azure components, and the default [Azure subscription and service limits, quotas, and constraints](#) affect your ability to install OpenShift Container Platform clusters.



IMPORTANT

You must increase quota limits for your account before you install a default cluster on Azure.

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

Component	Number of components required by default	Default Azure limit	Description

Component	Number of components required by default	Default Azure limit	Description
vCPU	34	20 per region	<p>A default cluster requires 34 vCPUs, so you must increase the account limit.</p> <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 control plane machines ● Three compute machines <p>Because the bootstrap machine uses Standard_D4s_v3 machines, which use 4 vCPUs, the control plane machines use Standard_D8s_v3 virtual machines, which use 8 vCPUs, and the worker machines use Standard_D2s_v3 virtual machines, which use 2 vCPUs, a default cluster requires 34 vCPUs.</p> <p>To deploy more worker nodes, enable autoscaling, deploy large workloads, or use a different instance type, you must further increase the vCPU limit for your account to ensure that your cluster can deploy the machines that you require.</p> <p>By default, the installation program distributes control plane and compute machines across all availability zones within a region. To ensure high availability for your cluster, select a region with at least three availability zones. If your region contains fewer than three availability zones, the installation program places more than one control plane machine in the available zones.</p>
VNet	1	1000 per region	Each default cluster requires one Virtual Network (VNet), which contains two subnets.
Network interfaces	6	65,536 per region	Each default cluster requires six network interfaces. If you create more machines or your deployed workloads create load balancers, your cluster uses more network interfaces.

Component	Number of components required by default	Default Azure limit	Description						
Network security groups	2	5000	<p>Each default cluster Each cluster creates network security groups for each subnet in the VNet. The default cluster creates network security groups for the control plane and for the compute node subnets:</p> <table border="1"> <tr> <td>controlplane</td><td>Allows the control plane machines to be reached on port 6443 from anywhere</td></tr> <tr> <td>node</td><td>Allows worker nodes to be reached from the internet on ports 80 and 443</td></tr> </table>	controlplane	Allows the control plane machines to be reached on port 6443 from anywhere	node	Allows worker nodes to be reached from the internet on ports 80 and 443		
controlplane	Allows the control plane machines to be reached on port 6443 from anywhere								
node	Allows worker nodes to be reached from the internet on ports 80 and 443								
Network load balancers	3	1000 per region	<p>Each cluster creates the following load balancers:</p> <table border="1"> <tr> <td>default</td><td>Public IP address that load balances requests to ports 80 and 443 across worker machines</td></tr> <tr> <td>internal</td><td>Private IP address that load balances requests to ports 6443 and 22623 across control plane machines</td></tr> <tr> <td>external</td><td>Public IP address that load balances requests to port 6443 across control plane machines</td></tr> </table> <p>If your applications create more Kubernetes LoadBalancer Service objects, your cluster uses more load balancers.</p>	default	Public IP address that load balances requests to ports 80 and 443 across worker machines	internal	Private IP address that load balances requests to ports 6443 and 22623 across control plane machines	external	Public IP address that load balances requests to port 6443 across control plane machines
default	Public IP address that load balances requests to ports 80 and 443 across worker machines								
internal	Private IP address that load balances requests to ports 6443 and 22623 across control plane machines								
external	Public IP address that load balances requests to port 6443 across control plane machines								
Public IP addresses	3		Each of the two public load balancers uses a public IP address. The bootstrap machine also uses a public IP address so that you can SSH into the machine to troubleshoot issues during installation. The IP address for the bootstrap node is used only during installation.						
Private IP addresses	7		The internal loadbalancer, each of the three control plane machines, and each of the three worker machines each use a private IP address.						

2.1.2. Configuring a public DNS zone in Azure

To install OpenShift Container Platform, the Microsoft Azure account you use must have a dedicated public hosted DNS zone in your account. This zone must be authoritative for the domain. This 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 Azure or another source.



NOTE

For more information about purchasing domains through Azure, see [Buy a custom domain name for Azure App Service](#) in the Azure documentation.

2. If you are using an existing domain and registrar, migrate its DNS to Azure. See [Migrate an active DNS name to Azure App Service](#) in the Azure documentation.
3. Configure DNS for your domain. Follow the steps in the [Tutorial: Host your domain in Azure DNS](#) in the Azure documentation to create a public hosted zone for your domain or subdomain, extract the new authoritative name servers, and update the registrar records for the name servers that your domain uses.
Use an appropriate root domain, such as **openshiftcorp.com**, or subdomain, such as **clusters.openshiftcorp.com**.
4. If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain.

2.1.3. Increasing Azure account limits

To increase an account limit, file a support request on the Azure portal.



NOTE

You can increase only one type of quota per support request.

Procedure

1. From the Azure portal, click **Help + support** in the lower left corner.
2. Click **New support request** and then select the required values:
 - a. From the **Issue type** list, select **Service and subscription limits (quotas)**
 - b. From the **Subscription** list, select the subscription to modify.
 - c. From the **Quota type** list, select the quota to increase. For example, select **Compute-VM (cores-vCPUs) subscription limit increases** to increase the number of vCPUs, which is required to install a cluster.
 - d. Click **Next: Solutions**.
3. On the PROBLEM DETAILS page, provide the required information for your quota increase:
 - a. Click **Provide details** and provide the required details in the "Quota details" window.

- b. In the SUPPORT METHOD and CONTACT INFO sections, provide the issue severity and your contact details.
4. Click **Next: Review + create** and then click **Create**.

2.1.4. Required Azure roles

Your Microsoft Azure account must have the following roles for the subscription that you use: * **User Access Administrator**

To set roles on the Azure portal, see the [Manage access to Azure resources using RBAC and the Azure portal](#) in the Azure documentation.

2.1.5. Creating a service principal

Because OpenShift Container Platform and its installation program must create Microsoft Azure resources through Azure Resource Manager, you must create a service principal to represent it.

Prerequisites

- Install or update the [Azure CLI](#).
- Install the **jq** package.
- Your Azure account has the required roles for the subscription that you use.

Procedure

1. Log in to the Azure CLI:

```
$ az login
```

Log in to Azure in the web console by using your credentials.

2. If your Azure account uses subscriptions, ensure that you are using the right subscription.
 - a. View the list of available accounts and record the **tenantId** value for the subscription you want to use for your cluster:

```
$ az account list --refresh
[
  {
    "cloudName": "AzureCloud",
    "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
    "isDefault": true,
    "name": "Subscription Name",
    "state": "Enabled",
    "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee",
    "user": {
      "name": "you@example.com",
      "type": "user"
    }
  }
]
```

- b. View your active account details and confirm that the **tenantId** matches the subscription you want to use:

```
$ az account show
{
  "environmentName": "AzureCloud",
  "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee", 1
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

- 1** Ensure that the value of the **tenantId** parameter is the UUID of the correct subscription.

- c. If you are not using the right subscription, change the active subscription:

```
$ az account set -s <id> 1
```

- 1** Substitute the value of the **id** for the subscription that you want to use for **<id>**.

- d. If you changed the active subscription, display your account information again:

```
$ az account show
{
  "environmentName": "AzureCloud",
  "id": "33212d16-bdf6-45cb-b038-f6565b61edda",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee",
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

3. Record the values of the **tenantId** and **id** parameters from the previous output. You need these values during OpenShift Container Platform installation.
4. Create the service principal for your account:

```
$ az ad sp create-for-rbac --role Contributor --name <service_principal> 1
Changing "<service_principal>" to a valid URI of "http://<service_principal>", which is the required format used for service principal names
Retrying role assignment creation: 1/36
Retrying role assignment creation: 2/36
```

```

    Retrying role assignment creation: 3/36
    Retrying role assignment creation: 4/36
    {
      "appId": "8bd0d04d-0ac2-43a8-928d-705c598c6956",
      "displayName": "<service_principal>",
      "name": "http://<service_principal>",
      "password": "ac461d78-bf4b-4387-ad16-7e32e328aec6",
      "tenant": "6048c7e9-b2ad-488d-a54e-dc3f6be6a7ee"
    }
  
```

1 Replace **<service_principal>** with the name to assign to the service principal.

5. Record the values of the **appId** and **password** parameters from the previous output. You need these values during OpenShift Container Platform installation.
6. Grant additional permissions to the service principal. The service principal requires the legacy **Azure Active Directory Graph** → **Application.ReadWrite.OwnedBy** permission and the **User Access Administrator** role for the cluster to assign credentials for its components.
 - a. To assign the **User Access Administrator** role, run the following command:

```

$ az role assignment create --role "User Access Administrator" \
  --assignee-object-id $(az ad sp list --filter "appId eq '<appId>'" \ ①
  | jq '.[0].objectId' -r)
  
```

1 Replace **<appId>** with the **appId** parameter value for your service principal.

- b. To assign the **Azure Active Directory Graph** permission, run the following command:

```

$ az ad app permission add --id <appId> \ ①
  --api 00000002-0000-0000-c000-000000000000 \
  --api-permissions 824c81eb-e3f8-4ee6-8f6d-de7f50d565b7=Role
  
```

Invoking "az ad app permission grant --id 46d33abc-b8a3-46d8-8c84-f0fd58177435 --api 00000002-0000-0000-c000-000000000000" is needed to make the change effective

1 Replace **<appId>** with the **appId** parameter value for your service principal.

For more information about the specific permissions that you grant with this command, see the [GUID Table for Windows Azure Active Directory Permissions](#).

- c. Approve the permissions request. If your account does not have the Azure Active Directory tenant administrator role, follow the guidelines for your organization to request that the tenant administrator approve your permissions request.

```

$ az ad app permission grant --id <appId> \ ①
  --api 00000002-0000-0000-c000-000000000000
  
```

1 Replace **<appId>** with the **appId** parameter value for your service principal.

2.1.6. Supported Azure regions

The installation program dynamically generates the list of available Microsoft Azure regions based on your subscription. The following Azure regions were tested and validated in OpenShift Container Platform version 4.2.0:

- centralus (Central US)
- eastus (East US)
- eastus2 (East US 2)
- northcentralus (North Central US)
- southcentralus (South Central US)
- westcentralus (West Central US)
- westus (West US)
- westus2 (West US 2)
- uksouth (UK South)
- ukwest (UK West)
- francecentral (France Central)
- northeurope (North Europe)
- westeurope (West Europe)
- japaneast (Japan East)
- japanwest (Japan West)
- koreacentral (Korea Central)
- koreasouth (Korea South)
- eastasia (East Asia)
- southeastasia (Southeast Asia)
- southindia (South India)
- centralindia (Central India)
- westindia (West India)
- uaenorth (UAE North)

Next steps

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

2.2. INSTALLING A CLUSTER QUICKLY ON AZURE

In OpenShift Container Platform version 4.2, you can install a cluster on Microsoft Azure that uses the default configuration options.

Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure an Azure account](#) to host the cluster and determine the tested and validated region to deploy the cluster to.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.

2.2.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.2, you require access to the internet to install and entitle your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager](#). From there, you can allocate entitlements to your cluster.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management and entitlement. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster. If the Telemetry service cannot entitle your cluster, you must manually entitle it on the [Cluster registration](#) page.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

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

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.

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 installation program.

2.2.3. 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 500 MB of local disk space to download the installation program.

Procedure

- 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.
- 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.

2.2.4. Deploy the cluster

You can install OpenShift Container Platform on a compatible cloud.

**IMPORTANT**

You can run the **create cluster** command of 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=info ②
```

- ① For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file. directory name to store the files that the installation program creates.
- ② To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

**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 on, specify an SSH key that your **ssh-agent** process uses.

- b. Select **azure** as the platform to target.
- c. If you do not have a Microsoft Azure profile stored on your computer, specify the following Azure parameter values for your subscription and service principal:
 - **azure subscription id** The subscription ID to use for the cluster. Specify the **id** value in your account output.
 - **azure tenant id** The tenant ID. Specify the **tenantId** value in your account output.
 - **azure service principal client id** The value of the **appId** parameter for the service principal.
 - **azure service principal client secret** The value of the **password** parameter for the service principal.
- d. Select the region to deploy the cluster to.
- e. Select the base domain to deploy the cluster to. The base domain corresponds to the Azure DNS Zone that you created for your cluster.
- f. Enter a descriptive name for your cluster.



IMPORTANT

All Azure resources that are available through public endpoints are subject to resource name restrictions, and you cannot create resources that use certain terms. For a list of terms that Azure restricts, see [Resolve reserved resource name errors](#) in the Azure documentation.

- 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

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



IMPORTANT

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

2.2.5. Installing the CLI

You can install the CLI in order to interact with OpenShift Container Platform using a command-line interface.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.2. Download and install the new version of **oc**.

Procedure

1. From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**
2. Click the folder for your operating system and architecture and click the compressed file.
3. Save the file to your file system.
4. Extract the compressed file.
5. Place it in a directory that is on your **PATH**.

NOTE

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

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.2.6. Logging in to the cluster

You can log in to 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
```

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

2. Verify you can run **oc** commands successfully using the exported configuration:

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

Next steps

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

2.3. INSTALLING A CLUSTER ON AZURE WITH CUSTOMIZATIONS

In OpenShift Container Platform version 4.2, you can install a customized cluster on infrastructure that the installation program provisions on Microsoft Azure. To customize the installation, you modify 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 Azure account](#) to host the cluster and determine the tested and validated region to deploy the cluster to.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.

2.3.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.2, you require access to the internet to install and entitle your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager](#). From there, you can allocate entitlements to your cluster.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management and entitlement. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster. If the Telemetry service cannot entitle your cluster, you must manually entitle it on the [Cluster registration](#) page.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.

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



IMPORTANT

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

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.

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 installation program.

2.3.3. 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 500 MB of local disk space to download the installation program.

Procedure

- 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.
- 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.

- 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
```

- 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.

2.3.4. Creating the installation configuration file

You can customize your installation of OpenShift Container Platform on Microsoft Azure.

Prerequisites

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

Procedure

- Create the **install-config.yaml** file.
 - 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 on, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **azure** as the platform to target.
- iii. If you do not have a Microsoft Azure profile stored on your computer, specify the following Azure parameter values for your subscription and service principal:
 - **azure subscription id** The subscription ID to use for the cluster. Specify the **id** value in your account output.
 - **azure tenant id** The tenant ID. Specify the **tenantId** value in your account output.
 - **azure service principal client id** The value of the **appId** parameter for the service principal.
 - **azure service principal client secret** The value of the **password** parameter for the service principal.
- iv. Select the region to deploy the cluster to.
- v. Select the base domain to deploy the cluster to. The base domain corresponds to the Azure DNS Zone that you created for your cluster.
- vi. Enter a descriptive name for your cluster.



IMPORTANT

All Azure resources that are available through public endpoints are subject to resource name restrictions, and you cannot create resources that use certain terms. For a list of terms that Azure restricts, see [Resolve reserved resource name errors](#) in the Azure documentation.

- 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.
 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.

2.3.4.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster 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 in the **install-config.yaml** file after installation.

Table 2.1. 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 <code><metadata.name>. <baseDomain></code> 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, azure, gcp, openstack , or {}
compute.platform	The cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack , or {}

Parameter	Description	Values
metadata.name	The name of your cluster.	A string that contains uppercase or lowercase letters, such as dev .
platform.<platform>.region	The region to deploy your cluster in.	A valid region for your cloud, such as us-east-1 for AWS, centralus for Azure, or region1 for Red Hat OpenStack Platform (RHOSP).
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 2.2. Optional parameters

Parameter	Description	Values
sshKey	<p>The SSH key to use to access your cluster machines.</p>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery on, specify an SSH key that your ssh-agent process uses.</p>	A valid, local public SSH key that you added to the ssh-agent process.

Parameter	Description	Values
compute.hyperthreading	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.	Enabled or Disabled
	 IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	
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.hyperthreading	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.	Enabled or Disabled
	 IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	
controlPlane.replicas	The number of control plane machines to provision.	A positive integer greater than or equal to 3 . The default value is 3 .

Table 2.3. Additional Azure parameters

Parameter	Description	Values
machines.platform.azure.type	The Azure VM instance type.	VMs that use Windows or Linux as the operating system. See the Guest operating systems supported on Azure Stack in the Azure documentation.
machines.platform.azure.osDisk.diskSizeGB	The Azure disk size for the VM.	Integer that represents the size of the disk in GB, for example 512 . The minimum supported disk size is 120 .
platform.azure.baseDomainResourceGroupName	The name of the resource group that contains the DNS zone for your base domain.	String, for example production_cluster .
platform.azure.region	The name of the Azure region that hosts your cluster.	Any valid region name.
platform.azure.zone	List of availability zones to place machines in. For high availability, specify at least two zones.	List of zones, for example ["1", "2", "3"]



NOTE

You cannot customize [Azure Availability Zones](#) or [Use tags to organize your Azure resources](#) with an Azure cluster.

2.3.4.2. Sample customized `install-config.yaml` file for Azure

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:
  azure:
    osDisk:
      diskSizeGB: 512 5
      type: Standard_D8s_v3
    replicas: 3
compute: 6

```

```

- hyperthreading: Enabled 7
  name: worker
  platform:
    azure:
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 8
      zones: 9
        - "1"
        - "2"
        - "3"
  replicas: 5
  metadata:
    name: test-cluster 10
  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:
    azure:
      region: centralus 11
      baseDomainResourceGroupName: resource-group 12
    pullSecret: '{"auths": ...}' 13
    sshKey: ssh-ed25519 AAAA... 14

```

1 10 11 13 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 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 virtual machine types, such as **Standard_D8s_v3**, for your machines if you disable simultaneous multithreading.

5 8 You can specify the size of the disk to use in GB.

- 9 Specify a list of zones to deploy your machines to. For high availability, specify at least two zones.
- 12 Specify the name of the resource group that contains the DNS zone for your base domain.
- 14 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 on, specify an SSH key that your **ssh-agent** process uses.

2.3.5. Deploy the cluster

You can install OpenShift Container Platform on a compatible cloud.

**IMPORTANT**

You can run the **create cluster** command of 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=info ②
```

- 1 For **<installation_directory>**, specify the location of your customized **.install-config.yaml** file.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

**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**

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

**IMPORTANT**

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

2.3.6. Installing the CLI

You can install the CLI in order to interact with OpenShift Container Platform using a command-line interface.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.2. Download and install the new version of **oc**.

Procedure

1. From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**
2. Click the folder for your operating system and architecture and click the compressed file.
3. Save the file to your file system.
4. Extract the compressed file.
5. Place it in a directory that is on your **PATH**.

NOTE

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

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.3.7. Logging in to the cluster

You can log in to 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
```

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

2. Verify you can run **oc** commands successfully using the exported configuration:

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

Next steps

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

2.4. INSTALLING A CLUSTER ON AZURE WITH NETWORK CUSTOMIZATIONS

In OpenShift Container Platform version 4.2, you can install a cluster with a customized network configuration on infrastructure that the installation program provisions on Microsoft Azure. 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 Azure account](#) to host the cluster and determine the tested and validated region to deploy the cluster to.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.

2.4.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.2, you require access to the internet to install and entitle your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager](#). From there, you can allocate entitlements to your cluster.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and

perform subscription management and entitlement. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster. If the Telemetry service cannot entitle your cluster, you must manually entitle it on the [Cluster registration](#) page.

- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

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

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.

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>)

- 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 installation program.

2.4.3. 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 500 MB of local disk space to download the installation program.

Procedure

- 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.
- 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.

- 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
```

- 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.

2.4.4. Creating the installation configuration file

You can customize your installation of OpenShift Container Platform on

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 on, specify an SSH key that your **ssh-agent** process uses.

- ii. Enter a descriptive name for your cluster.
 - iii. 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.
3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



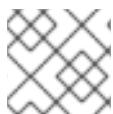
IMPORTANT

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

2.4.4.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster 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 in the **install-config.yaml** file after installation.

Table 2.4. Required parameters

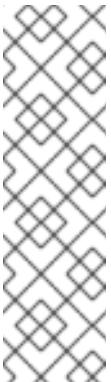
Parameter	Description	Values
baseDomain	The base domain of your cloud provider. This value is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <code><metadata.name>. <baseDomain></code> 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, azure, gcp, openstack , or {}
compute.platform	The cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack , or {}
metadata.name	The name of your cluster.	A string that contains uppercase or lowercase letters, such as dev .
platform.<platform>.region	The region to deploy your cluster in.	A valid region for your cloud, such as us-east-1 for AWS, centralus for Azure, or region1 for Red Hat OpenStack Platform (RHOSP).

Parameter	Description	Values
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 2.5. Optional parameters

Parameter	Description	Values
sshKey	The SSH key to use to access your cluster machines.	A valid, local public SSH key that you added to the ssh-agent process.

NOTE



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

Parameter	Description	Values
compute.hyperthreading	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.	Enabled or Disabled
	<p> IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p>	
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.hyperthreading	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.	Enabled or Disabled
	<p> IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p>	
controlPlane.replicas	The number of control plane machines to provision.	A positive integer greater than or equal to 3 . The default value is 3 .



IMPORTANT

The Open Virtual Networking (OVN) Kubernetes network plug-in is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of the OVN Technology Preview, see <https://access.redhat.com/articles/4380121>.

2.4.4.2. 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 in the **install-config.yaml** file after installation.

Table 2.6. Required network parameters

Parameter	Description	Value
networking.networkType	The network plug-in to deploy. The OpenShiftSDN plug-in is the only plug-in supported in OpenShift Container Platform 4.2. The OVNKubernetes plug-in is available as Technology Preview in OpenShift Container Platform 4.2.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork.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.clusterNetwork.hostPrefix	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.serviceNetwork	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.machineCIDR	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 .

2.4.4.3. Sample customized install-config.yaml file for Azure

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:
    azure:
      osDisk:
        diskSizeGB: 512 5
        type: Standard_D8s_v3
      replicas: 3
  compute: 6
    - hyperthreading: Enabled 7
    name: worker
    platform:
      azure:
        type: Standard_D2s_v3
        osDisk:
          diskSizeGB: 512 8
        zones: 9
          - "1"
          - "2"
          - "3"
      replicas: 5
  metadata:
    name: test-cluster 10
  networking: 11
    clusterNetwork:
      - cidr: 10.128.0.0/14
        hostPrefix: 23
    machineCIDR: 10.0.0.0/16
    networkType: OpenShiftSDN
    serviceNetwork:
      - 172.30.0.0/16
  platform:
    azure:
      region: centralus 12
      baseDomainResourceGroupName: resource-group 13
    pullSecret: '{"auths": ...}' 14
    sshKey: ssh-ed25519 AAAA... 15
```

1 **10** **12** **14** Required. The installation program prompts you for this value.

2 6 11 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 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 virtual machine types, such as **Standard_D8s_v3**, for your machines if you disable simultaneous multithreading.

5 8 You can specify the size of the disk to use in GB.

9 Specify a list of zones to deploy your machines to. For high availability, specify at least two zones.

13 Specify the name of the resource group that contains the DNS zone for your base domain.

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

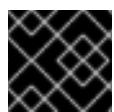


NOTE

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

2.4.5. 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, several network configuration files are in the **manifests**/ directory, as shown:

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

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** parameter are only an example. Specify your configuration for the Cluster Network Operator in the CR.

The CNO provides default values for the parameters in the CR, so you must specify only the parameters that you want to change.

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.

2.4.6. 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 Operator manages the cluster network.

You can specify the cluster network configuration for your OpenShift Container Platform cluster by setting the parameters for the **defaultNetwork** parameter in the CNO CR. 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: ①
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  serviceNetwork: ②
    - 172.30.0.0/16
  defaultNetwork: ③
...
  ...
  kubeProxyConfig: ④
  iptablesSyncPeriod: 30s ⑤
  proxyArguments:
    iptables-min-sync-period: ⑥
    - 30s
```

① ② Specified in the **install-config.yaml** file.

③ Configures the software-defined networking (SDN) for the cluster network.

④ 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.

⑤ 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.

⑥ 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](#)

2.4.6.1. Configuration parameters for OpenShift SDN

The following YAML object describes the configuration parameters for OpenShift SDN:

```
defaultNetwork:
  type: OpenShiftSDN ①
  openshiftSDNConfig: ②
  mode: NetworkPolicy ③
  mtu: 1450 ④
  vxlanPort: 4789 ⑤
```

- 1 Specified in the **install-config.yaml** file.
- 2 Specify only if you want to override part of the OpenShift SDN configuration.
- 3 Configures the network isolation mode for **OpenShiftSDN**. The allowed values are **Multitenant**, **Subnet**, or **NetworkPolicy**. The default value is **NetworkPolicy**.
- 4 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.
- 5 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**.

2.4.6.2. Configuration parameters for Open Virtual Network (OVN) SDN

The OVN SDN does not have any configuration parameters in OpenShift Container Platform 4.2.

2.4.6.3. Cluster Network Operator example CR

A complete CR for the CNO is displayed in the following example:

Cluster Network Operator example CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  serviceNetwork:
    - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
  kubeProxyConfig:
    iptablesSyncPeriod: 30s
    proxyArguments:
      iptables-min-sync-period:
        - 30s
```

2.4.7. Deploy the cluster

You can install OpenShift Container Platform on a compatible cloud.



IMPORTANT

You can run the **create cluster** command of 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=info ②
```

- ① For **<installation_directory>**, specify the
- ② To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



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

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



IMPORTANT

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

2.4.8. Installing the CLI

You can install the CLI in order to interact with OpenShift Container Platform using a command-line interface.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.2. Download and install the new version of **oc**.

Procedure

1. From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**
2. Click the folder for your operating system and architecture and click the compressed file.



NOTE

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

3. Save the file to your file system.
4. Extract the compressed file.
5. Place it in a directory that is on your **PATH**.

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.4.9. Logging in to the cluster

You can log in to 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
```

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

2. Verify you can run **oc** commands successfully using the exported configuration:

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

Next steps

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

2.5. UNINSTALLING A CLUSTER ON AZURE

You can remove a cluster that you deployed to Microsoft Azure.

2.5.1. Removing a cluster that uses installer-provisioned infrastructure

You can remove a cluster that uses installer-provisioned infrastructure from your cloud.

Prerequisites

- Have a copy of the installation program that you used to deploy the cluster.
- Have the files that the installation program generated when you created your cluster.

Procedure

1. 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.

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

CHAPTER 3. INSTALLING ON GCP

3.1. CONFIGURING A GCP PROJECT

Before you can install OpenShift Container Platform, you must configure a Google Cloud Platform (GCP) project to host it.

3.1.1. Creating a GCP project

To install OpenShift Container Platform, you must create a project in your Google Cloud Platform (GCP) account to host the cluster.

Procedure

- Create a project to host your OpenShift Container Platform cluster. See [Creating and Managing Projects](#) in the GCP documentation.

3.1.2. Enabling API services in GCP

Your Google Cloud Platform (GCP) project requires access to several API services to complete OpenShift Container Platform installation.

Prerequisites

- You created a project to host your cluster.

Procedure

- Enable the following required API services in the project that hosts your cluster. See [Enabling services](#) in the GCP documentation.

Table 3.1. Required API services

API service	Console service name
Compute Engine API	compute.googleapis.com
Google Cloud APIs	clouddapis.googleapis.com
Cloud Resource Manager API	cloudresourcemanager.googleapis.com
Google DNS API	dns.googleapis.com
IAM Service Account Credentials API	iamcredentials.googleapis.com
Identity and Access Management (IAM) API	iam.googleapis.com
Service Management API	servicemanagement.googleapis.com

API service	Console service name
Service Usage API	serviceusage.googleapis.com
Google Cloud Storage JSON API	storage-api.googleapis.com
Cloud Storage	storage-component.googleapis.com

3.1.3. Configuring DNS for GCP

To install OpenShift Container Platform, the Google Cloud Platform (GCP) account you use must have a dedicated public hosted zone in the same project that you host the OpenShift Container Platform cluster. This zone must be authoritative for the domain. The DNS 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 GCP or another source.



NOTE

If you purchase a new domain, it can take time for the relevant DNS changes to propagate. For more information about purchasing domains through Google, see [Google Domains](#).

2. Create a public hosted zone for your domain or subdomain in your GCP project. See [Creating public zones](#) in the GCP documentation.
Use an appropriate root domain, such as **openshiftcorp.com**, or subdomain, such as **clusters.openshiftcorp.com**.
3. Extract the new authoritative name servers from the hosted zone records. See [Look up your Cloud DNS name servers](#) in the GCP documentation.
You typically have four name servers.
4. Update the registrar records for the name servers that your domain uses. For example, if you registered your domain to Google Domains, see the following topic in the Google Domains Help: [How to switch to custom name servers](#).
5. If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain.

3.1.4. GCP account limits

The OpenShift Container Platform cluster uses a number of Google Cloud Platform (GCP) components, but the default [Quotas](#) do not affect your ability to install a default OpenShift Container Platform cluster.

A default cluster, which contains three compute and three control plane machines, uses the following resources. Note that some resources are required only during the bootstrap process and are removed after the cluster deploys.

Table 3.2. GCP resources used in a default cluster

Service	Component	Location	Total resources required	Resources removed after bootstrap
Service account	IAM	Global	5	0
Firewall Rules	Compute	Global	35	1
Forwarding Rules	Compute	Global	3	0
In-use IP addresses global	Compute	Global	4	1
Health checks	Compute	Global	3	0
Images	Compute	Global	1	0
Networks	Compute	Global	1	0
Static IP addresses	Compute	Region	4	1
Routers	Compute	Global	1	0
Routes	Compute	Global	3	0
Subnetworks	Compute	Global	2	0
Target Pools	Compute	Global	3	0
CPUs	Compute	Region	28	4
Persistent Disk SSD (GB)	Compute	Region	896	128

3.1.5. Creating a service account in GCP

OpenShift Container Platform requires a Google Cloud Platform (GCP) service account.

Prerequisites

- You created a project to host your cluster.

Procedure

1. Create a new service account in the project that you use to host your OpenShift Container Platform cluster. See [Creating a service account](#) in the GCP documentation.

2. Grant the service account the appropriate permissions. You can either grant the individual permissions that follow or assign the **Owner** role to it. See [Granting roles to a service account for specific resources](#).
3. Create the service account key. See [Creating service account keys](#) in the GCP documentation. The service account key is required to create a cluster.

3.1.5.1. Required GCP permissions

When you attach the **Owner** role to the service account that you create, you grant that service account all permissions, including those that are required to install OpenShift Container Platform. To deploy an OpenShift Container Platform cluster, the service account requires the following permissions:

Required roles for the installation program

- Compute Admin
- DNS Administrator
- Security Admin
- Service Account Admin
- Service Account User
- Storage Admin

Optional roles

For the cluster to create new limited credentials for its Operators, add the following role:

- Service Account Key Admin

The roles are applied to the service accounts that the control plane and compute machines use:

Table 3.3. GCP service account permissions

Account	Roles
Control Plane	roles/compute.instanceAdmin
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
Compute	roles/compute.viewer
	roles/storage.admin

3.1.6. Supported GCP regions

You can deploy an OpenShift Container Platform cluster to the following Google Cloud Platform (GCP) regions:

- asia-east1 (Changhua County, Taiwan)
- asia-east2 (Hong Kong)
- asia-northeast1 (Tokyo, Japan)
- asia-northeast2 (Osaka, Japan)
- asia-south1 (Mumbai, India)
- asia-southeast1 (Jurong West, Singapore)
- australia-southeast1 (Sydney, Australia)
- europe-north1 (Hamina, Finland)
- europe-west1 (St. Ghislain, Belgium)
- europe-west2 (London, England, UK)
- europe-west3 (Frankfurt, Germany)
- europe-west4 (Eemshaven, Netherlands)
- europe-west6 (Zürich, Switzerland)
- northamerica-northeast1 (Montréal, Québec, Canada)
- southamerica-east1 (São Paulo, Brazil)
- us-central1 (Council Bluffs, Iowa, USA)
- us-east1 (Moncks Corner, South Carolina, USA)
- us-east4 (Ashburn, Northern Virginia, USA)
- us-west1 (The Dalles, Oregon, USA)
- us-west2 (Los Angeles, California, USA)

Next steps

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

3.2. INSTALLING A CLUSTER QUICKLY ON GCP

In OpenShift Container Platform version 4.2, you can install a cluster on Google Cloud Platform (GCP) that uses the default configuration options.

Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure a GCP account](#) to host the cluster.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.

3.2.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.2, you require access to the internet to install and entitle your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager](#). From there, you can allocate entitlements to your cluster.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management and entitlement. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster. If the Telemetry service cannot entitle your cluster, you must manually entitle it on the [Cluster registration](#) page.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

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

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.

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 installation program.

3.2.3. 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 500 MB of local disk space to download the installation program.

Procedure

- 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.
- 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.

- 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.

3.2.4. Deploy the cluster

You can install OpenShift Container Platform on a compatible cloud.



IMPORTANT

You can run the **create cluster** command of 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. Remove any existing GCP credentials that do not use the service account key for the GCP account that you configured for your cluster and that are stored in the following locations:
 - The **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON**, or **GCLOUD_KEYFILE_JSON** environment variables
 - The **~/.gcp/osServiceAccount.json** file
 - The **gcloud cli** default credentials
2. Run the installation program:

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

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



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 on, specify an SSH key that your **ssh-agent** process uses.

- b. Select **gcp** as the platform to target.
- c. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.
- d. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
- e. Select the region to deploy the cluster to.
- f. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
- g. Enter a descriptive name for your cluster. If you provide a name that is longer than 6 characters, only the first 6 characters will be used in the infrastructure ID that is generated from the cluster name.
- h. 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

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



IMPORTANT

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

3.2.5. Installing the CLI

You can install the CLI in order to interact with OpenShift Container Platform using a command-line interface.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.2. Download and install the new version of **oc**.

Procedure

1. From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**
2. Click the folder for your operating system and architecture and click the compressed file.



NOTE

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

3. Save the file to your file system.
4. Extract the compressed file.
5. Place it in a directory that is on your **PATH**.

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

3.2.6. Logging in to the cluster

You can log in to 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
```

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

2. Verify you can run **oc** commands successfully using the exported configuration:

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

Next steps

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

3.3. INSTALLING A CLUSTER ON GCP WITH CUSTOMIZATIONS

In OpenShift Container Platform version 4.2, you can install a customized cluster on infrastructure that the installation program provisions on Google Cloud Platform (GCP). To customize the installation, you modify 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 a GCP account](#) to host the cluster.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.

3.3.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.2, you require access to the internet to install and entitle your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager](#). From there, you can allocate entitlements to your cluster.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management and entitlement. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster. If the Telemetry service cannot entitle your cluster, you must manually entitle it on the [Cluster registration](#) page.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

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

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.

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 installation program.

3.3.3. 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 500 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.

3.3.4. Creating the installation configuration file

You can customize your installation of OpenShift Container Platform on Google Cloud Platform (GCP).

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 on, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **gcp** as the platform to target.
 - iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.
 - iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
 - v. Select the region to deploy the cluster to.
 - vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
 - vii. Enter a descriptive name for your cluster. If you provide a name that is longer than 6 characters, only the first 6 characters will be used in the infrastructure ID that is generated from the cluster name.
 - viii. 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.
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.

3.3.4.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster 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 in the **install-config.yaml** file after installation.

Table 3.4. 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 <code><metadata.name>. <baseDomain></code> 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, azure, gcp, openstack , or {}
compute.platform	The cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack , or {}
metadata.name	The name of your cluster.	A string that contains uppercase or lowercase letters, such as dev .
platform. <platform>.region	The region to deploy your cluster in.	A valid region for your cloud, such as us-east-1 for AWS, centralus for Azure, or region1 for Red Hat OpenStack Platform (RHOSP).

Parameter	Description	Values
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 3.5. Optional parameters

Parameter	Description	Values
sshKey	The SSH key to use to access your cluster machines.	A valid, local public SSH key that you added to the ssh-agent process.

NOTE



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

Parameter	Description	Values
compute.hyperthreading	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.	Enabled or Disabled
	<p> IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p>	
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.hyperthreading	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.	Enabled or Disabled
	<p> IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p>	
controlPlane.replicas	The number of control plane machines to provision.	A positive integer greater than or equal to 3 . The default value is 3 .

3.3.4.2. Sample customized `install-config.yaml` file for GCP

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:
    gcp:
      rootVolume:
        iops: 4000
        size: 500
        type: io1
        type: n1-standard-4
      replicas: 3
  compute: 5
    - hyperthreading: Enabled 6
      name: worker
      platform:
        gcp:
          rootVolume:
            iops: 2000
            size: 500
            type: io1 7
            type: n1-standard-4
          replicas: 3
  metadata:
    name: test-cluster 8
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:
  gcp:
    ProjectID: openshift-production 9
    region: us-central-1 10
    pullSecret: '{"auths": ...}' 11
    sshKey: ssh-ed25519 AAAA... 12
```

1 **8** **9** **10** **11** Required. The installation program prompts you for this value.

2 5 If you do not provide these parameters and values, the installation program provides the default value.

3 6 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 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 machine types, such as **n1-standard-8**, for your machines if you disable simultaneous multithreading.

7 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 on, specify an SSH key that your **ssh-agent** process uses.

3.3.5. Deploy the cluster

You can install OpenShift Container Platform on a compatible cloud.



IMPORTANT

You can run the **create cluster** command of 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

- Remove any existing GCP credentials that do not use the service account key for the GCP account that you configured for your cluster and that are stored in the following locations:

- The **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON**, or **GCLLOUD_KEYFILE_JSON** environment variables
- The **~/.gcp/osServiceAccount.json** file
- The **gcloud cli** default credentials

2. Run the installation program:

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

- ① For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.
- ② To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



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

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



IMPORTANT

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

3.3.6. Installing the CLI

You can install the CLI in order to interact with OpenShift Container Platform using a command-line interface.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.2. Download and install the new version of **oc**.

Procedure

1. From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**
2. Click the folder for your operating system and architecture and click the compressed file.

**NOTE**

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

3. Save the file to your file system.
4. Extract the compressed file.
5. Place it in a directory that is on your **PATH**.

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

3.3.7. Logging in to the cluster

You can log in to 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
```

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

2. Verify you can run **oc** commands successfully using the exported configuration:

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

Next steps

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

3.4. UNINSTALLING A CLUSTER ON GCP

You can remove a cluster that you deployed to Google Cloud Platform (GCP).

3.4.1. Removing a cluster that uses installer-provisioned infrastructure

You can remove a cluster that uses installer-provisioned infrastructure from your cloud.

Prerequisites

- Have a copy of the installation program that you used to deploy the cluster.
- Have the files that the installation program generated when you created your cluster.

Procedure

1. 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.

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

CHAPTER 4. INSTALLING ON USER-PROVISIONED AWS

4.1. INSTALLING A CLUSTER ON USER-PROVISIONED INFRASTRUCTURE IN AWS BY USING CLOUDFORMATION TEMPLATES

In OpenShift Container Platform version 4.2, you can install a cluster on Amazon Web Services (AWS) by 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 allow the sites](#) that your cluster requires access to.



NOTE

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

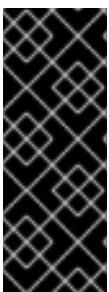
4.1.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.2, you require access to the internet to install and entitle your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager](#). From there, you can allocate entitlements to your cluster.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management and entitlement. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster. If the Telemetry service cannot entitle your cluster, you must manually entitle it on the [Cluster registration](#) page.

- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

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

4.1.2. 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.

4.1.2.1. 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 with the provided CloudFormation templates.



IMPORTANT

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

Table 4.1. Instance types for machines

Instance type	Bootstrap	Control plane	Compute
i3.large	x		
m4.large or m5.large			x

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

You might be able to use other instance types that meet the specifications of these instance types.

4.1.2.2. Other infrastructure components

- A VPC
- DNS entries
- Load balancers and listeners
- A public and a private 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 uses an endpoint that references the route tables for each subnet to improve communication with the registry that is hosted in S3.						
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 • AWS::EC2::SubnetRouteTableAssociation • AWS::EC2::NatGateway • AWS::EC2::EIP 	You must have a public internet gateway, with public routes, attached to the VPC. In the provided templates, each public subnet has a NAT gateway with an EIP address. These NAT gateways allow cluster resources, like private-subnet instances, to reach the internet and are not required for some restricted network or proxy scenarios.						
Network access control	<ul style="list-style-type: none"> • AWS::EC2::NetworkAcl • AWS::EC2::NetworkAclEntry 	<p>You must allow the VPC to access the following ports:</p> <table border="1"> <thead> <tr> <th>Port</th><th>Reason</th></tr> </thead> <tbody> <tr> <td>80</td><td>Inbound HTTP traffic</td></tr> <tr> <td>443</td><td>Inbound HTTPS traffic</td></tr> </tbody> </table>	Port	Reason	80	Inbound HTTP traffic	443	Inbound HTTPS traffic
Port	Reason							
80	Inbound HTTP traffic							
443	Inbound HTTPS traffic							

Component	AWS type	Description	
		22	Inbound SSH traffic
		1024 - 65535	Inbound ephemeral traffic
		0 - 65535	Outbound ephemeral traffic
Private subnets	<ul style="list-style-type: none"> ● AWS::EC2::Subnet ● AWS::EC2::RouteTable ● AWS::EC2::SubnetRouteTableAssociation 	<p>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.</p>	

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.

Component	AWS type	Description
External API server record	AWS::Route 53::RecordSetGroup	Alias records for the external API server.
External listener	AWS::Elastic LoadBalancingV2::Listener	A listener on port 6443 for the external load balancer.
External target group	AWS::Elastic LoadBalancingV2::Target Group	The target group for the external load balancer.
Private load balancer	AWS::Elastic LoadBalancingV2::LoadBalancer	The load balancer for your private subnets.
Internal API server record	AWS::Route 53::RecordSetGroup	Alias records for the internal API server.
Internal listener	AWS::Elastic LoadBalancingV2::Listener	A listener on port 22623 for the internal load balancer.
Internal target group	AWS::Elastic LoadBalancingV2::Target Group	The target group for the Internal load balancer.
Internal listener	AWS::Elastic LoadBalancingV2::Listener	A listener on port 6443 for the internal load balancer.
Internal target group	AWS::Elastic LoadBalancingV2::Target Group	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::Security Group	icmp	0
		tcp	22
		tcp	6443
		tcp	22623
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

Ingress group	Description	IP protocol	Port range
MasterIngress IngressServices	Kubernetes Ingress services	tcp	30000 - 32767
MasterIngress WorkerIngress Services	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 WorkerIngress Services	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	*
Worker	Allow	ec2:Describe*	*
Bootstrap	Allow	ec2:Describe*	*
	Allow	ec2:AttachVolume	*
	Allow	ec2:DetachVolume	*

4.1.2.3. 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:ModifyInstanceStateAttribute**
- **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**

4.1.3. 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 500 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.

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.

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 installation program. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

4.1.5. 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.

4.1.5.1. Creating the installation configuration file

Generate and customize the installation configuration file that the installation program needs to deploy your cluster.

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 on, specify an SSH key that your **ssh-agent** process uses.

- 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 installation process. If you want to reuse the file, you must back it up now.

4.1.5.2. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the Proxy object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The Proxy object's **status.noProxy** field is populated by default with the instance metadata endpoint (**169.254.169.254**) and with the values of the **networking.machineCIDR**, **networking.clusterNetwork.cidr**, and **networking.serviceNetwork** fields from your installation configuration.

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
  additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
  ...
  ...
```

- 1** A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.
- 2** A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. The URL scheme must be **http; https**; **https** is currently not supported.

- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with . to include all subdomains of
- 4 If provided, the installation program generates a ConfigMap that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** ConfigMap that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this ConfigMap is referenced in the Proxy object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.



NOTE

The installation program does not support the proxy **readinessEndpoints** field.

- 2 Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster** Proxy object is still created, but it will have a nil **spec**.



NOTE

Only the Proxy object named **cluster** is supported, and no additional proxies can be created.

4.1.5.3. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and 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.
- Create the **install-config.yaml** installation configuration file.

Procedure

- 1 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 installation directory that contains the **install-config.yaml** file you created.

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

- 2 Remove the Kubernetes manifest files that define the control plane machines:

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

By removing these files, you prevent the cluster from automatically generating control plane machines.

- 3 Remove the Kubernetes manifest files that define the worker machines:

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

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

- 4 Modify the **manifests/cluster-scheduler-02-config.yml** Kubernetes manifest file to prevent Pods from being scheduled on the control plane machines:

- a. Open the **manifests/cluster-scheduler-02-config.yml** file.
- b. Locate the **mastersSchedulable** parameter and set its value to **False**.
- c. Save and exit the file.



NOTE

Currently, due to a [Kubernetes limitation](#), router Pods running on control plane machines will not be reachable by the ingress load balancer. This step might not be required in a future minor version of OpenShift Container Platform.

- 5 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
  
```

4.1.6. 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). The provided CloudFormation templates contain references to this infrastructure name, 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 and view 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.

4.1.7. 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", ①
    "ParameterValue": "10.0.0.0/16" ②
},
{
    "ParameterKey": "AvailabilityZoneCount", ③
    "ParameterValue": "1" ④
},
{
    "ParameterKey": "SubnetBits", ⑤
    "ParameterValue": "12" ⑥
}
]

```

- ① The CIDR block for the VPC.
- ② Specify a CIDR block in the format **x.x.x.x/16-24**.
- ③ The number of availability zones to deploy the VPC in.
- ④ Specify an integer between **1** and **3**.
- ⑤ The size of each subnet in each availability zone.
- ⑥ 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> ①
--template-body file://<template>.yaml ②
--parameters file://<parameters>.json ③

```

- ① **<name>** is the name for the CloudFormation stack, such as **cluster-vpc**. 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:

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

4.1.7.1. 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 [&gt;!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 [&gt;!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 [&gt;!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 [&gt;!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
InternetGateway:
  Type: "AWS::EC2::InternetGateway"
GatewayToInternet:
  Type: "AWS::EC2::VPCGatewayAttachment"
  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
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"]]
  ]

```

4.1.8. 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

- Obtain the Hosted Zone ID for 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'
```

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

- 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  
 }]
```

- 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.
- 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.
ApiServerDnsName	Full host name of the API server.
RegisterNlbIpTargetsLambda	Lambda ARN useful to help register/deregister IP targets for these load balancers.
ExternalApiTargetGroupArn	ARN of external API target group.
InternalApiTargetGroupArn	ARN of internal API target group.
InternalServiceTargetGroupArn	ARN of internal service target group.

4.1.8.1. 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 ["-", [&gt;Ref InfrastructureName, "ext"]]
    IpAddressType: ipv4
    Subnets: !Ref PublicSubnets
    Type: network

```

```

IntApiElb:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Name: !Join ["-", [&gt;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 [".", [&gt;Ref ClusterName, !Ref HostedZoneName]]
    HostedZoneTags:
      - Key: Name
        Value: !Join ["-", [&gt;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

RegisterTargetLambdaRole:

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:
- "RegisterTargetLambdalarnRole"
- "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']}])

```

```

event['ResourceProperties']['TargetIp']}}])
    responseData = {}
    cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['TargetArn']+event['ResourceProperties']['TargetIp'])
    Runtime: "python3.7"
    Timeout: 120

```

RegisterSubnetTagsLambdalamRole:

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 = {}
    cfresponse.send(event, context, cfresponse.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]]

RegisterNlbIpTargetsLambda:

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

Value: !GetAtt RegisterNlbIpTargets.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

4.1.9. 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

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

```
[  
  {  
    "ParameterKey": "InfrastructureName", ①  
    "ParameterValue": "mycluster-<random_string>" ②  
  },  
  {  
    "ParameterKey": "VpcCidr", ③  
    "ParameterValue": "10.0.0.0/16" ④  
  },  
  {  
    "ParameterKey": "PrivateSubnets", ⑤  
    "ParameterValue": "subnet-<random_string>" ⑥  
  },  
  {  
    "ParameterKey": "VpcId", ⑦  
    "ParameterValue": "vpc-<random_string>" ⑧  
  }  
]
```

- The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
- The CIDR block for the VPC.
- Specify the CIDR block parameter that you used for the VPC that you defined in the form **x.x.x.x/16-24**.
- The private subnets that you created for your VPC.

- 6 Specify the **PrivateSubnetIds** value from the output of the CloudFormation template for the VPC.
- 7 The VPC that you created for the cluster.
- 8 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> 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-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:

MasterSecurityGroupIId	Master Security Group ID
WorkerSecurityGroupIId	Worker Security Group ID
MasterInstanceProfile	Master IAM Instance Profile

WorkerInstanceProfile	Worker IAM Instance Profile
------------------------------	-----------------------------

4.1.9.1. 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])(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

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
IpProtocol: tcp

```

```

WorkerIngressWorkerIngressServices:
Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt WorkerSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
Description: Kubernetes ingress services
FromPort: 30000
ToPort: 32767
IpProtocol: 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

4.1.10. 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 4.2. RHCOS AMIs

AWS zone	AWS AMI
ap-northeast-1	ami-0426ca3481a088c7b
ap-northeast-2	ami-014514ae47679721b
ap-south-1	ami-0bd772ba746948d9a
ap-southeast-1	ami-0d76ac0ebaac29e40
ap-southeast-2	ami-0391e92574fb09e08
ca-central-1	ami-04419691da69850cf
eu-central-1	ami-092b69120ecf915ed
eu-west-1	ami-04370efd78434697b
eu-west-2	ami-00c74e593125e0096
eu-west-3	ami-058ad17da14ff4d0d
sa-east-1	ami-03f6b71e93e630dab
us-east-1	ami-01e7fdcb66157b224
us-east-2	ami-0bc59aaa7363b805d
us-west-1	ami-0ba912f53c1fdcdf0

AWS zone	AWS AMI
us-west-2	ami-08e10b201e19fd5e7

4.1.11. 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": "RhcossAmi", ③
    "ParameterValue": "ami-<random_string>" ④
  },
  {
    "ParameterKey": "AllowedBootstrapSshCidr", ⑤
    "ParameterValue": "0.0.0.0/0" ⑥
  },
  {
    "ParameterKey": "PublicSubnet", ⑦
    "ParameterValue": "subnet-<random_string>" ⑧
  },
  {
    "ParameterKey": "MasterSecurityGroupId", ⑨
    "ParameterValue": "sg-<random_string>" ⑩
  },
  {
    "ParameterKey": "VpcId", ⑪
    "ParameterValue": "vpc-<random_string>" ⑫
  },
  {
    "ParameterKey": "BootstrapIgnitionLocation", ⑬
    "ParameterValue": "s3://<bucket_name>/bootstrap.ign" ⑭
  },
  {
    "ParameterKey": "AutoRegisterELB", ⑮
    "ParameterValue": "yes" ⑯
  },
  {
    "ParameterKey": "RegisterNlbIpTargetsLambdaArn", ⑰
  }
]
```

```

    "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNLBIPTargets-<random_string>" 18
},
{
  "ParameterKey": "ExternalApiTargetGroupArn", 19
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 20
},
{
  "ParameterKey": "InternalApiTargetGroupArn", 21
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 22
},
{
  "ParameterKey": "InternalServiceTargetGroupArn", 23
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 24
}
]

```

- 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 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 **RegisterNLBIPTargetsLambda** 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 machine** 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> 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-bootstrap**. 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:

BootstrapInstanceld	The bootstrap Instance ID.
BootstrapPublicIp	The bootstrap node public IP address.
BootstrapPrivateIp	The bootstrap node private IP address.

4.1.11.1. 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])(/[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
- RegisterNLBIPTargetsLambdaArn
- 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 "MasterSecurityGroupId"
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 RegisterNlbIpTargetsLambdaArn
TargetArn: !Ref ExternalApiTargetGroupArn
TargetIp: !GetAtt BootstrapInstance.PrivateIp

RegisterBootstrapInternalApiTarget:
Condition: DoRegistration
Type: Custom::NLBRegister
Properties:
ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
TargetArn: !Ref InternalApiTargetGroupArn
TargetIp: !GetAtt BootstrapInstance.PrivateIp

RegisterBootstrapInternalServiceTarget:
Condition: DoRegistration
Type: Custom::NLBRegister
Properties:
ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
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

4.1.12. 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": "RhcossAmi", 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" ⑳
},
{
  "ParameterKey": "CertificateAuthorities", ㉑
  "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz==" ㉒
},
{
  "ParameterKey": "MasterInstanceProfileName", ㉓
  "ParameterValue": "<roles_stack>-MasterInstanceProfile-<random_string>" ㉔
},
{
  "ParameterKey": "MasterInstanceType", ㉕
  "ParameterValue": "m4.xlarge" ㉖
},
{
  "ParameterKey": "AutoRegisterELB", ㉗
  "ParameterValue": "yes" ㉘
},
{
  "ParameterKey": "RegisterNlbIpTargetsLambdaArn", ㉙
  "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:<dns_stack_name>-RegisterNlbIpTargets-<random_string>" ㉚
},
{
  "ParameterKey": "ExternalApiTargetGroupArn", ㉛
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" ㉜
}

```

```
{
  "ParameterKey": "InternalApiTargetGroupArn", 33
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:  
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 34
},
{
  "ParameterKey": "InternalServiceTargetGroupArn", 35
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:  
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 36
}
]
```

- 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** CurrentRed 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**](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 **RegisterNLbIpTargetsLambda** 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> ①
--template-body file://<template>.yaml ②
--parameters file://<parameters>.json ③
```

- 1 **<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.
- 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>
```

4.1.12.1. 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 Z21IXYZABCZ2A4.

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

RegisterNLBIPTargetsLambdaArn:

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:
 - MasterInstanceId
 - 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
- RegisterNLbIpTargetsLambdaArn
- 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"
RhcossAmi:
  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 RegisterNlbIpTargetsLambdaArn
TargetArn: !Ref ExternalApiTargetGroupArn
TargetIp: !GetAtt Master0.PrivateIp

```

```

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

```

```

RegisterMaster0InternalServiceTarget:
Condition: DoRegistration
Type: Custom::NLBRegister
Properties:
ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
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 RegisterNlbIpTargetsLambdaArn
  TargetArn: !Ref ExternalApiTargetGroupArn
  TargetIp: !GetAtt Master1.PrivateIp

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

RegisterMaster1InternalServiceTarget:
Condition: DoRegistration
Type: Custom::NLBRegister
Properties:
  ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
  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 "MasterSecurityGroupId"
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 RegisterNlbIpTargetsLambdaArn
TargetArn: !Ref ExternalApiTargetGroupArn
TargetIp: !GetAtt Master2.PrivateIp

RegisterMaster2InternalApiTarget:
Condition: DoRegistration
Type: Custom::NLBRegister
Properties:
ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
TargetArn: !Ref InternalApiTargetGroupArn
TargetIp: !GetAtt Master2.PrivateIp

RegisterMaster2InternalServiceTarget:
Condition: DoRegistration
Type: Custom::NLBRegister
Properties:
ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
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]
]

```

4.1.13. 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=info ②
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

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

4.1.13.1. 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": "RhcossAmi", ③
    "ParameterValue": "ami-<random_string>" ④
  },
  {
    "ParameterKey": "Subnet", ⑤
    "ParameterValue": "subnet-<random_string>" ⑥
  },
  {
    "ParameterKey": "WorkerSecurityGroupId", 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 **WorkerSecurityGroupId** 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.

4.1.13.1.1. 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.

AWS::TemplateFormatVersion: **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

4.1.14. Installing the CLI

You can install the CLI in order to interact with OpenShift Container Platform using a command-line interface.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.2. Download and install the new version of **oc**.

Procedure

- From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**.
- Click the folder for your operating system and architecture and click the compressed file.



NOTE

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

- Save the file to your file system.
- Extract the compressed file.

5. Place it in a directory that is on your **PATH**.

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

4.1.15. Logging in to the cluster

You can log in to 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
```

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

2. Verify you can run **oc** commands successfully using the exported configuration:

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

4.1.16. 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.
- Install the **jq** package.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
------	--------	-------	-----	---------

```
master-0 Ready master 63m v1.14.6+c4799753c
master-1 Ready master 63m v1.14.6+c4799753c
master-2 Ready master 64m v1.14.6+c4799753c
worker-0 NotReady worker 76s v1.14.6+c4799753c
worker-1 NotReady worker 70s v1.14.6+c4799753c
```

The output lists all of the machines that you created.

- Review the pending certificate signing requests (CSRs) and ensure that 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
```

4.1.17. 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

1. Watch the cluster components come online:

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED
SINCE				
authentication	4.2.0	True	False	False 69s
cloud-credential	4.2.0	True	False	False 12m
cluster-autoscaler	4.2.0	True	False	False 11m
console	4.2.0	True	False	False 46s
dns	4.2.0	True	False	False 11m
image-registry	4.2.0	False	True	False 5m26s
ingress	4.2.0	True	False	False 5m36s
kube-apiserver	4.2.0	True	False	False 8m53s
kube-controller-manager	4.2.0	True	False	False 7m24s
kube-scheduler	4.2.0	True	False	False 12m
machine-api	4.2.0	True	False	False 12m
machine-config	4.2.0	True	False	False 7m36s
marketplace	4.2.0	True	False	False 7m54m
monitoring	4.2.0	True	False	False 7h54s
network	4.2.0	True	False	False 5m9s
node-tuning	4.2.0	True	False	False 11m
openshift-apiserver	4.2.0	True	False	False 11m
openshift-controller-manager	4.2.0	True	False	False 5m943s
openshift-samples	4.2.0	True	False	False 3m55s
operator-lifecycle-manager	4.2.0	True	False	False 11m
operator-lifecycle-manager-catalog	4.2.0	True	False	False 11m
service-ca	4.2.0	True	False	False 11m
service-catalog-apiserver	4.2.0	True	False	False 5m26s
service-catalog-controller-manager	4.2.0	True	False	False 5m25s
storage	4.2.0	True	False	False 5m30s

2. Configure the Operators that are not available.

4.1.17.1. 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.

4.1.17.1.1. 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 an S3 bucket and configure storage with the following procedure.

Prerequisites

- A cluster on AWS with user-provisioned infrastructure.
- For S3 on AWS storage the secret is expected to contain two keys:
 - **REGISTRY_STORAGE_S3_ACCESSKEY**
 - **REGISTRY_STORAGE_S3_SECRETKEY**

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.

4.1.17.1.2. 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.

4.1.18. 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, and wait for installation to complete.

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.

**IMPORTANT**

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

Next steps

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

CHAPTER 5. INSTALLING ON USER-PROVISIONED GCP

5.1. INSTALLING A CLUSTER ON GCP USING DEPLOYMENT MANAGER TEMPLATES

In OpenShift Container Platform version 4.2, you can install a cluster on Google Cloud Platform (GCP) by using infrastructure that you provide.

The steps for performing a user-provided infrastructure install are outlined here. Several [Deployment Manager](#) templates are provided to assist in completing these steps or to help model your own. You are also free to create the required resources through other methods; the templates are just an example.

5.1.1. Configuring your GCP project

Before you can install OpenShift Container Platform, you must configure a Google Cloud Platform (GCP) project to host it.

5.1.1.1. Creating a GCP project

To install OpenShift Container Platform, you must create a project in your Google Cloud Platform (GCP) account to host the cluster.

Procedure

- Create a project to host your OpenShift Container Platform cluster. See [Creating and Managing Projects](#) in the GCP documentation.

5.1.1.2. Enabling API services in GCP

Your Google Cloud Platform (GCP) project requires access to several API services to complete OpenShift Container Platform installation.

Prerequisites

- You created a project to host your cluster.

Procedure

- Enable the following required API services in the project that hosts your cluster. See [Enabling services](#) in the GCP documentation.

Table 5.1. Required API services

API service	Console service name
Cloud Deployment Manager V2 API	deploymentmanager.googleapis.com
Compute Engine API	compute.googleapis.com
Google Cloud APIs	cloudapis.googleapis.com

API service	Console service name
Cloud Resource Manager API	cloudresourcemanager.googleapis.com
Google DNS API	dns.googleapis.com
IAM Service Account Credentials API	iamcredentials.googleapis.com
Identity and Access Management (IAM) API	iam.googleapis.com
Service Management API	servicemanagement.googleapis.com
Service Usage API	serviceusage.googleapis.com
Google Cloud Storage JSON API	storage-api.googleapis.com
Cloud Storage	storage-component.googleapis.com

5.1.1.3. Configuring DNS for GCP

To install OpenShift Container Platform, the Google Cloud Platform (GCP) account you use must have a dedicated public hosted zone in the same project that you host the OpenShift Container Platform cluster. This zone must be authoritative for the domain. The DNS 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 GCP or another source.



NOTE

If you purchase a new domain, it can take time for the relevant DNS changes to propagate. For more information about purchasing domains through Google, see [Google Domains](#).

2. Create a public hosted zone for your domain or subdomain in your GCP project. See [Creating public zones](#) in the GCP documentation.
Use an appropriate root domain, such as **openshiftcorp.com**, or subdomain, such as **clusters.openshiftcorp.com**.
3. Extract the new authoritative name servers from the hosted zone records. See [Look up your Cloud DNS name servers](#) in the GCP documentation.
You typically have four name servers.
4. Update the registrar records for the name servers that your domain uses. For example, if you registered your domain to Google Domains, see the following topic in the Google Domains Help: [How to switch to custom name servers](#).

- If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain.

5.1.1.4. GCP account limits

The OpenShift Container Platform cluster uses a number of Google Cloud Platform (GCP) components, but the default [Quotas](#) do not affect your ability to install a default OpenShift Container Platform cluster.

A default cluster, which contains three compute and three control plane machines, uses the following resources. Note that some resources are required only during the bootstrap process and are removed after the cluster deploys.

Table 5.2. GCP resources used in a default cluster

Service	Component	Location	Total resources required	Resources removed after bootstrap
Service account	IAM	Global	5	0
Firewall Rules	Networking	Global	35	1
Forwarding Rules	Compute	Global	2	0
Health checks	Compute	Global	2	0
Images	Compute	Global	1	0
Networks	Networking	Global	1	0
Routers	Networking	Global	1	0
Routes	Networking	Global	3	0
Subnetworks	Compute	Global	2	0
Target Pools	Networking	Global	2	0

5.1.1.5. Creating a service account in GCP

OpenShift Container Platform requires a Google Cloud Platform (GCP) service account.

Prerequisites

- You created a project to host your cluster.

Procedure

- Create a new service account in the project that you use to host your OpenShift Container Platform cluster. See [Creating a service account](#) in the GCP documentation.

2. Grant the service account the appropriate permissions. You can either grant the individual permissions that follow or assign the **Owner** role to it. See [Granting roles to a service account for specific resources](#).
3. Create the service account key. See [Creating service account keys](#) in the GCP documentation. The service account key is required to create a cluster.

5.1.1.5.1. Required GCP permissions

When you attach the **Owner** role to the service account that you create, you grant that service account all permissions, including those that are required to install OpenShift Container Platform. To deploy an OpenShift Container Platform cluster, the service account requires the following permissions:

Required roles for the installation program

- Compute Admin
- DNS Administrator
- Security Admin
- Service Account Admin
- Service Account User
- Storage Admin

Required roles for user-provisioned GCP infrastructure

- Deployment Manager Editor
- Service Account Key Admin

Optional roles

For the cluster to create new limited credentials for its Operators, add the following role:

- Service Account Key Admin

The roles are applied to the service accounts that the control plane and compute machines use:

Table 5.3. GCP service account permissions

Account	Roles
Control Plane	roles/compute.instanceAdmin
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser

Account	Roles
Compute	roles/compute.viewer
	roles/storage.admin

5.1.1.6. Supported GCP regions

You can deploy an OpenShift Container Platform cluster to the following Google Cloud Platform (GCP) regions:

- asia-east1 (Changhua County, Taiwan)
- asia-east2 (Hong Kong)
- asia-northeast1 (Tokyo, Japan)
- asia-northeast2 (Osaka, Japan)
- asia-south1 (Mumbai, India)
- asia-southeast1 (Jurong West, Singapore)
- australia-southeast1 (Sydney, Australia)
- europe-north1 (Hamina, Finland)
- europe-west1 (St. Ghislain, Belgium)
- europe-west2 (London, England, UK)
- europe-west3 (Frankfurt, Germany)
- europe-west4 (Eemshaven, Netherlands)
- europe-west6 (Zürich, Switzerland)
- northamerica-northeast1 (Montréal, Québec, Canada)
- southamerica-east1 (São Paulo, Brazil)
- us-central1 (Council Bluffs, Iowa, USA)
- us-east1 (Moncks Corner, South Carolina, USA)
- us-east4 (Ashburn, Northern Virginia, USA)
- us-west1 (The Dalles, Oregon, USA)
- us-west2 (Los Angeles, California, USA)

5.1.1.7. Installing and configuring CLI tools for GCP

To install OpenShift Container Platform on Google Cloud Platform (GCP) using user-provisioned infrastructure, you must install and configure the CLI tools for GCP.

Prerequisites

- You created a project to host your cluster.
- You created a service account and granted it the required permissions.

Procedure

1. Install the following binaries in **\$PATH**:

- **gcloud**
- **gsutil**

See [Install the latest Cloud SDK version](#) in the GCP documentation.

2. Authenticate using the **gcloud** tool with your configured service account.

5.1.2. Creating the installation files for GCP

To install OpenShift Container Platform on Google Cloud Platform (GCP) 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.

5.1.2.1. Creating the installation configuration file

You can customize your installation of OpenShift Container Platform on Google Cloud Platform (GCP).

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 on, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **gcp** as the platform to target.
 - iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.
 - iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
 - v. Select the region to deploy the cluster to.
 - vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
 - vii. Enter a descriptive name for your cluster. If you provide a name that is longer than 6 characters, only the first 6 characters will be used in the infrastructure ID that is generated from the cluster name.
 - viii. Paste the pull secret that you obtained from the [OpenShift Infrastructure Providers](#) page.
- c. Optional: If you do not want the cluster to provision compute machines, empty the compute pool by editing the resulting **install-config.yaml** file to set **replicas** to **0** for the **compute** pool:

```
compute:
- hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 0 1
```

1 Set to **0**.

2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
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.

5.1.2.2. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the Proxy object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The Proxy object's **status.noProxy** field is populated by default with the instance metadata endpoint (**169.254.169.254**) and with the values of the **networking.machineCIDR**, **networking.clusterNetwork.cidr**, and **networking.serviceNetwork** fields from your installation configuration.

Procedure

- 1 Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> ①
  httpsProxy: http://<username>:<pswd>@<ip>:<port> ②
  noProxy: example.com ③
  additionalTrustBundle: | ④
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
  ...
  ...
```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. The URL scheme must be **http**; **https** is currently not supported.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to include all subdomains of that domain. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a ConfigMap that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** ConfigMap that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this ConfigMap is referenced in the Proxy object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

**NOTE**

The installation program does not support the proxy **readinessEndpoints** field.

- Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster** Proxy object is still created, but it will have a nil **spec**.

**NOTE**

Only the Proxy object named **cluster** is supported, and no additional proxies can be created.

5.1.2.3. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and 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.
- Create the **install-config.yaml** installation configuration file.

Procedure

- 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 installation directory that contains the **install-config.yaml** file you created.

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

- Remove the Kubernetes manifest files that define the control plane machines:

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

By removing these files, you prevent the cluster from automatically generating control plane machines.

3. Optional: If you do not want the cluster to provision compute machines, remove the Kubernetes manifest files that define the worker machines:

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

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

4. Modify the **manifests/cluster-scheduler-02-config.yml** Kubernetes manifest file to prevent Pods from being scheduled on the control plane machines:
 - a. Open the **manifests/cluster-scheduler-02-config.yml** file.
 - b. Locate the **mastersSchedulable** parameter and set its value to **False**.
 - c. Save and exit the file.



NOTE

Currently, due to a [Kubernetes limitation](#), router Pods running on control plane machines will not be reachable by the ingress load balancer. This step might not be required in a future minor version of OpenShift Container Platform.

5. Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the **privateZone** and **publicZone** sections from the **manifests/cluster-dns-02-config.yml** DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: ①
    id: mycluster-100419-private-zone
  publicZone: ②
    id: example.openshift.com
status: {}
```

① ② Remove these sections completely.

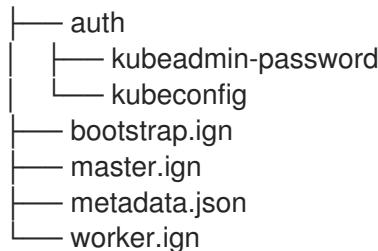
If you do so, you must add ingress DNS records manually in a later step.

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:



Additional resources

- [Optional: Adding the ingress DNS records](#)

5.1.3. Exporting common variables

5.1.3.1. Extracting the infrastructure name

The Ignition configs contain a unique cluster identifier that you can use to uniquely identify your cluster in Google Cloud Platform (GCP). The provided Deployment Manager templates contain references to this infrastructure name, 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

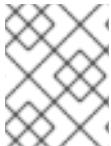
- To extract and view the infrastructure name from the Ignition config file metadata, run the following command:

```
$ 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.

5.1.3.2. Exporting common variables for Deployment Manager templates

You must export a common set of variables that are used with the provided Deployment Manager templates used to assist in completing a user-provided infrastructure install on Google Cloud Platform (GCP).

**NOTE**

Specific Deployment Manager templates can also require additional exported variables, which are detailed in their related procedures.

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

- Export the following common variables to be used by the provided Deployment Manager templates:

```
$ export BASE_DOMAIN='<base_domain>'  
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>'  
$ export NETWORK_CIDR='10.0.0.0/16'  
$ export MASTER_SUBNET_CIDR='10.0.0.0/19'  
$ export WORKER_SUBNET_CIDR='10.0.32.0/19'  
  
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig ①  
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`  
$ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`  
$ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`  
$ export REGION=`jq -r .gcp.region <installation_directory>/metadata.json`
```

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

5.1.4. Creating a VPC in GCP

You must create a VPC in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. You can customize the VPC to meet your requirements. One way to create the VPC is to modify the provided Deployment Manager template.

**NOTE**

If you do not use the provided Deployment Manager template to create your GCP 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 a GCP account.
- Generate the Ignition config files for your cluster.

Procedure

1. Copy the template from the **Deployment Manager template for the VPC** section of this topic and save it as **01_vpc.py** on your computer. This template describes the VPC that your cluster requires.
2. Create a **01_vpc.yaml** resource definition file:

```
$ cat <<EOF >01_vpc.yaml
imports:
- path: 01_vpc.py

resources:
- name: cluster-vpc
  type: 01_vpc.py
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2

    master_subnet_cidr: '${MASTER_SUBNET_CIDR}' 3
    worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' 4
EOF
```

- 1** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 2** **region** is the region to deploy the cluster into, for example **us-east1**.
- 3** **master_subnet_cidr** is the CIDR for the master subnet, for example **10.0.0.0/19**.
- 4** **worker_subnet_cidr** is the CIDR for the worker subnet, for example **10.0.32.0/19**.

3. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-vpc --config 01_vpc.yaml
```

5.1.4.1. Deployment Manager template for the VPC

You can use the following Deployment Manager template to deploy the VPC that you need for your OpenShift Container Platform cluster:

01_vpc.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [
        'name': context.properties['infra_id'] + '-network',
        'type': 'compute.v1.network',
        'properties': {
            'region': context.properties['region'],
            'autoCreateSubnetworks': False
        }
    ],
    [
        'name': context.properties['infra_id'] + '-master-subnet',
        'type': 'compute.v1.subnetwork',
```

```

'properties': {
    'region': context.properties['region'],
    'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
    'ipCidrRange': context.properties['master_subnet_cidr']
}
}, {
    'name': context.properties['infra_id'] + '-worker-subnet',
    'type': 'compute.v1.subnetwork',
    'properties': {
        'region': context.properties['region'],
        'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
        'ipCidrRange': context.properties['worker_subnet_cidr']
    }
}, {
    'name': context.properties['infra_id'] + '-master-nat-ip',
    'type': 'compute.v1.address',
    'properties': {
        'region': context.properties['region']
    }
}, {
    'name': context.properties['infra_id'] + '-worker-nat-ip',
    'type': 'compute.v1.address',
    'properties': {
        'region': context.properties['region']
    }
}, {
    'name': context.properties['infra_id'] + '-router',
    'type': 'compute.v1.router',
    'properties': {
        'region': context.properties['region'],
        'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
        'nats': [
            {
                'name': context.properties['infra_id'] + '-nat-master',
                'natIpAllocateOption': 'MANUAL_ONLY',
                'natIps': ['$ref.' + context.properties['infra_id'] + '-master-nat-ip.selfLink'],
                'minPortsPerVm': 7168,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [
                    {
                        'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
                        'sourceIpRangesToNat': ['ALL_IP_RANGES']
                    }
                ],
                'name': context.properties['infra_id'] + '-nat-worker',
                'natIpAllocateOption': 'MANUAL_ONLY',
                'natIps': ['$ref.' + context.properties['infra_id'] + '-worker-nat-ip.selfLink'],
                'minPortsPerVm': 128,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [
                    {
                        'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
                        'sourceIpRangesToNat': ['ALL_IP_RANGES']
                    }
                ]
            }
        ]
    }
},
return {'resources': resources}

```

5.1.5. Creating networking and load balancing components in GCP

You must configure networking and load balancing in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your GCP 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 a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.

Procedure

- Copy the template from the **Deployment Manager template for the network and load balancers** section of this topic and save it as **02_infra.py** on your computer. This template describes the networking and load balancing objects that your cluster requires.
- Export the following variable required by the resource definition:

```
$ export CLUSTER_NETWORK=`gcloud compute networks describe ${INFRA_ID}-network -format json | jq -r .selfLink`
```

- Create a **02_infra.yaml** resource definition file:

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_infra.py

resources:
- name: cluster-infra
  type: 02_infra.py
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2

    cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' 3
    cluster_network: '${CLUSTER_NETWORK}' 4
EOF
```

- 1** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 2** **region** is the region to deploy the cluster into, for example **us-east1**.
- 3** **cluster_domain** is the domain for the cluster, for example **openshift.example.com**.

- 4** **cluster_network** is the **selfLink** URL to the cluster network.

4. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

5. The templates do not create DNS entries due to limitations of Deployment Manager, so you must create them manually:

- a. Export the following variable:

```
$ export CLUSTER_IP=`gcloud compute addresses describe ${INFRA_ID}-cluster-public-ip --region=${REGION} --format json | jq -r .address`
```

- b. Add external DNS entries:

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
```

- c. Add internal DNS entries:

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

5.1.5.1. Deployment Manager template for the network and load balancers

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

02_infra.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [
        {
            'name': context.properties['infra_id'] + '-cluster-public-ip',
            'type': 'compute.v1.address',
            'properties': {
                'region': context.properties['region']
            }
        },
        {
            'name': context.properties['infra_id'] + '-api-http-health-check',
            'type': 'compute.v1.httpHealthCheck',
            'properties': {
                'port': 80
            }
        }
    ]
```

```

'properties': {
    'port': 6080,
    'requestPath': '/readyz'
}
}, {
    'name': context.properties['infra_id'] + '-api-target-pool',
    'type': 'compute.v1.targetPool',
    'properties': {
        'region': context.properties['region'],
        'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
        'instances': []
    }
}, {
    'name': context.properties['infra_id'] + '-api-forwarding-rule',
    'type': 'compute.v1.forwardingRule',
    'properties': {
        'region': context.properties['region'],
        'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink)',
        'target': '$(ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink)',
        'portRange': '6443'
    }
}, {
    'name': context.properties['infra_id'] + '-ign-http-health-check',
    'type': 'compute.v1.httpHealthCheck',
    'properties': {
        'port': 22624,
        'requestPath': '/healthz'
    }
}, {
    'name': context.properties['infra_id'] + '-ign-target-pool',
    'type': 'compute.v1.targetPool',
    'properties': {
        'region': context.properties['region'],
        'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-ign-http-health-check.selfLink)'],
        'instances': []
    }
}, {
    'name': context.properties['infra_id'] + '-ign-forwarding-rule',
    'type': 'compute.v1.forwardingRule',
    'properties': {
        'region': context.properties['region'],
        'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink)',
        'target': '$(ref.' + context.properties['infra_id'] + '-ign-target-pool.selfLink)',
        'portRange': '22623'
    }
}, {
    'name': context.properties['infra_id'] + '-private-zone',
    'type': 'dns.v1.managedZone',
    'properties': {
        'description': '',
        'dnsName': context.properties['cluster_domain'] + '',
        'visibility': 'private',
        'privateVisibilityConfig': {
            'networks': [
                'networkUrl': context.properties['cluster_network']
            ]
        }
    }
}

```

```

        }
    ]
}

return {'resources': resources}

```

5.1.6. Creating firewall rules and IAM roles in GCP

You must create security groups and roles in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your GCP 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 a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.

Procedure

- Copy the template from the **Deployment Manager template for firewall rules and IAM roles** section of this topic and save it as **03_security.py** on your computer. This template describes the security groups and roles that your cluster requires.
- Export the following variables required by the resource definition:

```

$ export MASTER_NAT_IP=`gcloud compute addresses describe ${INFRA_ID}-master-nat-ip --region ${REGION} --format json | jq -r .address`
$ export WORKER_NAT_IP=`gcloud compute addresses describe ${INFRA_ID}-worker-nat-ip --region ${REGION} --format json | jq -r .address`
```

- Create a **03_security.yaml** resource definition file:

```

$ cat <<EOF >03_security.yaml
imports:
- path: 03_security.py

resources:
- name: cluster-security
  type: 03_security.py
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2

  cluster_network: '${CLUSTER_NETWORK}' 3
```

```

network_cidr: '${NETWORK_CIDR}' 4
master_nat_ip: '${MASTER_NAT_IP}' 5
worker_nat_ip: '${WORKER_NAT_IP}' 6
EOF

```

- 1** `infra_id` is the **INFRA_ID** infrastructure name from the extraction step.
- 2** `region` is the region to deploy the cluster into, for example **us-east1**.
- 3** `cluster_network` is the **selfLink** URL to the cluster network.
- 4** `network_cidr` is the CIDR of the VPC network, for example **10.0.0.0/16**.
- 5** `master_nat_ip` is the IP address of the master NAT, for example **34.94.100.1**.
- 6** `worker_nat_ip` is the IP address of the worker NAT, for example **34.94.200.1**.

4. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-security --config
03_security.yaml
```

5. The templates do not create the policy bindings due to limitations of Deployment Manager, so you must create them manually:

```

$ export MASTER_SA=${INFRA_ID}-m@${PROJECT_NAME}.iam.gserviceaccount.com
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SA}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SA}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SA}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SA}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SA}" --role "roles/storage.admin"

$ export WORKER_SA=${INFRA_ID}-w@${PROJECT_NAME}.iam.gserviceaccount.com
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SA}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SA}" --role "roles/storage.admin"

```

6. Create a service account key and store it locally for later use:

```
$ gcloud iam service-accounts keys create service-account-key.json --iam-
account=${MASTER_SA}
```

5.1.6.1. Deployment Manager template for firewall rules and IAM roles

You can use the following Deployment Manager template to deploy the security objects that you need for your OpenShift Container Platform cluster:

03_security.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [
        {
            'name': context.properties['infra_id'] + '-api',
            'type': 'compute.v1.firewall',
            'properties': {
                'network': context.properties['cluster_network'],
                'allowed': [
                    {
                        'IPProtocol': 'tcp',
                        'ports': ['6443']
                    }
                ],
                'sourceRanges': ['0.0.0.0/0'],
                'targetTags': [context.properties['infra_id'] + '-master']
            }
        },
        {
            'name': context.properties['infra_id'] + '-mcs',
            'type': 'compute.v1.firewall',
            'properties': {
                'network': context.properties['cluster_network'],
                'allowed': [
                    {
                        'IPProtocol': 'tcp',
                        'ports': ['22623']
                    }
                ],
                'sourceRanges': [
                    context.properties['network_cidr'],
                    context.properties['master_nat_ip'],
                    context.properties['worker_nat_ip']
                ],
                'targetTags': [context.properties['infra_id'] + '-master']
            }
        },
        {
            'name': context.properties['infra_id'] + '-health-checks',
            'type': 'compute.v1.firewall',
            'properties': {
                'network': context.properties['cluster_network'],
                'allowed': [
                    {
                        'IPProtocol': 'tcp',
                        'ports': ['6080', '22624']
                    }
                ],
                'sourceRanges': ['35.191.0.0/16', '209.85.152.0/22', '209.85.204.0/22'],
                'targetTags': [context.properties['infra_id'] + '-master']
            }
        },
        {
            'name': context.properties['infra_id'] + '-etcd',
            'type': 'compute.v1.firewall',
            'properties': {
                'network': context.properties['cluster_network'],
                'allowed': [
                    {
                        'IPProtocol': 'tcp',
                        'ports': ['2379-2380']
                    }
                ],
                'sourceTags': [context.properties['infra_id'] + '-master'],
                'targetTags': [context.properties['infra_id'] + '-master']
            }
        }
    ]

```

```

}, {
  'name': context.properties['infra_id'] + '-control-plane',
  'type': 'compute.v1.firewall',
  'properties': {
    'network': context.properties['cluster_network'],
    'allowed': [
      {
        'IPProtocol': 'tcp',
        'ports': ['10257']
      },
      {
        'IPProtocol': 'tcp',
        'ports': ['10259']
      }
    ],
    'sourceTags': [
      context.properties['infra_id'] + '-master',
      context.properties['infra_id'] + '-worker'
    ],
    'targetTags': [context.properties['infra_id'] + '-master']
  }
}, {
  'name': context.properties['infra_id'] + '-internal-network',
  'type': 'compute.v1.firewall',
  'properties': {
    'network': context.properties['cluster_network'],
    'allowed': [
      {
        'IPProtocol': 'icmp'
      },
      {
        'IPProtocol': 'tcp',
        'ports': ['22']
      }
    ],
    'sourceRanges': [context.properties['network_cidr']],
    'targetTags': [
      context.properties['infra_id'] + '-master',
      context.properties['infra_id'] + '-worker'
    ]
  }
}, {
  'name': context.properties['infra_id'] + '-internal-cluster',
  'type': 'compute.v1.firewall',
  'properties': {
    'network': context.properties['cluster_network'],
    'allowed': [
      {
        'IPProtocol': 'udp',
        'ports': ['4789', '6081']
      },
      {
        'IPProtocol': 'tcp',
        'ports': ['9000-9999']
      },
      {
        'IPProtocol': 'udp',
        'ports': ['9000-9999']
      },
      {
        'IPProtocol': 'tcp',
        'ports': ['10250']
      },
      {
        'IPProtocol': 'tcp',
        'ports': ['30000-32767']
      }
    ]
  }
}

```

```

        'IPProtocol': 'udp',
        'ports': ['30000-32767']
    }],
    'sourceTags': [
        context.properties['infra_id'] + '-master',
        context.properties['infra_id'] + '-worker'
    ],
    'targetTags': [
        context.properties['infra_id'] + '-master',
        context.properties['infra_id'] + '-worker'
    ]
}
}, {
    'name': context.properties['infra_id'] + '-master-node-sa',
    'type': 'iam.v1.serviceAccount',
    'properties': {
        'accountId': context.properties['infra_id'] + '-m',
        'displayName': context.properties['infra_id'] + '-master-node'
    }
}, {
    'name': context.properties['infra_id'] + '-worker-node-sa',
    'type': 'iam.v1.serviceAccount',
    'properties': {
        'accountId': context.properties['infra_id'] + '-w',
        'displayName': context.properties['infra_id'] + '-worker-node'
    }
}
]

return {'resources': resources}

```

5.1.7. Creating the RHCOS cluster image for the GCP infrastructure

You must use a valid Red Hat Enterprise Linux CoreOS (RHCOS) image for Google Cloud Platform (GCP) for your OpenShift Container Platform nodes.

Procedure

1. Obtain the RHCOS 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 **rhcoss-<version>-gcp.tar**.

2. Export the following variable:

```
$ export IMAGE_SOURCE=<downloaded_image_file_path>
```

3. Create the cluster image:

```
$ gcloud compute images create "${INFRA_ID}-rhcoss-image" \
--source-uri="${IMAGE_SOURCE}"
```

5.1.8. Creating the bootstrap machine in GCP

You must create the bootstrap machine in Google Cloud Platform (GCP) to use during OpenShift Container Platform cluster initialization. One way to create this machine is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your bootstrap machine, 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 a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.
- Create and configure networking and load balancers in GCP.
- Create control plane and compute roles.

Procedure

1. Copy the template from the **Deployment Manager template for the bootstrap machine** section of this topic and save it as **04_bootstrap.py** on your computer. This template describes the bootstrap machine that your cluster requires.
2. Export the following variables required by the resource definition:

```
$ export CONTROL_SUBNET=`gcloud compute networks subnets describe ${INFRA_ID}-master-subnet --region=${REGION} --format json | jq -r .selfLink`
$ export CLUSTER_IMAGE=`gcloud compute images describe ${INFRA_ID}-rhcoss-image --format json | jq -r .selfLink`
$ export ZONE_0=`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[0] | cut -d "/" -f9`
$ export ZONE_1=`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[1] | cut -d "/" -f9`
$ export ZONE_2=`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[2] | cut -d "/" -f9`
```

3. Create a bucket and upload the **bootstrap.ign** file:

```
$ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
$ gsutil cp bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
```

4. Create a signed URL for the bootstrap instance to use to access the Ignition config. Export the URL from the output as a variable:

```
$ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json \
gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign | grep '^gs:' | awk '{print $5}'`
```

5. Create a **04_bootstrap.yaml** resource definition file:

```
$ cat <<EOF >04_bootstrap.yaml
imports:
- path: 04_bootstrap.py

resources:
- name: cluster-bootstrap
  type: 04_bootstrap.py
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2
    zone: '${ZONE_0}' 3

    cluster_network: '${CLUSTER_NETWORK}' 4
    control_subnet: '${CONTROL_SUBNET}' 5
    image: '${CLUSTER_IMAGE}' 6
    machine_type: 'n1-standard-4' 7
    root_volume_size: '128' 8

    bootstrap_ign: '${BOOTSTRAP_IGN}' 9
EOF
```

- 1** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 2** **region** is the region to deploy the cluster into, for example **us-east1**.
- 3** **zone** is the zone to deploy the bootstrap instance into, for example **us-east1-b**.
- 4** **cluster_network** is the **selfLink** URL to the cluster network.
- 5** **control_subnet** is the **selfLink** URL to the control subnet.
- 6** **image** is the **selfLink** URL to the RHCOS image.
- 7** **machine_type** is the machine type of the instance, for example **n1-standard-4**.
- 8** **bootstrap_ign** is the URL output when creating a signed URL above.

6. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
04_bootstrap.yaml
```

7. The templates do not manage load balancer membership due to limitations of Deployment Manager, so you must add the bootstrap machine manually:

```
$ gcloud compute target-pools add-instances \
    ${INFRA_ID}-api-target-pool --instances-zone="${ZONE_0}" --instances=${INFRA_ID}-
bootstrap
$ gcloud compute target-pools add-instances \
    ${INFRA_ID}-ign-target-pool --instances-zone="${ZONE_0}" --instances=${INFRA_ID}-
bootstrap
```

5.1.8.1. Deployment Manager template for the bootstrap machine

You can use the following Deployment Manager template to deploy the bootstrap machine that you need for your OpenShift Container Platform cluster:

04_bootstrap.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [
        {
            'name': context.properties['infra_id'] + '-bootstrap-public-ip',
            'type': 'compute.v1.address',
            'properties': {
                'region': context.properties['region']
            }
        },
        {
            'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
            'type': 'compute.v1.firewall',
            'properties': {
                'network': context.properties['cluster_network'],
                'allowed': [
                    {
                        'IPProtocol': 'tcp',
                        'ports': ['22']
                    }
                ],
                'sourceRanges': ['0.0.0.0/0'],
                'targetTags': [context.properties['infra_id'] + '-bootstrap']
            }
        },
        {
            'name': context.properties['infra_id'] + '-bootstrap',
            'type': 'compute.v1.instance',
            'properties': {
                'disks': [
                    {
                        'autoDelete': True,
                        'boot': True,
                        'initializeParams': {
                            'diskSizeGb': context.properties['root_volume_size'],
                            'sourceImage': context.properties['image']
                        }
                    }
                ],
                'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
                context.properties['machine_type'],
                'metadata': {
                    'items': [
                        {
                            'key': 'user-data',
                            'value': '{"ignition":{"config":{"replace":{"source":"' + context.properties['bootstrap_ign'] +
                            '", "verification":{}}, "timeouts":{}, "version":"2.1.0"}, "networkd":{}, "passwd":{}, "storage":{}, "systemd":{}}}'}
                    ]
                }
            }
        }
    ]
```

```

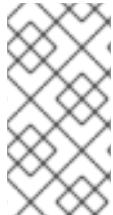
'networkInterfaces': [
    'subnetwork': context.properties['control_subnet'],
    'accessConfigs': [
        'natIP': '${ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address)'
    ]
},
'tags': {
    'items': [
        context.properties['infra_id'] + '-master',
        context.properties['infra_id'] + '-bootstrap'
    ]
},
'zone': context.properties['zone']
}
]

return {'resources': resources}

```

5.1.9. Creating the control plane machines in GCP

You must create the control plane machines in Google Cloud Platform (GCP) for your cluster to use. One way to create these machines is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your control plane machines, 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 a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.
- Create and configure networking and load balancers in GCP.
- Create control plane and compute roles.
- Create the bootstrap machine.

Procedure

1. Copy the template from the **Deployment Manager template for control plane machines** section of this topic and save it as **05_control_plane.py** on your computer. This template describes the control plane machines that your cluster requires.
2. Export the following variables needed by the resource definition:

```

$ export MASTER_SERVICE_ACCOUNT_EMAIL=`gcloud iam service-accounts list | grep
"^{${INFRA_ID}-master-node " | awk '{print $2}'`"
$ export MASTER_IGNITION=`cat master.ign`
```

3. Create a **05_control_plane.yaml** resource definition file:

```
$ cat <<EOF >05_control_plane.yaml
imports:
- path: 05_control_plane.py

resources:
- name: cluster-control-plane
  type: 05_control_plane.py
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2
    zones: 3
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' 4
    image: '${CLUSTER_IMAGE}' 5
    machine_type: 'n1-standard-4' 6
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT_EMAIL}' 7

    ignition: '${MASTER_IGNITION}' 8
EOF
```

- 1** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 2** **region** is the region to deploy the cluster into, for example **us-east1**.
- 3** **zones** are the zones to deploy the bootstrap instance into, for example **us-east1-b**, **us-east1-c**, and **us-east1-d**.
- 4** **control_subnet** is the **selfLink** URL to the control subnet.
- 5** **image** is the **selfLink** URL to the RHCOS image.
- 6** **machine_type** is the machine type of the instance, for example **n1-standard-4**.
- 7** **service_account_email** is the email address for the master service account created above.
- 8** **ignition** is the contents of the **master.ign** file.

4. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. The templates do not manage DNS entries due to limitations of Deployment Manager, so you must add the etcd entries manually:

```
$ export MASTER0_IP=`gcloud compute instances describe ${INFRA_ID}-m-0 --zone
${ZONE_0} --format json | jq -r .networkInterfaces[0].networkIP`
```

```
$ export MASTER1_IP=`gcloud compute instances describe ${INFRA_ID}-m-1 --zone ${ZONE_1} --format json | jq -r .networkInterfaces[0].networkIP`
$ export MASTER2_IP=`gcloud compute instances describe ${INFRA_ID}-m-2 --zone ${ZONE_2} --format json | jq -r .networkInterfaces[0].networkIP`
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${MASTER0_IP} --name etcd-
0.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-private-
zone
$ gcloud dns record-sets transaction add ${MASTER1_IP} --name etcd-
1.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-private-
zone
$ gcloud dns record-sets transaction add ${MASTER2_IP} --name etcd-
2.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-private-
zone
$ gcloud dns record-sets transaction add \
"0 10 2380 etcd-0.${CLUSTER_NAME}.${BASE_DOMAIN}." \
"0 10 2380 etcd-1.${CLUSTER_NAME}.${BASE_DOMAIN}." \
"0 10 2380 etcd-2.${CLUSTER_NAME}.${BASE_DOMAIN}." \
--name _etcd-server-ssl._tcp.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type SRV -
-zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

6. The templates do not manage load balancer membership due to limitations of Deployment Manager, so you must add the control plane machines manually:

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_0}" --instances=${INFRA_ID}-m-0
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_1}" --instances=${INFRA_ID}-m-1
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_2}" --instances=${INFRA_ID}-m-2
$ gcloud compute target-pools add-instances ${INFRA_ID}-ign-target-pool --instances-
zone="${ZONE_0}" --instances=${INFRA_ID}-m-0
$ gcloud compute target-pools add-instances ${INFRA_ID}-ign-target-pool --instances-
zone="${ZONE_1}" --instances=${INFRA_ID}-m-1
$ gcloud compute target-pools add-instances ${INFRA_ID}-ign-target-pool --instances-
zone="${ZONE_2}" --instances=${INFRA_ID}-m-2
```

5.1.9.1. Deployment Manager template for control plane machines

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

05_control_plane.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [
        {
            'name': context.properties['infra_id'] + '-m-0',
            'type': 'compute.v1.instance',
            'properties': {
                'disks': [
                    {
                        'autoDelete': True,
                        'boot': True,
```

```

'initializeParams': {
    'diskSizeGb': context.properties['root_volume_size'],
    'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
    'sourceImage': context.properties['image']
}
}],
'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
context.properties['machine_type'],
'metadata': {
    'items': [
        {'key': 'user-data',
         'value': context.properties['ignition']}
    ]
},
'networkInterfaces': [
    {'subnetwork': context.properties['control_subnet']}
],
'serviceAccounts': [
    {'email': context.properties['service_account_email'],
     'scopes': ['https://www.googleapis.com/auth/cloud-platform']}
],
'tags': {
    'items': [
        context.properties['infra_id'] + '-master',
    ]
},
'zone': context.properties['zones'][0]
},
{
'name': context.properties['infra_id'] + '-m-1',
'type': 'compute.v1.instance',
'properties': {
    'disks': [
        {'autoDelete': True,
         'boot': True,
         'initializeParams': {
             'diskSizeGb': context.properties['root_volume_size'],
             'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
             'sourceImage': context.properties['image']
         }
     ],
     'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
context.properties['machine_type'],
     'metadata': {
         'items': [
             {'key': 'user-data',
              'value': context.properties['ignition']}
         ]
     },
     'networkInterfaces': [
         {'subnetwork': context.properties['control_subnet']}
     ],
     'serviceAccounts': [
         {'email': context.properties['service_account_email'],
          'scopes': ['https://www.googleapis.com/auth/cloud-platform']}
     ],
}
}

```

```

'tags': {
  'items': [
    context.properties['infra_id'] + '-master',
  ]
},
'zone': context.properties['zones'][1]
}
}, {
  'name': context.properties['infra_id'] + '-m-2',
  'type': 'compute.v1.instance',
  'properties': {
    'disks': [{
      'autoDelete': True,
      'boot': True,
      'initializeParams': {
        'diskSizeGb': context.properties['root_volume_size'],
        'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
        'sourceImage': context.properties['image']
      }
    }],
    'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
    'metadata': {
      'items': [
        {
          'key': 'user-data',
          'value': context.properties['ignition']
        }
      ],
      'networkInterfaces': [
        {
          'subnetwork': context.properties['control_subnet']
        }
      ],
      'serviceAccounts': [
        {
          'email': context.properties['service_account_email'],
          'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }
      ],
      'tags': {
        'items': [
          context.properties['infra_id'] + '-master',
        ]
      },
      'zone': context.properties['zones'][2]
    }
  }
}

return {'resources': resources}

```

5.1.10. Wait for bootstrap completion and remove bootstrap resources in GCP

After you create all of the required infrastructure in Google Cloud Platform (GCP), wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.
- Create and configure networking and load balancers in GCP.
- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane 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 info ②
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

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

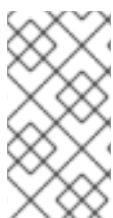
2. Delete the bootstrap resources:

```
$ gcloud compute target-pools remove-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_0}" --instances=${INFRA_ID}-bootstrap
$ gcloud compute target-pools remove-instances ${INFRA_ID}-ign-target-pool --instances-
zone="${ZONE_0}" --instances=${INFRA_ID}-bootstrap
$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

5.1.11. Creating additional worker machines in GCP

You can create worker machines in Google Cloud Platform (GCP) for your cluster to use by launching individual instances discretely or by automated processes outside the cluster, such as Auto Scaling Groups. You can also take advantage of the built-in cluster scaling mechanisms and the machine API in OpenShift Container Platform.

In this example, you manually launch one instance by using the Deployment Manager template. Additional instances can be launched by including additional resources of type **06_worker.py** in the file.



NOTE

If you do not use the provided Deployment Manager template to create your worker machines, 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 a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.
- Create and configure networking and load balancers in GCP.
- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.

Procedure

1. Copy the template from the **Deployment Manager template for worker machines** section of this topic and save it as **06_worker.py** on your computer. This template describes the worker machines that your cluster requires.
2. Export the following variables needed by the resource definition:

```
$ export COMPUTE_SUBNET=`gcloud compute networks subnets describe ${INFRA_ID}-worker-subnet --region=${REGION} --format json | jq -r .selfLink`
$ export WORKER_SERVICE_ACCOUNT_EMAIL=`gcloud iam service-accounts list | grep "^\${INFRA_ID}-worker-node " | awk '{print $2}'`
$ export WORKER_IGNITION=`cat worker.ign`
```

3. Create a **06_worker.yaml** resource definition file:

```
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'w-a-0' 1
  type: 06_worker.py
  properties:
    infra_id: '\${INFRA_ID}' 2
    region: '\${REGION}' 3
    zone: '\${ZONE_0}' 4

    compute_subnet: '\${COMPUTE_SUBNET}' 5
    image: '\${CLUSTER_IMAGE}' 6
    machine_type: 'n1-standard-4' 7
    root_volume_size: '128'
    service_account_email: '\${WORKER_SERVICE_ACCOUNT_EMAIL}' 8

    ignition: '\${WORKER_IGNITION}' 9
EOF
```

1 **name** is the name of the worker machine, for example **w-a-0**.

2 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

- 3 **region** is the region to deploy the cluster into, for example **us-east1**.
- 4 **zone** is the zone to deploy the worker machine into, for example **us-east1-b**.
- 5 **compute_subnet** is the **selfLink** URL to the compute subnet.
- 6 **image** is the **selfLink** URL to the RHCOS image.
- 7 **machine_type** is the machine type of the instance, for example **n1-standard-4**.
- 8 **service_account_email** is the email address for the worker service account created above.
- 9 **ignition** is the contents of the **worker.ign** file.

4. Optional: If you want to launch additional instances, include additional resources of type **06_worker.py** in your **06_worker.yaml** resource definition file.
5. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config 06_worker.yaml
```

5.1.11.1. Deployment Manager template for worker machines

You can use the following Deployment Manager template to deploy the worker machines that you need for your OpenShift Container Platform cluster:

06_worker.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [
        {
            'name': context.properties['infra_id'] + '-' + context.env['name'],
            'type': 'compute.v1.instance',
            'properties': {
                'disks': [
                    {
                        'autoDelete': True,
                        'boot': True,
                        'initializeParams': {
                            'diskSizeGb': context.properties['root_volume_size'],
                            'sourceImage': context.properties['image']
                        }
                    }
                ],
                'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
                context.properties['machine_type'],
                'metadata': {
                    'items': [
                        {
                            'key': 'user-data',
                            'value': context.properties['ignition']
                        }
                    ],
                    'networkInterfaces': [
                        {
                            'subnetwork': context.properties['compute_subnet']
                        }
                    ]
                }
            }
        }
    ]
```

```
'serviceAccounts': [
    'email': context.properties['service_account_email'],
    'scopes': ['https://www.googleapis.com/auth/cloud-platform']
],
'tags': {
    'items': [
        context.properties['infra_id'] + '-worker',
    ]
},
'zone': context.properties['zone']
}

return {'resources': resources}
```

5.1.12. Installing the CLI

You can install the CLI in order to interact with OpenShift Container Platform using a command-line interface.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.2. Download and install the new version of **oc**.

Procedure

1. From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**.
 2. Click the folder for your operating system and architecture and click the compressed file.
- A decorative icon consisting of a grid of small squares forming a larger diamond pattern.
- #### NOTE
- You can install **oc** on Linux, Windows, or macOS.
3. Save the file to your file system.
 4. Extract the compressed file.
 5. Place it in a directory that is on your **PATH**.

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

5.1.13. Logging in to the cluster

You can log in to 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
```

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

2. Verify you can run **oc** commands successfully using the exported configuration:

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

5.1.14. 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.
- Install the **jq** package.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes

NAME      STATUS    ROLES   AGE     VERSION
master-0   Ready     master   63m    v1.14.6+c4799753c
master-1   Ready     master   63m    v1.14.6+c4799753c
master-2   Ready     master   64m    v1.14.6+c4799753c
worker-0   NotReady  worker   76s    v1.14.6+c4799753c
worker-1   NotReady  worker   70s    v1.14.6+c4799753c
```

The output lists all of the machines that you created.

2. Review the pending certificate signing requests (CSRs) and ensure that 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-8vnp 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.

3. 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
```

5.1.15. Optional: Adding the ingress DNS records

If you removed the DNS Zone configuration when creating Kubernetes manifests and generating Ignition configs, you must manually create DNS records that point at the ingress load balancer. You can create either a wildcard ***.apps.{baseDomain}**, or specific records. You can use A, CNAME, and other records per your requirements.

Prerequisites

- Configure a GCP account.

- Remove the DNS Zone configuration when creating Kubernetes manifests and generating Ignition configs.
- Create and configure a VPC and associated subnets in GCP.
- Create and configure networking and load balancers in GCP.
- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.
- Create the worker machines.

Procedure

1. Wait for the Ingress router to create a load balancer and populate the **EXTERNAL-IP** field:

```
$ oc -n openshift-ingress get service router-default
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
router-default  LoadBalancer  172.30.18.154  35.233.157.184
              80:32288/TCP,443:31215/TCP  98
```

2. Add the A record to your public and private zones:

```
$ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-headers | awk
'{print $4}'`  
  

$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}  
  

$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

If you prefer to add explicit domains instead of using a wildcard, you can create entries for each of the cluster's current routes:

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]'{range .status.ingress[*]}{.host}{"\n"}  

{end}{end}' routes
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
grafana-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

5.1.16. Completing a GCP installation on user-provisioned infrastructure

After you start the OpenShift Container Platform installation on Google Cloud Platform (GCP) user-provisioned infrastructure, you can monitor the cluster events until the cluster is ready.

Prerequisites

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

Procedure

1. 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.



IMPORTANT

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

2. Observe the running state of your cluster.

- a. Run the following command to view the current cluster version and status:

```
$ oc get clusterversion
NAME      VERSION  AVAILABLE  PROGRESSING  SINCE  STATUS
version    False     True       24m        Working towards 4.2.0-0: 99% complete
```

- b. Run the following command to view the Operators managed on the control plane by the Cluster Version Operator (CVO):

```
$ oc get clusteroperators
NAME          VERSION  AVAILABLE  PROGRESSING  DEGRADED
authentication  4.2.0-0  True       False       False   6m18s
cloud-credential 4.2.0-0  True       False       False   17m
cluster-autoscaler 4.2.0-0  True       False       False   80s
console         4.2.0-0  True       False       False   3m57s
dns             4.2.0-0  True       False       False   22m
image-registry  4.2.0-0  True       False       False   5m4s
ingress         4.2.0-0  True       False       False   4m38s
insights        4.2.0-0  True       False       False   21m
kube-apiserver  4.2.0-0  True       False       False   12m
```

kube-controller-manager	4.2.0-0	True	False	False	12m
kube-scheduler	4.2.0-0	True	False	False	11m
machine-api	4.2.0-0	True	False	False	18m
machine-config	4.2.0-0	True	False	False	22m
marketplace	4.2.0-0	True	False	False	5m38s
monitoring	4.2.0-0	True	False	False	86s
network	4.2.0-0	True	False	False	14m
node-tuning	4.2.0-0	True	False	False	6m8s
openshift-apiserver	4.2.0-0	True	False	False	6m48s
openshift-controller-manager	4.2.0-0	True	False	False	12m
openshift-samples	4.2.0-0	True	False	False	67s
operator-lifecycle-manager	4.2.0-0	True	False	False	15m
operator-lifecycle-manager-catalog	4.2.0-0	True	False	False	15m
operator-lifecycle-manager-packageserver	4.2.0-0	True	False	False	6m48s
service-ca	4.2.0-0	True	False	False	17m
service-catalog-apiserver	4.2.0-0	True	False	False	6m18s
service-catalog-controller-manager	4.2.0-0	True	False	False	6m19s
storage	4.2.0-0	True	False	False	6m20s

- c. Run the following command to view your cluster Pods:

```
$ oc get pods --all-namespaces
NAMESPACE          NAME
READY   STATUS  RESTARTS AGE
kube-system        etcd-member-ip-10-0-3-111.us-east-
2.compute.internal 1/1    Running  0      35m
kube-system        etcd-member-ip-10-0-3-239.us-east-
2.compute.internal 1/1    Running  0      37m
kube-system        etcd-member-ip-10-0-3-24.us-east-
2.compute.internal 1/1    Running  0      35m
openshift-apiserver-operator
h7t2t           1/1    Running  1      37m
openshift-apiserver
1/1    Running  0      30m
openshift-apiserver
1/1    Running  0      29m
openshift-apiserver
1/1    Running  0      29m
...
openshift-service-ca-operator
9r257           1/1    Running  0      37m
openshift-service-ca
1/1    Running  0      35m
openshift-service-ca
25qn6           1/1    Running  0      35m
openshift-service-ca
1/1    Running  0      35m
openshift-service-catalog-apiserver-operator
operator-549f44668b-b5q2w 1/1    Running  0      32m
openshift-service-catalog-controller-manager-operator
controller-manager-operator-b78cr2lnm 1/1    Running  0      31m
```

When the current cluster version is **AVAILABLE**, the installation is complete.

Final steps

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

CHAPTER 6. INSTALLING ON BARE METAL

6.1. INSTALLING A CLUSTER ON BARE METAL

In OpenShift Container Platform version 4.2, 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 allow the sites](#) that your cluster requires access to.



NOTE

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

6.1.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.2, you require access to the internet to install and entitle your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager](#). From there, you can allocate entitlements to your cluster.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management and entitlement. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster. If the Telemetry service cannot entitle your cluster, you must manually entitle it on the [Cluster registration](#) page.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

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

6.1.2. 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.

6.1.2.1. 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.

6.1.2.2. 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.

6.1.2.3. 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

Machine	Operating System	vCPU	RAM	Storage
Compute	RHCOS or RHEL 7.6	2	8 GB	120 GB

6.1.2.4. 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.

6.1.3. Creating the user-provisioned infrastructure

Before you deploy an 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.

6.1.3.1. 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 supported 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 6.1. All machines to all machines

Protocol	Port	Description
TCP	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
UDP	4789	VXLAN and GENEVE
	6081	VXLAN and GENEVE
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
	30000-32767	Kubernetes NodePort

Network topology requirements

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



IMPORTANT

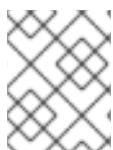
OpenShift Container Platform requires all nodes to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

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

Port	Machines	Internal	External	Description
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.

6.1.3.2. User-provisioned DNS requirements

The following DNS records are required for an 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 6.2. Required DNS records

Component	Record	Description
Kubernetes API	<code>api.<cluster_name>.<base_domain></code>	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.

Component	Record	Description
	<code>api-int.<cluster_name>.<base_domain></code>	<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>  <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>
Routes	<code>*.apps.<cluster_name>.<base_domain></code>	<p>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.</p>

Component	Record	Description
etcd	etcd-<index>.<cluster_name>.<base_domain>	<p>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.</p>
	_etcd-server-ssl._tcp.<cluster_name>.<base_domain>	<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>

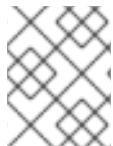
```
# _service._proto.name.
TTL  class SRV priority weight port target.
_etcd-server-ssl._tcp.<cluster_name>.<base_domain> 86400 IN
SRV 0    10   2380 etcd-0.
<cluster_name>.<base_domain>.
```

```
_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>.
```

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.

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 installation program. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

6.1.5. 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 500 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.

6.1.6. Installing the CLI

You can install the CLI in order to interact with OpenShift Container Platform using a command-line interface.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.2. Download and install the new version of **oc**.

Procedure

1. From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**.
2. Click the folder for your operating system and architecture and click the compressed file.



NOTE

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

3. Save the file to your file system.
4. Extract the compressed file.
5. Place it in a directory that is on your **PATH**.

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

6.1.7. 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.

6.1.7.1. 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:
  clusterNetwork:
  - cidr: 10.128.0.0/14 10
    hostPrefix: 23 11
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16 12
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 pod IPs addresses. If you are required to provide access to nodes from an external network, configure load balancers and routers to manage the traffic.
- 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 on, specify an SSH key that your **ssh-agent** process uses.

6.1.7.2. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.



NOTE

For bare metal installations, if you do not assign node IP addresses from the range that is specified in the **networking.machineCIDR** field in the **install-config.yaml** file, you must include them in the **proxy.noProxy** field.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the Proxy object's **spec.noProxy** field to bypass the proxy if necessary.

**NOTE**

The Proxy object's **status.noProxy** field is populated by default with the instance metadata endpoint (**169.254.169.254**) and with the values of the **networking.machineCIDR**, **networking.clusterNetwork.cidr**, and **networking.serviceNetwork** fields from your installation configuration.

Procedure

- 1 Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> ①
  httpsProxy: http://<username>:<pswd>@<ip>:<port> ②
  noProxy: example.com ③
  additionalTrustBundle: | ④
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
...
```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. The URL scheme must be **http**; **https** is currently not supported.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to include all subdomains of that domain. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a ConfigMap that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** ConfigMap that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this ConfigMap is referenced in the Proxy object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

**NOTE**

The installation program does not support the proxy **readinessEndpoints** field.

- 2 Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster** Proxy object is still created, but it will have a nil **spec**.

**NOTE**

Only the Proxy object named **cluster** is supported, and no additional proxies can be created.

6.1.8. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and 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.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. 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 installation directory that contains the **install-config.yaml** file you created.

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

2. Modify the **manifests/cluster-scheduler-02-config.yml** Kubernetes manifest file to prevent Pods from being scheduled on the control plane machines:
 - a. Open the **manifests/cluster-scheduler-02-config.yml** file.
 - b. Locate the **mastersSchedulable** parameter and set its value to **False**.
 - c. Save and exit the file.

**NOTE**

Currently, due to a [Kubernetes limitation](#), router Pods running on control plane machines will not be reachable by the ingress load balancer. This step might not be required in a future minor version of OpenShift Container Platform.

3. 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
  
```

6.1.9. 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.

6.1.9.1. 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: **rhcoss-<version>-<architecture>-installer.iso**
- Compressed metal BIOS: **rhcoss-<version>-<architecture>-metal-bios.raw.gz**
- Compressed metal UEFI: **rhcoss-<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.

6.1.9.2. 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: **rhcoss-<version>-<architecture>-installer.iso**
 - Compressed metal BIOS: **rhcoss-<version>-<architecture>-metal-bios.raw.gz**
 - **kernel: rhcoss-<version>-<architecture>-installer-kernel**
 - **initramfs: rhcoss-<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>/rhcoss-<version>-<architecture>-installer-kernel 1
    APPEND ip=dhcp rd.neednet=1 initrd=http://<HTTP_server>/rhcoss-<version>-<architecture>-installer-initramfs.img console=tty0 console=ttyS0 coreos.inst=yes
      coreos.inst.install_dev=sda coreos.inst.image_url=http://<HTTP_server>/rhcoss-<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**

- 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
```

- 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.
- 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**.
- Specify the location of the **initramfs** file that you uploaded to your HTTP server.

- 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
}
```

- 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.
- Specify the location of the **initramfs** file that you uploaded to your HTTP server.

- 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.

6.1.10. Creating the cluster

To create the OpenShift Container Platform cluster, you wait for the bootstrap process to complete on the machines that you provisioned 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. Monitor the bootstrap process:

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

INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.14.6+c4799753c 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 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

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.

6.1.11. Logging in to the cluster

You can log in to 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
```

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

2. Verify you can run **oc** commands successfully using the exported configuration:

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

6.1.12. 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.
- Install the **jq** package.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes  
  
NAME     STATUS   ROLES   AGE    VERSION  
master-0  Ready    master   63m   v1.14.6+c4799753c  
master-1  Ready    master   63m   v1.14.6+c4799753c  
master-2  Ready    master   64m   v1.14.6+c4799753c  
worker-0  NotReady worker  76s   v1.14.6+c4799753c  
worker-1  NotReady worker  70s   v1.14.6+c4799753c
```

The output lists all of the machines that you created.

2. Review the pending certificate signing requests (CSRs) and ensure that 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.

3. 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
```

6.1.13. 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	4.2.0	True	False	False	69s	
cloud-credential	4.2.0	True	False	False	12m	
cluster-autoscaler	4.2.0	True	False	False	11m	
console	4.2.0	True	False	False	46s	
dns	4.2.0	True	False	False	11m	
image-registry	4.2.0	False	True	False	5m26s	
ingress	4.2.0	True	False	False	5m36s	
kube-apiserver	4.2.0	True	False	False	8m53s	
kube-controller-manager	4.2.0	True	False	False	7m24s	
kube-scheduler	4.2.0	True	False	False	12m	
machine-api	4.2.0	True	False	False	12m	
machine-config	4.2.0	True	False	False	7m36s	
marketplace	4.2.0	True	False	False	7m54m	
monitoring	4.2.0	True	False	False	7h54s	
network	4.2.0	True	False	False	5m9s	
node-tuning	4.2.0	True	False	False	11m	
openshift-apiserver	4.2.0	True	False	False	11m	
openshift-controller-manager	4.2.0	True	False	False	5m943s	
openshift-samples	4.2.0	True	False	False	3m55s	
operator-lifecycle-manager	4.2.0	True	False	False	11m	
operator-lifecycle-manager-catalog	4.2.0	True	False	False	11m	
service-ca	4.2.0	True	False	False	11m	
service-catalog-apiserver	4.2.0	True	False	False	5m26s	
service-catalog-controller-manager	4.2.0	True	False	False	5m25s	
storage	4.2.0	True	False	False	5m30s	

- Configure the Operators that are not available.

6.1.13.1. 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.

6.1.13.1.1. 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

- To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.
- 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 cannot 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
```

3. 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.

4. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

6.1.13.1.2. 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.

6.1.14. 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 SINCE	VERSION	AVAILABLE	PROGRESSING	DEGRADED
authentication	4.2.0	True	False	False 10m
cloud-credential	4.2.0	True	False	False 22m
cluster-autoscaler	4.2.0	True	False	False 21m
console	4.2.0	True	False	False 10m
dns	4.2.0	True	False	False 21m
image-registry	4.2.0	True	False	False 16m
ingress	4.2.0	True	False	False 16m
kube-apiserver	4.2.0	True	False	False 19m
kube-controller-manager	4.2.0	True	False	False 18m
kube-scheduler	4.2.0	True	False	False 22m
machine-api	4.2.0	True	False	False 22m
machine-config	4.2.0	True	False	False 18m
marketplace	4.2.0	True	False	False 18m
monitoring	4.2.0	True	False	False 18m
network	4.2.0	True	False	False 16m
node-tuning	4.2.0	True	False	False 21m
openshift-apiserver	4.2.0	True	False	False 21m
openshift-controller-manager	4.2.0	True	False	False 17m
openshift-samples	4.2.0	True	False	False 14m
operator-lifecycle-manager	4.2.0	True	False	False 21m
operator-lifecycle-manager-catalog	4.2.0	True	False	False 21m
service-ca	4.2.0	True	False	False 21m
service-catalog-apiserver	4.2.0	True	False	False 16m
service-catalog-controller-manager	4.2.0	True	False	False 16m
storage	4.2.0	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.



IMPORTANT

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

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
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
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> 1
```

- 1** 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 remote health reporting](#).

CHAPTER 7. INSTALLING ON OPENSTACK

7.1. INSTALLING A CLUSTER ON OPENSTACK WITH CUSTOMIZATIONS

In OpenShift Container Platform version 4.2, you can install a customized cluster on Red Hat OpenStack Platform (RHOSP). To customize the installation, modify parameters in the `install-config.yaml` before you install the cluster.

Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- Have access to a RHOSP administrator's account

7.1.1. Resource guidelines for installing OpenShift Container Platform on OpenStack

Your quota must meet the following requirements to run the OpenShift Container Platform installation program in Red Hat OpenStack Platform (RHOSP):

Table 7.1. Recommended resources for a default OpenShift Container Platform cluster on RHOSP

Resource	Value
Floating IP addresses	2
Ports	15
Routers	1
Subnets	1
RAM	112 GB
vCPUs	28
Volume storage	175 GB
Instances	7
Security groups	3
Security group rules	60
Swift containers	2
Swift objects	1
Swift available space	10 MB or more

**NOTE**

Swift space requirements vary depending on the size of the bootstrap Ignition file and image registry.

A cluster might function with fewer than recommended resources, but its performance is not guaranteed.

**NOTE**

By default, your security group and security group rule quotas might be low. If you encounter problems, run **openstack quota set --secgroups 3 --secgroup-rules 60 <project>** to increase them.

An OpenShift Container Platform deployment comprises control plane machines, compute machines, and a bootstrap machine.

7.1.1.1. Control plane and compute machines

By default, the OpenShift Container Platform installation program stands up three control plane and compute machines.

Each machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

TIP

Compute machines host the applications that you run on OpenShift Container Platform; aim to run as many as you can.

7.1.1.2. Bootstrap machine

At installation, a bootstrap machine is temporarily provisioned to stand up the control plane. After the production control plane is ready, the bootstrap machine is deprovisioned.

The bootstrap machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

**NOTE**

The installation program cannot pass certificate authority bundles to Ignition on control plane machines. Therefore, the bootstrap machine cannot retrieve Ignition configurations from Swift if your endpoint uses self-signed certificates.

7.1.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.2, you require access to the internet to install and entitle your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager](#). From there, you can allocate entitlements to your cluster.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management and entitlement. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster. If the Telemetry service cannot entitle your cluster, you must manually entitle it on the [Cluster registration](#) page.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

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

7.1.3. Enabling Swift on OpenStack

OpenShift Container Platform on Red Hat OpenStack Platform (RHOSP) uses [OpenStack Object Storage \(Swift\)](#) to store and serve user configuration files.

Swift is operated by a user account with the **swiftoperator** role and **temp-url** support.

Prerequisites

- A RHOSP administrator account on the target environment
- On Ceph RGW, [the account in url option must be enabled](#)

Procedure

To enable Swift on RHOSP:

1. As an administrator in the RHOSP CLI, add the **swiftoperator** role to the account that will access Swift:

```
$ openstack role add --user <user> --project <project> swiftoperator
```

2. As the account with the **swiftoperator** role, set a temporary URL property for the account:

```
$ openstack object store account set --property Temp-URL-Key=superkey
```

Your RHOSP deployment can now use Swift to store and serve files.

7.1.4. Creating the Red Hat Enterprise Linux CoreOS (RHCOS) image

The OpenShift Container Platform installation program requires that a Red Hat Enterprise Linux CoreOS (RHCOS) image be present in the Red Hat OpenStack Platform (RHOSP) cluster. Retrieve the latest RHCOS image, then upload it using the RHOSP CLI.

Prerequisites

- The RHOSP CLI is installed.

Procedure

1. Download the latest RHCOS image from the Red Hat customer portal's [Product Downloads page](#).



NOTE

The RHOSP QCOW2 images are delivered in compressed format, so you must specify additional options to download them. With `curl`, specify `curl --compressed -J -L -O <image_url>`. With `wget`, specify `wget --compression=auto <image_url>`.

2. From the image that you downloaded, create an image that is named **rhcoss** to your cluster by using the RHOSP CLI:

```
$ openstack image create --container-format=bare --disk-format=qcow2 --file rhcos-$(RHCOSVERSION)-openstack.qcow2 rhcos
```

CAUTION

If the installation program finds multiple images with the same name, it chooses one of them at random. To avoid this behavior, create unique names for resources in RHOSP.



NOTE

Depending on your RHOSP environment, the image can be in either **qcow2** or **raw** formats. If your environment requires the **raw** format, substitute **raw** for **qcow2** in the preceding example. For more information about image formats, see [Disk and container formats for images](#).

After you upload the image to RHOSP, it is available to the installation program.

7.1.5. Verifying external network access

The OpenShift Container Platform installer requires external network access. You must provide an external network value to it, or deployment fails. Before you run the installer, verify that a network with the External router type exists in Red Hat OpenStack Platform (RHOSP).

Prerequisites

- Configure OpenStack's networking service to have DHCP agents forward instances' DNS queries

Procedure

1. Using the RHOSP CLI, verify the name and ID of the 'External' network:

```
$ openstack network list --long -c ID -c Name -c "Router Type"
+-----+-----+-----+
| ID      | Name    | Router Type |
+-----+-----+-----+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External   |
+-----+-----+-----+
```

A network with an External router type appears in the network list. If at least one does not, see [Create an external network](#).



IMPORTANT

If the external network's CIDR range overlaps one of the default network ranges, you must change the matching network ranges in the **install-config.yaml** file before you run the installation program.

The default network ranges are:

Network	Range
machineCIDR	10.0.0.0/16
serviceNetwork	172.30.0.0/16
clusterNetwork	10.128.0.0/14

CAUTION

If the installation program finds multiple networks with the same name, it sets one of them at random. To avoid this behavior, create unique names for resources in RHOSP.



NOTE

If the Neutron trunk service plug-in is enabled, a trunk port is created by default. For more information, see [Neutron trunk port](#).

7.1.6. Defining parameters for the installation program

The OpenShift Container Platform installation program relies on a file called **clouds.yaml**. The file describes Red Hat OpenStack Platform (RHOSP) configuration parameters, including the project name, log in information, and authorization service URLs.

Procedure

1. Create the **clouds.yaml** file:

- If your OpenStack distribution includes the Horizon web UI, generate a **clouds.yaml** file in it.
- If your OpenStack distribution does not include the Horizon web UI, or you do not want to use Horizon, create the file yourself. For detailed information about **clouds.yaml**, see [Config files](#) in the RHOSP documentation.

```

clouds:
shiftstack:
auth:
auth_url: http://10.10.14.42:5000/v3
project_name: shiftstack
username: shiftstack_user
password: XXX
user_domain_name: Default
project_domain_name: Default
dev-env:
region_name: RegionOne
auth:
username: 'devuser'
password: XXX
project_name: 'devonly'
auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. Place the file that you generate in one of the following locations:
 - a. The value of the **OS_CLIENT_CONFIG_FILE** environment variable
 - b. The current directory
 - c. A Unix-specific user configuration directory, for example **~/.config/openstack/clouds.yaml**
 - d. A Unix-specific site configuration directory, for example **/etc/openstack/clouds.yaml**
The installation program searches for **clouds.yaml** in that order.

7.1.7. 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 500 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.

7.1.8. Creating the installation configuration file

You can customize your installation of OpenShift Container Platform on OpenStack.

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 on, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **openstack** as the platform to target.
 - iii. Specify the Red Hat OpenStack Platform (RHOSP) external network name to use for installing the cluster.
 - iv. Specify the Floating IP address to use for external access to the OpenShift API.
 - v. Specify a RHOSP flavor with at least 16 GB RAM to use for control plane and compute nodes.
 - vi. Select the base domain to deploy the cluster to. All DNS records will be sub-domains of this base and will also include the cluster name.
 - vii. Enter a name for your cluster. The name must be 14 or fewer characters long.
 - viii. 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.
 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.

7.1.9. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster 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 in the **install-config.yaml** file after installation.

Table 7.2. 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 <code><metadata.name>.<baseDomain></code> 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, azure, gcp, openstack , or {}
compute.platform	The cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack , or {}
metadata.name	The name of your cluster.	A string that contains uppercase or lowercase letters, such as dev . The string must be 14 characters or fewer long.
platform.<platform>.region	The region to deploy your cluster in.	A valid region for your cloud, such as us-east-1 for AWS, centralus for Azure, or region1 for Red Hat OpenStack Platform (RHOSP).
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 7.3. Optional parameters

Parameter	Description	Values
sshKey	The SSH key to use to access your cluster machines.	A valid, local public SSH key that you added to the ssh-agent process.
	<p></p> <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery on, specify an SSH key that your ssh-agent process uses.</p>	
compute.hyperthreading	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.	Enabled or Disabled
	<p></p> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p>	
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 .

Parameter	Description	Values
controlPlane.hyperthreading	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.	Enabled or Disabled
	<p>IMPORTANT</p>  <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p>	
controlPlane.replicas	The number of control plane machines to provision.	A positive integer greater than or equal to 3 . The default value is 3 .

Table 7.4. Additional Red Hat OpenStack Platform (RHOSP) parameters

Parameter	Description	Values
compute.platform.openstack.rootVolume.size	For compute machines, the size in gigabytes of the root volume. If you do not set this value, machines use ephemeral storage.	Integer, for example 30 .
compute.platform.openstack.rootVolume.type	For compute machines, the root volume's type.	String, for example performance .
controlPlane.platform.openstack.rootVolume.size	For control plane machines, the size in gigabytes of the root volume. If you do not set this value, machines use ephemeral storage.	Integer, for example 30 .

Parameter	Description	Values
controlPlane.platform.openstack.rootVolume.type	For control plane machines, the root volume's type.	String, for example performance .
machines.platform.openstack.region	The region where the RHOSP cluster is created.	String, for example region1 .
machines.platform.openstack.cloud	The name of the RHOSP cloud to use from the list of clouds in the clouds.yaml file.	String, for example MyCloud .
machines.platform.openstack.externalNetwork	The RHOSP external network name to be used for installation.	String, for example external .
machines.platform.openstack.computeFlavor	The RHOSP flavor to use for control plane and compute machines.	String, for example m1.xlarge .
machines.platform.openstack.lbFloatingIP	An existing floating IP address to associate with the load balancer API.	An IP address, for example 128.0.0.1 .
machines.platform.openstack.trunkSupport	Whether RHOSP ports can be trunked.	true or false
machines.platform.openstack.octaviaSupport	Whether RHOSP supports Octavia.	true or false
machines.platform.openstack.defaultMachinePlatform	<i>Optional.</i> The default machine pool platform configuration.	<pre>{ "type": "ml.large", "rootVolume": { "size": 30, "type": "performance" } }</pre>

7.1.9.1. Sample customized `install-config.yaml` file for OpenStack

This sample `install-config.yaml` demonstrates all of the possible Red Hat OpenStack Platform (RHOSP) customization options.



IMPORTANT

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

```
apiVersion: v1
baseDomain: example.com
clusterID: os-test
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineCIDR: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: OpenShiftSDN
platform:
  openstack:
    region: region1
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    lbFloatingIP: 128.0.0.1
    trunkSupport: false
    octaviaSupport: false
  pullSecret: '{"auths": ...}'
  sshKey: ssh-ed25519 AAAA...
```

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.

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 installation program.

7.1.11. Enabling access to the environment

At deployment, all OpenShift Container Platform machines are created in a Red Hat OpenStack Platform (RHOSP)-tenant network. Therefore, they are not accessible directly in most RHOSP deployments.

You can configure the OpenShift Container Platform API to be accessible either with or without floating IP addresses.

7.1.11.1. Enabling access with floating IP addresses

Make OpenShift Container Platform API endpoints accessible by attaching two floating IP (FIP) addresses to them: one for the API load balancer (**lb FIP**), and one for OpenShift Container Platform applications (**apps FIP**).



IMPORTANT

The load balancer FIP is also used in the **install-config.yaml** file.

Procedure

1. Using the Red Hat OpenStack Platform (RHOSP) CLI, create a new external network:

```
$ openstack floating ip create <external network>
```

2. Add a record that follows this pattern to your DNS server:

```
api.<cluster name>.<base domain> IN A <lb FIP>
```



NOTE

If you do not control the DNS server you can add the record to your **/etc/hosts** file instead. This action makes the API accessible to you only, which is not suitable for production deployment but does allow installation for development and testing.

TIP

You can make OpenShift Container Platform resources available outside of the cluster by assigning a floating IP address and updating your firewall configuration.

7.1.11.2. Enabling access without floating IP addresses

If you cannot use floating IP addresses, the OpenShift Container Platform installation might still finish. However, the installation program fails after it times out waiting for API access.

After the installation program times out, the cluster might still initialize. After the bootstrapping processing begins, it must complete. You must edit the cluster's networking configuration after it is deployed, however.

7.1.12. Deploy the cluster

You can install OpenShift Container Platform on a compatible cloud.



IMPORTANT

You can run the **create cluster** command of 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=info ②
```

- 1 For <installation_directory>, specify the location of your customized `./install-config.yaml` file.
- 2 To view different installation details, specify `warn`, `debug`, or `error` instead of `info`.

**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**

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

**IMPORTANT**

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

7.1.13. Verifying cluster status

To verify your OpenShift Container Platform cluster's status during or after installation:

Procedure

- 1 In the cluster environment, export the administrator's kubeconfig file:

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

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

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.

- 2 View the control plane and compute machines created after a deployment:

```
$ oc get nodes
```

- 3 View your cluster's version:

```
$ oc get clusterversion
```

- 4 View your operators' status:

```
$ oc get clusteroperator
```

5. View all running Pods in the cluster:

```
$ oc get pods -A
```

7.1.14. Logging in to the cluster

You can log in to 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
```

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

2. Verify you can run **oc** commands successfully using the exported configuration:

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

7.1.15. Configuring application access with floating IP addresses

After you install OpenShift Container Platform, configure Red Hat OpenStack Platform (RHOSP) to allow application network traffic.

Prerequisites

- OpenShift Container Platform cluster must be installed
- Floating IP addresses are enabled as described in *Enabling access to the environment*.

Procedure

After you install the OpenShift Container Platform cluster, attach a floating IP address to the ingress port:

1. Show the port:

```
$ openstack port show <cluster name>-<clusterID>-ingress-port
```

2. Attach the port to the IP address:

```
$ openstack floating ip set --port <ingress port ID> <apps FIP>
```

3. Add a wildcard **A** record for ***apps**. to your DNS file:

```
*.apps.<cluster name>.<base domain> IN A <apps FIP>
```



NOTE

If you do not control the DNS server but want to enable application access for non-production purposes, you can add these hostnames to **/etc/hosts**:

```
<apps FIP> console-openshift-console.apps.<cluster name>.<base domain>
<apps FIP> integrated-oauth-server-openshift-authentication.apps.<cluster name>.
<base domain>
<apps FIP> oauth-openshift.apps.<cluster name>.<base domain>
<apps FIP> prometheus-k8s-openshift-monitoring.apps.<cluster name>.<base
domain>
<apps FIP> grafana-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> <app name>.apps.<cluster name>.<base domain>
```

Next steps

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

CHAPTER 8. INSTALLING ON VSphere

8.1. INSTALLING A CLUSTER ON VSphere

In OpenShift Container Platform version 4.2, 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 allow the sites](#) that your cluster requires access to.



NOTE

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

8.1.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.2, you require access to the internet to install and entitle your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager](#). From there, you can allocate entitlements to your cluster.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management and entitlement. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster. If the Telemetry service cannot entitle your cluster, you must manually entitle it on the [Cluster registration](#) page.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

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

8.1.2. 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 13 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.

8.1.3. 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.

8.1.3.1. 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.

8.1.3.2. 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.

8.1.3.3. 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
Control plane	RHCOS	4	16 GB	120 GB
Compute	RHCOS or RHEL 7.6	2	8 GB	120 GB

8.1.3.4. 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.

8.1.4. Creating the user-provisioned infrastructure

Before you deploy an 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.

8.1.4.1. 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 supported 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 8.1. All machines to all machines

Protocol	Port	Description
TCP	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
UDP	4789	VXLAN and GENEVE
	6081	VXLAN and GENEVE
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
	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

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.

Ethernet adaptor hardware address requirements

When provisioning VMs for the cluster, the ethernet interfaces configured for each VM must use a MAC address from the VMware Organizationally Unique Identifier (OUI) allocation ranges:

- **00:05:69:00:00:00** to **00:05:69:FF:FF:FF**
- **00:0c:29:00:00:00** to **00:0c:29:FF:FF:FF**
- **00:1c:14:00:00:00** to **00:1c:14:FF:FF:FF**
- **00:50:56:00:00:00** to **00:50:56:FF:FF:FF**

If a MAC address outside the VMware OUI is used, the cluster installation will not succeed.

8.1.4.2. User-provisioned DNS requirements

The following DNS records are required for an 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 8.2. Required DNS records

Component	Record	Description

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>	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.
Routes	*.apps.<cluster_name>.<base_domain>	<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>

Component	Record	Description
etcd	etcd-<index>.<cluster_name>.<base_domain>	<p>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.</p>
	_etcd-server-ssl._tcp.<cluster_name>.<base_domain>	<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>

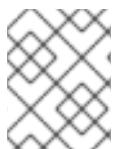
```
# _service._proto.name. TTL class SRV priority weight port target.
_etcd-server-ssl._tcp.<cluster_name>.<base_domain> 86400 IN SRV 0 10 2380 etcd-0.
```

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

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.

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 installation program. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

8.1.6. 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 500 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.

8.1.7. 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.

- 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**.

- 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.

8.1.7.1. 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

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 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.
- 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 on, specify an SSH key that your **ssh-agent** process uses.

8.1.7.2. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the Proxy object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The Proxy object's **status.noProxy** field is populated by default with the instance metadata endpoint (**169.254.169.254**) and with the values of the **networking.machineCIDR**, **networking.clusterNetwork.cidr**, and **networking.serviceNetwork** fields from your installation configuration.

Procedure

- 1 Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> ①
  httpsProxy: http://<username>:<pswd>@<ip>:<port> ②
  noProxy: example.com ③
  additionalTrustBundle: | ④
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
  ...
  ...
```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. The URL scheme must be **http**; **https** is currently not supported.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to include all subdomains of that domain. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a ConfigMap that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** ConfigMap that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this ConfigMap is referenced in the Proxy object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

**NOTE**

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster** Proxy object is still created, but it will have a nil **spec**.

**NOTE**

Only the Proxy object named **cluster** is supported, and no additional proxies can be created.

8.1.8. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and 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.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. 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

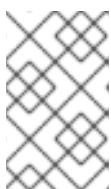
- ① For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

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

2. Modify the **manifests/cluster-scheduler-02-config.yaml** Kubernetes manifest file to prevent Pods from being scheduled on the control plane machines:
 - a. Open the **manifests/cluster-scheduler-02-config.yaml** file.

- b. Locate the **mastersSchedulable** parameter and set its value to **False**.

- c. Save and exit the file.



NOTE

Currently, due to a [Kubernetes limitation](#), router Pods running on control plane machines will not be reachable by the ingress load balancer. This step might not be required in a future minor version of OpenShift Container Platform.

3. 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:



8.1.9. 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": {}
}

```

- 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.

- 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

```

- 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 **rhcoss-<version>-<architecture>-vmware.ova**.

- In the vSphere Client, create a folder in your datacenter to store your VMs.
 - Click the **VMs and Templates** view.
 - Right-click the name of your datacenter.
 - 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**.
- Alternatively, prior to powering on the virtual machine add via vApp properties:
 - Navigate to a virtual machine from the vCenter Server inventory.
 - On the **Configure** tab, expand **Settings** and select **vApp options**.
 - Scroll down and under **Properties** apply the configurations from above.
- 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.

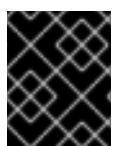


IMPORTANT

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.

8.1.10. Installing the CLI

You can install the CLI in order to interact with OpenShift Container Platform using a command-line interface.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.2. Download and install the new version of **oc**.

Procedure

1. From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**.
2. Click the folder for your operating system and architecture and click the compressed file.



NOTE

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

3. Save the file to your file system.
4. Extract the compressed file.
5. Place it in a directory that is on your **PATH**.

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

8.1.11. Creating the cluster

To create the OpenShift Container Platform cluster, you wait for the bootstrap process to complete on the machines that you provisioned 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. Monitor the bootstrap process:

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

INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.14.6+c4799753c 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 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

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.

8.1.12. Logging in to the cluster

You can log in to 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
```

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

2. Verify you can run **oc** commands successfully using the exported configuration:

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

8.1.13. 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.
- Install the **jq** package.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	63m	v1.14.6+c4799753c
master-1	Ready	master	63m	v1.14.6+c4799753c
master-2	Ready	master	64m	v1.14.6+c4799753c
worker-0	NotReady	worker	76s	v1.14.6+c4799753c
worker-1	NotReady	worker	70s	v1.14.6+c4799753c

The output lists all of the machines that you created.

2. Review the pending certificate signing requests (CSRs) and ensure that 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

8.1.14. 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

1. Watch the cluster components come online:

NAME SINCE	VERSION	AVAILABLE	PROGRESSING	DEGRADED
authentication	4.2.0	True	False	False 69s
cloud-credential	4.2.0	True	False	False 12m
cluster-autoscaler	4.2.0	True	False	False 11m
console	4.2.0	True	False	False 46s
dns	4.2.0	True	False	False 11m
image-registry	4.2.0	False	True	False 5m26s
ingress	4.2.0	True	False	False 5m36s
kube-apiserver	4.2.0	True	False	False 8m53s
kube-controller-manager	4.2.0	True	False	False 7m24s
kube-scheduler	4.2.0	True	False	False 12m
machine-api	4.2.0	True	False	False 12m
machine-config	4.2.0	True	False	False 7m36s
marketplace	4.2.0	True	False	False 7m54m
monitoring	4.2.0	True	False	False 7h54s
network	4.2.0	True	False	False 5m9s
node-tuning	4.2.0	True	False	False 11m
openshift-apiserver	4.2.0	True	False	False 11m
openshift-controller-manager	4.2.0	True	False	False 5m943s
openshift-samples	4.2.0	True	False	False 3m55s
operator-lifecycle-manager	4.2.0	True	False	False 11m
operator-lifecycle-manager-catalog	4.2.0	True	False	False 11m
service-ca	4.2.0	True	False	False 11m
service-catalog-apiserver	4.2.0	True	False	False 5m26s
service-catalog-controller-manager	4.2.0	True	False	False 5m25s
storage	4.2.0	True	False	False 5m30s

2. Configure the Operators that are not available.

8.1.14.1. 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.

8.1.14.1.1. 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**.

**IMPORTANT**

vSphere volumes do not support the **ReadWriteMany** access mode. You must use a different storage backend, such as **NFS**, to configure the registry storage.

- 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 cannot 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
```

3. 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.

4. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

8.1.14.1.2. 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.

8.1.15. 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 SINCE	VERSION	AVAILABLE	PROGRESSING	DEGRADED
authentication	4.2.0	True	False	False 10m
cloud-credential	4.2.0	True	False	False 22m
cluster-autoscaler	4.2.0	True	False	False 21m
console	4.2.0	True	False	False 10m
dns	4.2.0	True	False	False 21m
image-registry	4.2.0	True	False	False 16m
ingress	4.2.0	True	False	False 16m
kube-apiserver	4.2.0	True	False	False 19m
kube-controller-manager	4.2.0	True	False	False 18m
kube-scheduler	4.2.0	True	False	False 22m
machine-api	4.2.0	True	False	False 22m
machine-config	4.2.0	True	False	False 18m
marketplace	4.2.0	True	False	False 18m
monitoring	4.2.0	True	False	False 18m
network	4.2.0	True	False	False 16m
node-tuning	4.2.0	True	False	False 21m
openshift-apiserver	4.2.0	True	False	False 21m
openshift-controller-manager	4.2.0	True	False	False 17m
openshift-samples	4.2.0	True	False	False 14m
operator-lifecycle-manager	4.2.0	True	False	False 21m

operator-lifecycle-manager-catalog	4.2.0	True	False	False	21m
service-ca	4.2.0	True	False	False	21m
service-catalog-apiserver	4.2.0	True	False	False	16m
service-catalog-controller-manager	4.2.0	True	False	False	16m
storage	4.2.0	True	False	False	16m

When all of the cluster Operators are **AVAILABLE**, you can complete the installation.

- Monitor for cluster completion:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete ①
INFO Waiting up to 30m0s for the cluster to initialize...
```

- 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.



IMPORTANT

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

- Confirm that the Kubernetes API server is communicating with the Pods.

- 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			
...			

- 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> ①
```

- 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 remote health reporting](#).

CHAPTER 9. INSTALLING IN RESTRICTED NETWORKS

9.1. CREATING A MIRROR REGISTRY FOR INSTALLATION IN A RESTRICTED NETWORK

Before you install a cluster on infrastructure that you provision in a restricted network, you must create a mirror registry. Installations on a restricted network are supported on only infrastructure that you provision, not infrastructure that the installer provisions.



IMPORTANT

You must have access to the internet to obtain the data that populates the mirror repository. In this procedure, you place the mirror registry on a bastion host that has access to both your network and the internet. If you do not have access to a bastion host, use the method that best fits your restrictions to bring the contents of the mirror registry into your restricted network.

9.1.1. About the mirror registry

You can mirror the contents of the OpenShift Container Platform registry and the images that are required to generate the installation program.

The mirror registry is a key component that is required to complete an installation in a restricted network. You can create this mirror on a bastion host, which can access both the internet and your closed network, or by using other methods that meet your restrictions.

Because of the way that OpenShift Container Platform verifies integrity for the release payload, the image references in your local registry are identical to the ones that are hosted by Red Hat on [Quay.io](#). During the bootstrapping process of installation, the images must have the same digests no matter which repository they are pulled from. To ensure that the release payload is identical, you mirror the images to your local repository.

9.1.2. Preparing the bastion host

Before you create the mirror registry, you must prepare the bastion host.

9.1.2.1. Installing the CLI

You can install the CLI in order to interact with OpenShift Container Platform using a command-line interface.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.2. Download and install the new version of **oc**.

Procedure

1. From the [OpenShift Infrastructure Providers](#) page, click **Download Command-line Tools**.
2. Click the folder for your operating system and architecture and click the compressed file.

**NOTE**

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

3. Save the file to your file system.
4. Extract the compressed file.
5. Place it in a directory that is on your **PATH**.

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

9.1.3. Creating a mirror registry

Create a registry to host the mirrored content that you require for installing OpenShift Container Platform. For installation in a restricted network, you must place the mirror on your bastion host.

**NOTE**

The following procedure creates a simple registry that stores data in the **/opt/registry** folder and runs in a **podman** container. You can use a different registry solution, such as [Red Hat Quay](#). Review the following procedure to ensure that your registry functions correctly.

Prerequisites

- You have a Red Hat Enterprise Linux (RHEL) server on your network to use as the registry host.
- The registry host can access the internet.

Procedure

On the bastion host, take the following actions:

1. Install the required packages:

```
# yum -y install podman httpd-tools
```

The **podman** package provides the container package that you run the registry in. The **httpd-tools** package provides the **htpasswd** utility, which you use to create users.

2. Create folders for the registry:

```
# mkdir -p /opt/registry/{auth,certs,data}
```

These folders are mounted inside the registry container.

3. Provide a certificate for the registry. If you do not have an existing, trusted certificate authority, you can generate a self-signed certificate:

```
$ cd /opt/registry/certs
# openssl req -newkey rsa:4096 -nodes -sha256 -keyout domain.key -x509 -days 365 -out
domain.crt
```

At the prompts, provide the required values for the certificate:

Country Name (2 letter code)	Specify the two-letter ISO country code for your location. See the ISO 3166 country codes standard.
State or Province Name (full name)	Enter the full name of your state or province.
Locality Name (eg, city)	Enter the name of your city.
Organization Name (eg, company)	Enter your company name.
Organizational Unit Name (eg, section)	Enter your department name.
Common Name (eg, your name or your server's hostname)	Enter the host name for the registry host. Ensure that your hostname is in DNS and that it resolves to the expected IP address.
Email Address	Enter your email address. For more information, see the <code>req</code> description in the OpenSSL documentation.

4. Generate a user name and a password for your registry that uses the **bcrpt** format:

```
# htpasswd -bBc /opt/registry/auth/htpasswd <user_name> <password> ①
```

- 1 Replace **<user_name>** and **<password>** with a user name and a password.

5. Create the **mirror-registry** container to host your registry:

```
# podman run --name mirror-registry -p <local_registry_host_port>:5000 \
-v /opt/registry/data:/var/lib/registry:z \
-v /opt/registry/auth:/auth:z \ ①
```

```
-e "REGISTRY_AUTH=htpasswd" \
-e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" \
-e REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd \
-v /opt/registry/certs:/certs:z \
-e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt \
-e REGISTRY_HTTP_TLS_KEY=/certs/domain.key \
docker.io/library/registry:2
```

- 1** For **<local_registry_host_port>**, specify the port that your mirror registry uses to serve content.

6. Open the required ports for your registry:

```
# firewall-cmd --add-port=<local_registry_host_port>/tcp --zone=internal --permanent 1
# firewall-cmd --add-port=<local_registry_host_port>/tcp --zone=public --permanent 2
# firewall-cmd --reload
```

- 1** **2** For **<local_registry_host_port>**, specify the port that your mirror registry uses to serve content.

7. Add the self-signed certificate to your list of trusted certificates:

```
# cp /opt/registry/certs/domain.crt /etc/pki/ca-trust/source/anchors/
# update-ca-trust
```

You must trust your certificate to log in to your registry during the mirror process.

8. Confirm that the registry is available:

```
$ curl -u <user_name>:<password> -k https://<local_registry_host_name>:<local_registry_host_port>/v2/_catalog 1
{"repositories":[]}
```

- 1** For **<user_name>** and **<password>**, specify the user name and password for your registry. For **<local_registry_host_name>**, specify the registry domain name that you specified in your certificate, such as **registry.example.com**. For **<local_registry_host_port>**, specify the port that your mirror registry uses to serve content.

If the command output displays an empty repository, your registry is available.

9.1.4. Creating a pull secret for your mirror registry

In a restricted network, you create a pull secret that contains only the information for your registry.

Prerequisites

- You configured a mirror registry to use in your restricted network and have its domain name and port as well as credentials for it.

Procedure

- On the bastion host, generate the pull secret for your registry:

```
$ podman login --authfile ~/pullsecret_config.json <local_registry_host_name>:  
<local_registry_host_port> ①
```

- ① For **<local_registry_host_name>**, specify the registry domain name for your mirror registry, such as **registry.example.com**. For **<local_registry_host_port>**, specify the port that your mirror registry uses to serve content.

Provide your credentials for the mirror registry at the prompts.

- View the pull secret that you created:

```
# cat ~/pullsecret_config.json  
  
{ "auths": { "<local_registry_host_name>:<local_registry_host_port>": { "auth":  
"ZHVtbXk6ZHVTbXk=" } } }
```

9.1.5. Mirroring the OpenShift Container Platform image repository

Mirror the OpenShift Container Platform image repository to use during cluster installation or upgrade.

Prerequisites

- You configured a mirror registry to use in your restricted network and can access the certificate and credentials that you configured.
- You downloaded the pull secret from the [OpenShift Infrastructure Providers](#) page and modified it to include authentication to your mirror repository.

Procedure

Complete the following steps on the bastion host:

- Review the [OpenShift Container Platform downloads page](#) to determine the version of OpenShift Container Platform that you want to install.
- Set the required environment variables:

```
$ export OCP_RELEASE=<release_version> ①  
$ export LOCAL_REGISTRY='<local_registry_host_name>:<local_registry_host_port>' ②  
$ export LOCAL_REPOSITORY='<repository_name>' ③  
$ export PRODUCT_REPO='openshift-release-dev' ④  
$ export LOCAL_SECRET_JSON='<path_to_pull_secret>' ⑤  
$ export RELEASE_NAME="ocp-release" ⑥
```

- ① For **<release_version>**, specify the version number of OpenShift Container Platform to install, such as **4.2.0**.
- ② For **<local_registry_host_name>**, specify the registry domain name for your mirror repository, and for **<local_registry_host_port>**, specify the port that it serves content on.

- 3** For <repository_name>, specify the name of the repository to create in your registry, such as **ocp4/openshift4**.
- 4** The repository to mirror. For a production release, you must specify **openshift-release-dev**.
- 5** For <path_to_pull_secret>, specify the absolute path to and file name of the pull secret for your mirror registry that you created.
- 6** The release mirror. For a production release, you must specify **ocp-release**.

3. Mirror the repository:

```
$ oc adm -a ${LOCAL_SECRET_JSON} release mirror \
--from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE} \
--to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} \
--to-release-image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}
```

This command pulls the release information as a digest, and its output includes text that resembles the following sample:

```
...
Success
Update image: <local_registry_host_name>:
<local_registry_host_port>/<local_registry>/<local_repository>:<release_version>
Mirror prefix: <local_registry_host_name>:
<local_registry_host_port>/<local_registry>/<local_repository>
```

To use the new mirrored repository to install, add the following section to the `install-config.yaml`:

```
imageContentSources:
- mirrors:
  - <local_registry_host_name>:<local_registry_host_port>/<repository_name>/release
    source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry_host_name>:<local_registry_host_port>/<repository_name>/release
    source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
...
...
```

4. Record the **imageContentSources** section from the output of the previous command. This information is required during OpenShift Container Platform installation.
5. To create the installation program that is based on the content that you mirrored, extract it and pin it to the release:

```
$ oc adm release extract --command=openshift-install
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}"
```



IMPORTANT

To ensure that you use the correct images for the version of OpenShift Container Platform that you selected, you must extract the installation program from the mirrored content.

9.1.6. Using sample imagestreams in a restricted network installation

Most imagestreams in the OpenShift namespace managed by the Samples Operator point to images located in the Red Hat registry at registry.redhat.io. Mirroring will not apply to these imagestreams.

The **jenkins**, **jenkins-agent-maven**, and **jenkins-agent-nodejs** imagestreams do come from the install payload and are managed by the Samples Operator, so no further mirroring procedures are needed for those imagestreams.



NOTE

The **cli**, **installer**, **must-gather**, and **tests** imagestreams, while part of the install payload, are not managed by the Samples Operator. These are not addressed in this procedure.

Prerequisites

- Access to the cluster as a user with the **cluster-admin** role.
- Create a pull secret for your mirror registry.

Procedure

1. Mirror images from registry.redhat.io associated with any imagestreams you need in the restricted network environment into one of the defined mirrors:

```
$ oc image mirror myregistry.com/myimage:latest myregistry.com/myimage:stable
```

2. Add the required trusted CAs for the mirror in the cluster's image configuration object:

```
$ oc create configmap registry-config --from-file=$path/ca.crt -n openshift-config
$ oc patch image.config.openshift.io/cluster --patch '{"spec":{"additionalTrustedCA": {"name":"registry-config"}}}' --type=merge
```

3. Update the **samplesRegistry** field in the Samples Operator configuration object to contain the **hostname** portion of the mirror location defined in the mirror configuration:

```
$ oc get configs.samples.operator.openshift.io -n openshift-cluster-samples-operator
```



NOTE

This is required because the imagestream import process does not use the mirror or search mechanism at this time.

4. Add any imagestreams that are not mirrored into the **skippedImagestreams** field of the Samples Operator configuration object. Or if you do not want to support any of the sample imagestreams, set the Samples Operator to **Removed** in the Samples Operator configuration object.



NOTE

Any unmirrored imagestreams that are not skipped, or if the Samples Operator is not changed to **Removed**, will result in the Samples Operator reporting a **Degraded** status two hours after the imagestream imports start failing.

Many of the templates in the OpenShift namespace reference the imagestreams. So using **Removed** to purge both the imagestreams and templates will eliminate the possibility of attempts to use them if they are not functional because of any missing imagestreams.

Next steps

- Install a cluster on infrastructure that you provision in your restricted network, such as on [VMware vSphere](#), [bare metal](#), or [Amazon Web Services](#).

9.2. INSTALLING A CLUSTER ON AWS THAT USES MIRRORED INSTALLATION CONTENT

In OpenShift Container Platform version 4.2, you can install a cluster on Amazon Web Services (AWS) using infrastructure that you provide and an internal mirror of the installation release content.



IMPORTANT

While you can install an OpenShift Container Platform cluster by using mirrored installation release content, your cluster still requires internet access to use the AWS APIs.

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

- [Create a mirror registry on your bastion host](#) and obtain the **imageContentSources** data for your version of OpenShift Container Platform.



IMPORTANT

Because the installation media is on the bastion host, use that computer to complete all installation steps.

- 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 and plan to use telemetry, you must [configure it to allow the sites](#) that your cluster requires access to.



NOTE

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

9.2.1. About installations in restricted networks

In OpenShift Container Platform 4.2, you can perform an installation that does not require an active connection to the internet to obtain software components. You complete an installation in a restricted network on only infrastructure that you provision, not infrastructure that the installation program provisions, so your platform selection is limited.

If you choose to perform a restricted network installation on a cloud platform, you still require access to its cloud APIs. Some cloud functions, like Amazon Web Service’s IAM service, require internet access, so you might still require internet access. Depending on your network, you might require less internet access for an installation on bare metal hardware or on VMware vSphere.

To complete a restricted network installation, you must create a registry that mirrors the contents of the OpenShift Container Platform registry and contains the installation media. You can create this mirror on a bastion host, which can access both the internet and your closed network, or by using other methods that meet your restrictions.



IMPORTANT

Restricted network installations always use user-provisioned infrastructure. Because of the complexity of the configuration for user-provisioned installations, consider completing a standard user-provisioned infrastructure installation before you attempt a restricted network installation. Completing this test installation might make it easier to isolate and troubleshoot any issues that might arise during your installation in a restricted network.

9.2.1.1. Additional limits

Clusters in restricted networks have the following additional limitations and restrictions:

- The ClusterVersion status includes an **Unable to retrieve available updates** error.
- By default, you cannot use the contents of the Developer Catalog because you cannot access the required ImageStreamTags.

9.2.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.2, you require access to the internet to install and entitle your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager](#). From there, you can allocate entitlements to your cluster.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management and entitlement. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster. If the Telemetry

service cannot entitle your cluster, you must manually entitle it on the [Cluster registration](#) page.

- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

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

9.2.3. 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.

9.2.3.1. 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 with the provided CloudFormation templates.



IMPORTANT

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

Table 9.1. Instance types for machines

Instance type	Bootstrap	Control plane	Compute
i3.large	x		

Instance type	Bootstrap	Control plane	Compute
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
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

You might be able to use other instance types that meet the specifications of these instance types.

9.2.3.2. Other infrastructure components

- A VPC
- DNS entries

- Load balancers and listeners
- A public and a private 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 uses an endpoint that references the route tables for each subnet to improve communication with the registry that is hosted in S3.				
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 • AWS::EC2::SubnetRouteTableAssociation • AWS::EC2::NatGateway • AWS::EC2::EIP 	You must have a public internet gateway, with public routes, attached to the VPC. In the provided templates, each public subnet has a NAT gateway with an EIP address. These NAT gateways allow cluster resources, like private-subnet instances, to reach the internet and are not required for some restricted network or proxy scenarios.				
Network access control	<ul style="list-style-type: none"> • AWS::EC2::NetworkAcl • AWS::EC2::NetworkAclEntry 	<p>You must allow the VPC to access the following ports:</p> <table border="1"> <thead> <tr> <th>Port</th><th>Reason</th></tr> </thead> <tbody> <tr> <td>80</td><td>Inbound HTTP traffic</td></tr> </tbody> </table>	Port	Reason	80	Inbound HTTP traffic
Port	Reason					
80	Inbound HTTP traffic					

Component	AWS type	Description	
		443	Inbound HTTPS traffic
		22	Inbound SSH traffic
		1024 - 65535	Inbound ephemeral traffic
		0 - 65535	Outbound ephemeral traffic
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.

Component	AWS type	Description
Public load balancer	AWS::Elastic LoadBalancingV2::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::Elastic LoadBalancingV2::Listener	A listener on port 6443 for the external load balancer.
External target group	AWS::Elastic LoadBalancingV2::Target Group	The target group for the external load balancer.
Private load balancer	AWS::Elastic LoadBalancingV2::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::Elastic LoadBalancingV2::Listener	A listener on port 22623 for the internal load balancer.
Internal target group	AWS::Elastic LoadBalancingV2::Target Group	The target group for the Internal load balancer.
Internal listener	AWS::Elastic LoadBalancingV2::Listener	A listener on port 6443 for the internal load balancer.
Internal target group	AWS::Elastic LoadBalancingV2::Target Group	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::Security Group	icmp	0
		tcp	22
		tcp	6443
		tcp	22623
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

Ingress group	Description	IP protocol	Port range
MasterIngress WorkerKube	Kubernetes kubelet, scheduler and controller manager	tcp	10250 - 10259
MasterIngress IngressServices	Kubernetes Ingress services	tcp	30000 - 32767
MasterIngress WorkerIngress Services	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 WorkerIngress Services	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	*
Worker	Allow	ec2:Describe*	*
Bootstrap	Allow	ec2:Describe*	*
	Allow	ec2:AttachVolume	*
	Allow	ec2:DetachVolume	*

9.2.3.3. 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:DescribelImages**
- **ec2:DescribelInstanceAttribute**
- **ec2:DescribelInstanceCreditSpecifications**
- **ec2:DescribelInstances**
- **ec2:DescribelInternetGateways**
- **ec2:DescribelKeyPairs**
- **ec2:DescribelNatGateways**
- **ec2:DescribelNetworkAcls**
- **ec2:DescribelPrefixLists**
- **ec2:DescribelRegions**
- **ec2:DescribelRouteTables**
- **ec2:DescribelSecurityGroups**
- **ec2:DescribelSubnets**
- **ec2:DescribelTags**
- **ec2:DescribelVpcEndpoints**
- **ec2:DescribelVpcs**

- **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**

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.

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 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>
```

```
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 installation program. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

9.2.5. 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.

9.2.5.1. Creating the installation configuration file

Generate and customize the installation configuration file that the installation program needs to deploy your cluster.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster. For a restricted network installation, these files are on your bastion host.

Procedure

- Obtain the **install-config.yaml** file.

- Run the following command:

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

- At the prompts, provide the configuration details for your cloud:

- 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 on, specify an SSH key that your **ssh-agent** process uses.

- 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. Edit the **install-config.yaml** file to provide the additional information that is required for an installation in a restricted network.

- a. Update the **pullSecret** value to contain the authentication information for your registry:

```
pullSecret: '{"auths":{"<bastion_host_name>:5000": {"auth": "<credentials>","email": "you@example.com"}}}
```

For **bastion_host_name**, specify the registry domain name that you specified in the certificate for your mirror registry, and for **<credentials>**, specify the base64-encoded user name and password for your mirror registry.

- b. Add the **additionalTrustBundle** parameter and value. The value must be the contents of the certificate file that you used for your mirror registry, which can be an exiting, trusted certificate authority or the self-signed certificate that you generated for the mirror registry.

```
additionalTrustBundle: |
-----BEGIN CERTIFICATE-----
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
-----END CERTIFICATE-----
```

- c. Add the image content resources:

```
imageContentSources:
- mirrors:
  - <bastion_host_name>:5000/<repo_name>/release
    source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <bastion_host_name>:5000/<repo_name>/release
    source: registry.svc.ci.openshift.org/ocp/release
```

Use the **imageContentSources** section from the output of the command to mirror the repository.

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



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.

9.2.5.2. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the Proxy object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The Proxy object's **status.noProxy** field is populated by default with the instance metadata endpoint (**169.254.169.254**) and with the values of the **networking.machineCIDR**, **networking.clusterNetwork.cidr**, and **networking.serviceNetwork** fields from your installation configuration.

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> ①
  httpsProxy: http://<username>:<pswd>@<ip>:<port> ②
  noProxy: example.com ③
  additionalTrustBundle: | ④
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
  ...
  ...
```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. The URL scheme must be **http**; **https** is currently not supported.

- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with `.` to include all subdomains of
- 4 If provided, the installation program generates a ConfigMap that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** ConfigMap that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this ConfigMap is referenced in the Proxy object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.



NOTE

The installation program does not support the proxy **readinessEndpoints** field.

- 2 Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster** Proxy object is still created, but it will have a nil **spec**.

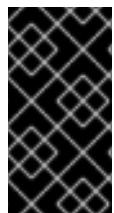


NOTE

Only the Proxy object named **cluster** is supported, and no additional proxies can be created.

9.2.5.3. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and 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. For a restricted network installation, these files are on your bastion host.
- Create the **install-config.yaml** installation configuration file.

Procedure

- 1 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 installation directory that contains the **install-config.yaml** file you created.

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

- 2 Remove the Kubernetes manifest files that define the control plane machines:

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

By removing these files, you prevent the cluster from automatically generating control plane machines.

- 3 Remove the Kubernetes manifest files that define the worker machines:

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

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

- 4 Modify the **manifests/cluster-scheduler-02-config.yml** Kubernetes manifest file to prevent Pods from being scheduled on the control plane machines:

- a. Open the **manifests/cluster-scheduler-02-config.yml** file.
- b. Locate the **mastersSchedulable** parameter and set its value to **False**.
- c. Save and exit the file.



NOTE

Currently, due to a [Kubernetes limitation](#), router Pods running on control plane machines will not be reachable by the ingress load balancer. This step might not be required in a future minor version of OpenShift Container Platform.

- 5 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

```

9.2.6. 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). The provided CloudFormation templates contain references to this infrastructure name, 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 and view 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.

9.2.7. 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

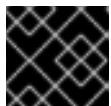
1. Create a JSON file that contains the parameter values that the template requires:

```
[  
  {  
    "ParameterKey": "VpcCidr", ①  
    "ParameterValue": "10.0.0.0/16" ②  
  },  
  {  
    "ParameterKey": "AvailabilityZoneCount", ③  
    "ParameterValue": "1" ④  
  },  
  {  
    "ParameterKey": "SubnetBits", ⑤  
    "ParameterValue": "12" ⑥  
  }  
]
```

- ① The CIDR block for the VPC.
- ② Specify a CIDR block in the format **x.x.x.x/16-24**.
- ③ The number of availability zones to deploy the VPC in.
- ④ Specify an integer between **1** and **3**.
- ⑤ The size of each subnet in each availability zone.
- ⑥ 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> ①  
--template-body file://<template>.yaml ②  
--parameters file://<parameters>.json ③
```

- ① **<name>** is the name for the CloudFormation stack, such as **cluster-vpc**. 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:

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

9.2.7.1. 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 [&gt;!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 [&gt;!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 [&gt;!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 [&gt;!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
InternetGateway:
  Type: "AWS::EC2::InternetGateway"
GatewayToInternet:
  Type: "AWS::EC2::VPCGatewayAttachment"

```

```

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
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"]]
]

```

9.2.8. 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 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>" ⑯
}
]

```

- 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 has 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.
- 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.
ApiServerDnsName	Full host name of the API server.
RegisterNlbIpTargetsLambda	Lambda ARN useful to help register/deregister IP targets for these load balancers.
ExternalApiTargetGroupArn	ARN of external API target group.
InternalApiTargetGroupArn	ARN of internal API target group.
InternalServiceTargetGroupArn	ARN of internal service target group.

9.2.8.1. 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 ["-", [&gt;!Ref InfrastructureName, "ext"]]
  IpAddressType: ipv4
  Subnets: !Ref PublicSubnets
  Type: network

IntApiElb:
Type: AWS::ElasticLoadBalancingV2::LoadBalancer
Properties:
  Name: !Join ["-", [&gt;!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 [".", [&gt;!Ref ClusterName, !Ref HostedZoneName]]
  HostedZoneTags:
- Key: Name
  Value: !Join ["-", [&gt;!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:
      - "RegisterTargetLambdalamRole"
      - "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

```

RegisterSubnetTagsLambdalamRole:

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]]

RegisterNlbIpTargetsLambda:

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

Value: !GetAtt RegisterNlbIpTargets.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
```

9.2.9. 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

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

```
[
  {
    "ParameterKey": "InfrastructureName", ①
    "ParameterValue": "mycluster-<random_string>" ②
  },
  {
    "ParameterKey": "VpcCidr", ③
    "ParameterValue": "10.0.0.0/16" ④
  },
  {
    "ParameterKey": "PrivateSubnets", ⑤
    "ParameterValue": "subnet-<random_string>" ⑥
  },
  {
    "ParameterKey": "VpcId", ⑦
    "ParameterValue": "vpc-<random_string>" ⑧
  }
]
```

- The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.

- 3 The CIDR block for the VPC.
- 4 Specify the CIDR block parameter that you used for the VPC that you defined in the form **x.x.x.x/16-24**.
- 5 The private subnets that you created for your VPC.
- 6 Specify the **PrivateSubnetIds** value from the output of the CloudFormation template for the VPC.
- 7 The VPC that you created for the cluster.
- 8 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> 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-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:

MasterSecurityGroup	Master Security Group ID
----------------------------	--------------------------

WorkerSecurityGroup	Worker Security Group ID
MasterInstanceProfile	Master IAM Instance Profile
WorkerInstanceProfile	Worker IAM Instance Profile

9.2.9.1. 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**
 IpProtocol: tcp

WorkerIngressWorkerIngressServices:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
 Description: Kubernetes ingress services
 FromPort: **30000**
 ToPort: **32767**
 IpProtocol: 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`

9.2.10. 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 9.2. RHCOS AMIs

AWS zone	AWS AMI
<code>ap-northeast-1</code>	<code>ami-0426ca3481a088c7b</code>
<code>ap-northeast-2</code>	<code>ami-014514ae47679721b</code>
<code>ap-south-1</code>	<code>ami-0bd772ba746948d9a</code>
<code>ap-southeast-1</code>	<code>ami-0d76ac0ebaac29e40</code>
<code>ap-southeast-2</code>	<code>ami-0391e92574fb09e08</code>
<code>ca-central-1</code>	<code>ami-04419691da69850cf</code>
<code>eu-central-1</code>	<code>ami-092b69120ecf915ed</code>
<code>eu-west-1</code>	<code>ami-04370efd78434697b</code>
<code>eu-west-2</code>	<code>ami-00c74e593125e0096</code>

AWS zone	AWS AMI
eu-west-3	ami-058ad17da14ff4d0d
sa-east-1	ami-03f6b71e93e630dab
us-east-1	ami-01e7fdcb66157b224
us-east-2	ami-0bc59aaa7363b805d
us-west-1	ami-0ba912f53c1fdcdf0
us-west-2	ami-08e10b201e19fd5e7

9.2.11. 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

- 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.

- Create the bucket:

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

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

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

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

- Verify that the file uploaded:

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

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

```
[  
  {  
    "ParameterKey": "InfrastructureName", ①  
    "ParameterValue": "mycluster-<random_string>" ②  
  },  
  {  
    "ParameterKey": "RhcossAmi", ③  
    "ParameterValue": "ami-<random_string>" ④  
  },  
  {  
    "ParameterKey": "AllowedBootstrapSshCidr", ⑤  
    "ParameterValue": "0.0.0.0/0" ⑥  
  },  
  {  
    "ParameterKey": "PublicSubnet", ⑦  
    "ParameterValue": "subnet-<random_string>" ⑧  
  },  
  {  
    "ParameterKey": "MasterSecurityGroupId", ⑨  
    "ParameterValue": "sg-<random_string>" ⑩  
  },  
  {  
    "ParameterKey": "VpcId", ⑪  
    "ParameterValue": "vpc-<random_string>" ⑫  
  },  
]
```

```
{
  "ParameterKey": "BootstrapIgnitionLocation", ⑬
  "ParameterValue": "s3://<bucket_name>/bootstrap.ign" ⑭
},
{
  "ParameterKey": "AutoRegisterELB", ⑮
  "ParameterValue": "yes" ⑯
},
{
  "ParameterKey": "RegisterNLBIPTargetsLambdaArn", ⑰
  "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:<dns_stack_name>-RegisterNLBIPTargets-<random_string>" ⑱
},
{
  "ParameterKey": "ExternalApiTargetGroupArn", ⑲
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" ⑳
},
{
  "ParameterKey": "InternalApiTargetGroupArn", ㉑
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" ㉒
},
{
  "ParameterKey": "InternalServiceTargetGroupArn", ㉓
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" ㉔
}
]
```

- 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 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 **RegisterNLBIPTargetsLambda** 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 machine** 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> 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-bootstrap**. 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:

BootstrapInstanceld	The bootstrap Instance ID.
BootstrapPublicip	The bootstrap node public IP address.
BootstrapPrivateip	The bootstrap node private IP address.

9.2.11.1. 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])(/[0-9]{1,3})\$

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
 - RegisterNLBIPTargetsLambdaArn
 - ExternalAPITargetGroupArn
 - InternalAPITargetGroupArn
 - InternalServiceTargetGroupArn

ParameterLabels:
 InfrastructureName:
 default: "Infrastructure Name"
 VpcId:

```

default: "VPC ID"
AllowedBootstrapSshCidr:
  default: "Allowed SSH Source"
PublicSubnet:
  default: "Public Subnet"
RhcoseAmi:
  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:

```

BootstraplamRole:
  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: "BootstraplamRole"

```

```

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 "MasterSecurityGroupId"
      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
        }
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
    TargetArn: !Ref ExternalApiTargetGroupArn
    TargetIp: !GetAtt BootstrapInstance.PrivateIp

RegisterBootstrapApiTarget:
Condition: DoRegistration
Type: Custom::NLBRegister
Properties:
  ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
  TargetArn: !Ref InternalApiTargetGroupArn
  TargetIp: !GetAtt BootstrapInstance.PrivateIp

RegisterBootstrapInternalServiceTarget:
Condition: DoRegistration
Type: Custom::NLBRegister
Properties:

```

```

ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
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

9.2.12. 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": "RhcossAmi", 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" ⑳
},
{
  "ParameterKey": "CertificateAuthorities", ㉑
  "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz==" ㉒
},
{
  "ParameterKey": "MasterInstanceProfileName", ㉓
  "ParameterValue": "<roles_stack>-MasterInstanceProfile-<random_string>" ㉔
},
{
  "ParameterKey": "MasterInstanceType", ㉕
  "ParameterValue": "m4.xlarge" ㉖
},
{
  "ParameterKey": "AutoRegisterELB", ㉗
  "ParameterValue": "yes" ㉘
},
{
  "ParameterKey": "RegisterNlbIpTargetsLambdaArn", ㉙
}

```

```

    "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNLbIpTargets-<random_string>" 30
},
{
  "ParameterKey": "ExternalApiTargetGroupArn", 31
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 32
},
{
  "ParameterKey": "InternalApiTargetGroupArn", 33
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 34
},
{
  "ParameterKey": "InternalServiceTargetGroupArn", 35
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 36
}
]

```

- 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 **RegisterNLBIPTargetsLambda** 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> ①
--template-body file://<template>.yaml ②
--parameters file://<parameters>.json ③
```

- ① **<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.
- 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>
```

9.2.12.1. 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
RhcossAmi:
  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 Z21IXYZABCZ2A4.  

  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
RegisterNLBIPTargetsLambdaArn:
  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
- RegisterNLbIpTargetsLambdaArn
- 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"
RhcossAmi:
  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 RegisterNlbIpTargetsLambdaArn

TargetArn: !Ref ExternalApiTargetGroupArn

TargetIp: !GetAtt Master0.PrivateIp

RegisterMaster0InternalApiTarget:

Condition: DoRegistration

Type: Custom::NLBRegister

Properties:

ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn

TargetArn: !Ref InternalApiTargetGroupArn

TargetIp: !GetAtt Master0.PrivateIp

RegisterMaster0InternalServiceTarget:

Condition: DoRegistration

Type: Custom::NLBRegister

Properties:

```

ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
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 RegisterNlbIpTargetsLambdaArn
TargetArn: !Ref ExternalApiTargetGroupArn
TargetIp: !GetAtt Master1.PrivateIp

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

RegisterMaster1InternalServiceTarget:
Condition: DoRegistration
Type: Custom::NLBRegister
Properties:
ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
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 "MasterSecurityGroupId"
    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 RegisterNlbIpTargetsLambdaArn
    TargetArn: !Ref ExternalApiTargetGroupArn
    TargetIp: !GetAtt Master2.PrivateIp

RegisterMaster2InternalApiTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
    TargetArn: !Ref InternalApiTargetGroupArn
    TargetIp: !GetAtt Master2.PrivateIp

RegisterMaster2InternalServiceTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
    TargetArn: !Ref InternalServiceTargetGroupArn
    TargetIp: !GetAtt Master2.PrivateIp

```

EtcSrvRecords:

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

Etc0Record:

Condition: DoDns

Type: AWS::Route53::RecordSet

Properties:

HostedZoneId: !Ref PrivateHostedZoneId

Name: !Join [".", ["etcd-0", !Ref PrivateHostedZoneName]]

ResourceRecords:

- !GetAtt Master0.PrivateIp

TTL: 60

Type: A

Etc1Record:

Condition: DoDns

Type: AWS::Route53::RecordSet

Properties:

HostedZoneId: !Ref PrivateHostedZoneId

Name: !Join [".", ["etcd-1", !Ref PrivateHostedZoneName]]

ResourceRecords:

- !GetAtt Master1.PrivateIp

TTL: 60

Type: A

Etc2Record:

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]
]
```

9.2.13. 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=info ②
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

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

9.2.13.1. 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": "RhcossAmi", ③  
    "ParameterValue": "ami-<random_string>" ④  
  },  
  {  
    "ParameterKey": "Subnet", ⑤  
    "ParameterValue": "subnet-<random_string>" ⑥  
  },  
  {  
    "ParameterKey": "WorkerSecurityGroupId", 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 **WorkerSecurityGroupId** 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**](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.

9.2.13.1.1. 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
    - RhcossAmi
    - 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

9.2.14. Logging in to the cluster

You can log in to 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
```

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

2. Verify you can run **oc** commands successfully using the exported configuration:

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

9.2.15. 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.
- Install the **jq** package.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes

NAME     STATUS   ROLES   AGE    VERSION
master-0  Ready    master   63m   v1.14.6+c4799753c
master-1  Ready    master   63m   v1.14.6+c4799753c
master-2  Ready    master   64m   v1.14.6+c4799753c
worker-0  NotReady worker   76s   v1.14.6+c4799753c
worker-1  NotReady worker   70s   v1.14.6+c4799753c
```

The output lists all of the machines that you created.

2. Review the pending certificate signing requests (CSRs) and ensure that 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.

3. 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
```

9.2.16. 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	4.2.0	True	False	False	69s	
cloud-credential	4.2.0	True	False	False	12m	
cluster-autoscaler	4.2.0	True	False	False	11m	
console	4.2.0	True	False	False	46s	
dns	4.2.0	True	False	False	11m	
image-registry	4.2.0	False	True	False	5m26s	
ingress	4.2.0	True	False	False	5m36s	
kube-apiserver	4.2.0	True	False	False	8m53s	
kube-controller-manager	4.2.0	True	False	False	7m24s	
kube-scheduler	4.2.0	True	False	False	12m	
machine-api	4.2.0	True	False	False	12m	
machine-config	4.2.0	True	False	False	7m36s	
marketplace	4.2.0	True	False	False	7m54m	
monitoring	4.2.0	True	False	False	7h54s	
network	4.2.0	True	False	False	5m9s	
node-tuning	4.2.0	True	False	False	11m	
openshift-apiserver	4.2.0	True	False	False	11m	
openshift-controller-manager	4.2.0	True	False	False	5m943s	
openshift-samples	4.2.0	True	False	False	3m55s	
operator-lifecycle-manager	4.2.0	True	False	False	11m	
operator-lifecycle-manager-catalog	4.2.0	True	False	False	11m	
service-ca	4.2.0	True	False	False	11m	
service-catalog-apiserver	4.2.0	True	False	False	5m26s	
service-catalog-controller-manager	4.2.0	True	False	False	5m25s	
storage	4.2.0	True	False	False	5m30s	

- Configure the Operators that are not available.

9.2.16.1. 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.

9.2.16.1.1. 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 an S3 bucket and configure storage with the following procedure.

Prerequisites

- A cluster on AWS with user-provisioned infrastructure.
- For S3 on AWS storage the secret is expected to contain two keys:
 - REGISTRY_STORAGE_S3_ACCESSKEY**
 - REGISTRY_STORAGE_S3_SECRETKEY**

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.

9.2.16.1.2. 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.

9.2.17. 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, and wait for installation to complete.

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.



IMPORTANT

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

3. Register your cluster on the [Cluster registration](#) page.

Next steps

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

9.3. INSTALLING A CLUSTER ON BARE METAL IN A RESTRICTED NETWORK

In OpenShift Container Platform version 4.2, you can install a cluster on bare metal infrastructure that you provision in a restricted network.



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

- [Create a mirror registry on your bastion host](#) and obtain the **imageContentSources** data for your version of OpenShift Container Platform.



IMPORTANT

Because the installation media is on the bastion host, use that computer to complete all installation steps.

- 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 and plan to use telemetry, you must [configure it to allow the sites](#) that your cluster requires access to.



NOTE

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

9.3.1. About installations in restricted networks

In OpenShift Container Platform 4.2, you can perform an installation that does not require an active connection to the internet to obtain software components. You complete an installation in a restricted network on only infrastructure that you provision, not infrastructure that the installation program provisions, so your platform selection is limited.

If you choose to perform a restricted network installation on a cloud platform, you still require access to its cloud APIs. Some cloud functions, like Amazon Web Service's IAM service, require internet access, so you might still require internet access. Depending on your network, you might require less internet access for an installation on bare metal hardware or on VMware vSphere.

To complete a restricted network installation, you must create a registry that mirrors the contents of the OpenShift Container Platform registry and contains the installation media. You can create this mirror on a bastion host, which can access both the internet and your closed network, or by using other methods that meet your restrictions.



IMPORTANT

Restricted network installations always use user-provisioned infrastructure. Because of the complexity of the configuration for user-provisioned installations, consider completing a standard user-provisioned infrastructure installation before you attempt a restricted network installation. Completing this test installation might make it easier to isolate and troubleshoot any issues that might arise during your installation in a restricted network.

9.3.1.1. Additional limits

Clusters in restricted networks have the following additional limitations and restrictions:

- The ClusterVersion status includes an **Unable to retrieve available updates** error.
- By default, you cannot use the contents of the Developer Catalog because you cannot access the required ImageStreamTags.

9.3.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.2, you require access to the internet to install and entitle your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager](#). From there, you can allocate entitlements to your cluster.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management and entitlement. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster. If the Telemetry service cannot entitle your cluster, you must manually entitle it on the [Cluster registration](#) page.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

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

9.3.3. 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.

9.3.3.1. 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.

9.3.3.2. 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.

9.3.3.3. 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
Control plane	RHCOS	4	16 GB	120 GB
Compute	RHCOS or RHEL 7.6	2	8 GB	120 GB

9.3.3.4. 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.

9.3.4. Creating the user-provisioned infrastructure

Before you deploy an 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.

9.3.4.1. 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 supported 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 9.3. All machines to all machines

Protocol	Port	Description
TCP	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
UDP	4789	VXLAN and GENEVE

Protocol	Port	Description
	6081	VXLAN and GENEVE
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
	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.

9.3.4.2. User-provisioned DNS requirements

The following DNS records are required for an 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 9.4. 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>	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.
Routes	*.apps.<cluster_name>.<base_domain>	<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> <p>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.</p>

Component	Record	Description
etcd	etcd-<index>.<cluster_name>.<base_domain>	<p>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.</p>
	_etcd-server-ssl._tcp.<cluster_name>.<base_domain>	<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>
_etcdb-server-ssl._tcp.<cluster_name>.<base_domain> 86400 IN SRV 0 10 2380 etcd-1.
<cluster_name>.<base_domain>
_etcdb-server-ssl._tcp.<cluster_name>.<base_domain> 86400 IN SRV 0 10 2380 etcd-2.
<cluster_name>.<base_domain>.
```

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.

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 installation program. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

9.3.6. 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.
- Obtain the **imageContentSources** section from the output of the command to mirror the repository.
- Obtain the contents of the certificate for your mirror registry.

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**.

- Unless you use a registry that RHCOS trusts by default, such as **docker.io**, you must provide the contents of the certificate for your mirror repository in the **additionalTrustBundle** section. In most cases, you must provide the certificate for your mirror.
- You must include the **imageContentSources** section from the output of the command to mirror the repository.

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.

9.3.6.1. 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:
  clusterNetwork:
  - cidr: 10.128.0.0/14 10
    hostPrefix: 23 11
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  none: {} 13
pullSecret: '{"auths":{"<bastion_host_name>:5000": {"auth": "<credentials>","email": "you@example.com"}}}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
additionalTrustBundle: | 16
-----BEGIN CERTIFICATE-----
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
-----END CERTIFICATE-----
imageContentSources: 17
- mirrors:
  - <bastion_host_name>:5000/<repo_name>/release
    source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <bastion_host_name>:5000/<repo_name>/release
    source: registry.svc.ci.openshift.org/ocp/release

```

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 ($2^{32-23} - 2$) pod IPs addresses. If you are required to provide access to nodes from an external network, configure load balancers and routers to manage the traffic.
- 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 For **bastion_host_name**, specify the registry domain name that you specified in the certificate for your mirror registry, and for **<credentials>**, specify the base64-encoded user name and password for your mirror registry.
- 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 on, specify an SSH key that your **ssh-agent** process uses.

- 16 Provide the contents of the certificate file that you used for your mirror registry.
- 17 Provide the **imageContentSources** section from the output of the command to mirror the repository.

9.3.6.2. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the Proxy object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The Proxy object's **status.noProxy** field is populated by default with the instance metadata endpoint (**169.254.169.254**) and with the values of the **networking.machineCIDR**, **networking.clusterNetwork.cidr**, and **networking.serviceNetwork** fields from your installation configuration.

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> ①
  httpsProxy: http://<username>:<pswd>@<ip>:<port> ②
  noProxy: example.com ③
  additionalTrustBundle: | ④
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
  ...
  ...
```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. The URL scheme must be **http**; **https** is currently not supported.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to include all subdomains of that domain. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a ConfigMap that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** ConfigMap that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this ConfigMap is referenced in the Proxy object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

**NOTE**

The installation program does not support the proxy **readinessEndpoints** field.

- Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster** Proxy object is still created, but it will have a nil **spec**.

**NOTE**

Only the Proxy object named **cluster** is supported, and no additional proxies can be created.

9.3.7. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and 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. For a restricted network installation, these files are on your bastion host.
- Create the **install-config.yaml** installation configuration file.

Procedure

- 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

- For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

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

- Modify the **manifests/cluster-scheduler-02-config.yml** Kubernetes manifest file to prevent Pods from being scheduled on the control plane machines:

- Open the **manifests/cluster-scheduler-02-config.yml** file.

- b. Locate the **mastersSchedulable** parameter and set its value to **False**.

- c. Save and exit the file.



NOTE

Currently, due to a [Kubernetes limitation](#), router Pods running on control plane machines will not be reachable by the ingress load balancer. This step might not be required in a future minor version of OpenShift Container Platform.

3. 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:



9.3.8. 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.

9.3.8.1. 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: **rhcoss-<version>-<architecture>-installer.iso**
- Compressed metal BIOS: **rhcoss-<version>-<architecture>-metal-bios.raw.gz**
- Compressed metal UEFI: **rhcoss-<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.

9.3.8.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE or iPXE booting

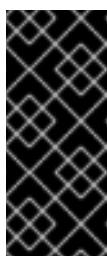
Before you install a cluster on bare metal infrastructure that you provision, you must create RHCOS machines for it to use. You can use PXE or iPXE booting to create the machines.

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: **rhcoss-<version>-<architecture>-installer.iso**
 - Compressed metal BIOS: **rhcoss-<version>-<architecture>-metal-bios.raw.gz**
 - **kernel**: **rhcoss-<version>-<architecture>-installer-kernel**
 - **initramfs**: **rhcoss-<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>/rhcoss-<version>-<architecture>-installer-kernel ①
APPEND ip=dhcp rd.neednet=1 initrd=http://<HTTP_server>/rhcoss-<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 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.

9.3.9. Creating the cluster

To create the OpenShift Container Platform cluster, you wait for the bootstrap process to complete on the machines that you provisioned 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. Monitor the bootstrap process:

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

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.14.6+c4799753c 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 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

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.

9.3.10. Logging in to the cluster

You can log in to 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
```

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

2. Verify you can run **oc** commands successfully using the exported configuration:

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

9.3.11. 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.
- Install the **jq** package.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	63m	v1.14.6+c4799753c
master-1	Ready	master	63m	v1.14.6+c4799753c
master-2	Ready	master	64m	v1.14.6+c4799753c
worker-0	NotReady	worker	76s	v1.14.6+c4799753c
worker-1	NotReady	worker	70s	v1.14.6+c4799753c

The output lists all of the machines that you created.

- Review the pending certificate signing requests (CSRs) and ensure that 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-8vnpn  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
```

9.3.12. 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 SINCE	VERSION	AVAILABLE	PROGRESSING	DEGRADED
authentication	4.2.0	True	False	False 69s
cloud-credential	4.2.0	True	False	False 12m
cluster-autoscaler	4.2.0	True	False	False 11m
console	4.2.0	True	False	False 46s
dns	4.2.0	True	False	False 11m
image-registry	4.2.0	False	True	False 5m26s
ingress	4.2.0	True	False	False 5m36s
kube-apiserver	4.2.0	True	False	False 8m53s
kube-controller-manager	4.2.0	True	False	False 7m24s
kube-scheduler	4.2.0	True	False	False 12m
machine-api	4.2.0	True	False	False 12m
machine-config	4.2.0	True	False	False 7m36s
marketplace	4.2.0	True	False	False 7m54m
monitoring	4.2.0	True	False	False 7h54s
network	4.2.0	True	False	False 5m9s
node-tuning	4.2.0	True	False	False 11m
openshift-apiserver	4.2.0	True	False	False 11m
openshift-controller-manager	4.2.0	True	False	False 5m943s
openshift-samples	4.2.0	True	False	False 3m55s
operator-lifecycle-manager	4.2.0	True	False	False 11m
operator-lifecycle-manager-catalog	4.2.0	True	False	False 11m
service-ca	4.2.0	True	False	False 11m
service-catalog-apiserver	4.2.0	True	False	False 5m26s
service-catalog-controller-manager	4.2.0	True	False	False 5m25s
storage	4.2.0	True	False	False 5m30s

2. Configure the Operators that are not available.

9.3.12.1. 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.

9.3.12.1.1. 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 cannot 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
```

3. 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.

4. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

9.3.12.1.2. 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.

9.3.13. 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 SINCE	VERSION	AVAILABLE	PROGRESSING	DEGRADED
authentication	4.2.0	True	False	False 10m
cloud-credential	4.2.0	True	False	False 22m
cluster-autoscaler	4.2.0	True	False	False 21m
console	4.2.0	True	False	False 10m
dns	4.2.0	True	False	False 21m
image-registry	4.2.0	True	False	False 16m
ingress	4.2.0	True	False	False 16m
kube-apiserver	4.2.0	True	False	False 19m
kube-controller-manager	4.2.0	True	False	False 18m
kube-scheduler	4.2.0	True	False	False 22m
machine-api	4.2.0	True	False	False 22m
machine-config	4.2.0	True	False	False 18m
marketplace	4.2.0	True	False	False 18m
monitoring	4.2.0	True	False	False 18m
network	4.2.0	True	False	False 16m
node-tuning	4.2.0	True	False	False 21m
openshift-apiserver	4.2.0	True	False	False 21m
openshift-controller-manager	4.2.0	True	False	False 17m
openshift-samples	4.2.0	True	False	False 14m
operator-lifecycle-manager	4.2.0	True	False	False 21m

operator-lifecycle-manager-catalog	4.2.0	True	False	False	21m
service-ca	4.2.0	True	False	False	21m
service-catalog-apiserver	4.2.0	True	False	False	16m
service-catalog-controller-manager	4.2.0	True	False	False	16m
storage	4.2.0	True	False	False	16m

When all of the cluster Operators are **AVAILABLE**, you can complete the installation.

- Monitor for cluster completion:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete ①
INFO Waiting up to 30m0s for the cluster to initialize...
```

- 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.



IMPORTANT

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

- Confirm that the Kubernetes API server is communicating with the Pods.

- 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			
...			

- 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> ①
```

- 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.

4. Register your cluster on the [Cluster registration](#) page.

Next steps

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

9.4. INSTALLING A CLUSTER ON VSphere IN A RESTRICTED NETWORK

In OpenShift Container Platform version 4.2, you can install a cluster on VMware vSphere infrastructure that you provision in a restricted network.

Prerequisites

- [Create a mirror registry on your bastion host](#) and obtain the **imageContentSources** data for your version of OpenShift Container Platform.



IMPORTANT

Because the installation media is on the bastion host, use that computer to complete all installation steps.

- 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 and plan to use telemetry, you must [configure it to allow the sites](#) that your cluster requires access to.



NOTE

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

9.4.1. About installations in restricted networks

In OpenShift Container Platform 4.2, you can perform an installation that does not require an active connection to the internet to obtain software components. You complete an installation in a restricted network on only infrastructure that you provision, not infrastructure that the installation program provisions, so your platform selection is limited.

If you choose to perform a restricted network installation on a cloud platform, you still require access to its cloud APIs. Some cloud functions, like Amazon Web Service's IAM service, require internet access, so you might still require internet access. Depending on your network, you might require less internet access for an installation on bare metal hardware or on VMware vSphere.

To complete a restricted network installation, you must create a registry that mirrors the contents of the OpenShift Container Platform registry and contains the installation media. You can create this mirror on a bastion host, which can access both the internet and your closed network, or by using other methods

that meet your restrictions.



IMPORTANT

Restricted network installations always use user-provisioned infrastructure. Because of the complexity of the configuration for user-provisioned installations, consider completing a standard user-provisioned infrastructure installation before you attempt a restricted network installation. Completing this test installation might make it easier to isolate and troubleshoot any issues that might arise during your installation in a restricted network.

9.4.1.1. Additional limits

Clusters in restricted networks have the following additional limitations and restrictions:

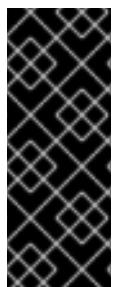
- The ClusterVersion status includes an **Unable to retrieve available updates** error.
- By default, you cannot use the contents of the Developer Catalog because you cannot access the required ImageStreamTags.

9.4.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.2, you require access to the internet to install and entitle your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager](#). From there, you can allocate entitlements to your cluster.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management and entitlement. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster. If the Telemetry service cannot entitle your cluster, you must manually entitle it on the [Cluster registration](#) page.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

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

9.4.3. 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 13 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.

9.4.4. 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.

9.4.4.1. 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.

9.4.4.2. 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.

9.4.4.3. 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

9.4.4.4. 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.

9.4.5. Creating the user-provisioned infrastructure

Before you deploy an 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.

9.4.5.1. 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 supported 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 9.5. All machines to all machines

Protocol	Port	Description
TCP	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
UDP	4789	VXLAN and GENEVE
	6081	VXLAN and GENEVE
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
	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

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.

9.4.5.2. User-provisioned DNS requirements

The following DNS records are required for an 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 9.6. Required DNS records

Component	Record	Description
Kubernetes API	<code>api.<cluster_name>.<base_domain></code>	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.

Component	Record	Description
	<code>api-int.<cluster_name>.<base_domain></code>	<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> <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>
Routes	<code>*.apps.<cluster_name>.<base_domain></code>	<p>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.</p>
etcd	<code>etcd-<index>.<cluster_name>.<base_domain></code>	<p>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 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.</p>

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

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.

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 installation program. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

9.4.7. 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.
- Obtain the **imageContentSources** section from the output of the command to mirror the repository.
- Obtain the contents of the certificate for your mirror registry.

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**.

- Unless you use a registry that RHCOS trusts by default, such as **docker.io**, you must provide the contents of the certificate for your mirror repository in the **additionalTrustBundle** section. In most cases, you must provide the certificate for your mirror.
- You must include the **imageContentSources** section from the output of the command to mirror the repository.

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.

9.4.7.1. 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":{"<bastion_host_name>:5000": {"auth": "<credentials>","email": "you@example.com"}}}' 14
  sshKey: 'ssh-ed25519 AAAA...' 15
  additionalTrustBundle: | 16
    -----BEGIN CERTIFICATE-----
    ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
    -----END CERTIFICATE-----
imageContentSources: 17
- mirrors:
  - <bastion_host_name>:5000/<repo_name>/release
    source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <bastion_host_name>:5000/<repo_name>/release
    source: registry.svc.ci.openshift.org/ocp/release

```

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 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.
- 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** For **bastion_host_name**, specify the registry domain name that you specified in the certificate for your mirror registry, and for **<credentials>**, specify the base64-encoded user name and password for your mirror registry.
- 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 on, specify an SSH key that your **ssh-agent** process uses.

- 16** Provide the contents of the certificate file that you used for your mirror registry.
- 17** Provide the **imageContentSources** section from the output of the command to mirror the repository.

9.4.7.2. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the Proxy object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The Proxy object's **status.noProxy** field is populated by default with the instance metadata endpoint (**169.254.169.254**) and with the values of the **networking.machineCIDR**, **networking.clusterNetwork.cidr**, and **networking.serviceNetwork** fields from your installation configuration.

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> ①
  httpsProxy: http://<username>:<pswd>@<ip>:<port> ②
  noProxy: example.com ③
  additionalTrustBundle: | ④
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
  ...
  ...
```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. The URL scheme must be **http**; **https** is currently not supported.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to include all subdomains of that domain. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a ConfigMap that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** ConfigMap that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this ConfigMap is referenced in the Proxy object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.



NOTE

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster** Proxy object is still created, but it will have a nil **spec**.



NOTE

Only the Proxy object named **cluster** is supported, and no additional proxies can be created.

9.4.8. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and 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. For a restricted network installation, these files are on your bastion host.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. 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 installation directory that contains the **install-config.yaml** file you created.

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

2. Modify the **manifests/cluster-scheduler-02-config.yml** Kubernetes manifest file to prevent Pods from being scheduled on the control plane machines:
 - a. Open the **manifests/cluster-scheduler-02-config.yml** file.
 - b. Locate the **mastersSchedulable** parameter and set its value to **False**.
 - c. Save and exit the file.

**NOTE**

Currently, due to a [Kubernetes limitation](#), router Pods running on control plane machines will not be reachable by the ingress load balancer. This step might not be required in a future minor version of OpenShift Container Platform.

3. 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
```

9.4.9. 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": {}
}
}

```

- 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.

- 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

```

- 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 **rhcoss-<version>-<architecture>-vmware.ova**.

- In the vSphere Client, create a folder in your datacenter to store your VMs.
 - Click the **VMs and Templates** view.
 - Right-click the name of your datacenter.
 - Click **New Folder → New VM and Template Folder**.
 - 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.
- 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**.

- Alternatively, prior to powering on the virtual machine add via vApp properties:
 - Navigate to a virtual machine from the vCenter Server inventory.
 - On the **Configure** tab, expand **Settings** and select **vApp options**.
 - Scroll down and under **Properties** apply the configurations from above.
- 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.

9.4.10. Creating the cluster

To create the OpenShift Container Platform cluster, you wait for the bootstrap process to complete on the machines that you provisioned 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. Monitor the bootstrap process:

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

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.14.6+c4799753c 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 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

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.

9.4.11. Logging in to the cluster

You can log in to 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
```

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

2. Verify you can run **oc** commands successfully using the exported configuration:

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

9.4.12. 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.
- Install the **jq** package.

Procedure

1. Confirm that the cluster recognizes the machines:



```
$ oc get nodes

NAME      STATUS   ROLES   AGE   VERSION
master-0  Ready    master   63m  v1.14.6+c4799753c
master-1  Ready    master   63m  v1.14.6+c4799753c
master-2  Ready    master   64m  v1.14.6+c4799753c
worker-0  NotReady worker  76s  v1.14.6+c4799753c
worker-1  NotReady worker  70s  v1.14.6+c4799753c
```

The output lists all of the machines that you created.

- Review the pending certificate signing requests (CSRs) and ensure that 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
```

9.4.13. 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:

NAME SINCE	VERSION	AVAILABLE	PROGRESSING	DEGRADED
authentication	4.2.0	True	False	False 69s
cloud-credential	4.2.0	True	False	False 12m
cluster-autoscaler	4.2.0	True	False	False 11m
console	4.2.0	True	False	False 46s
dns	4.2.0	True	False	False 11m
image-registry	4.2.0	False	True	False 5m26s
ingress	4.2.0	True	False	False 5m36s
kube-apiserver	4.2.0	True	False	False 8m53s
kube-controller-manager	4.2.0	True	False	False 7m24s
kube-scheduler	4.2.0	True	False	False 12m
machine-api	4.2.0	True	False	False 12m
machine-config	4.2.0	True	False	False 7m36s
marketplace	4.2.0	True	False	False 7m54m
monitoring	4.2.0	True	False	False 7h54s
network	4.2.0	True	False	False 5m9s
node-tuning	4.2.0	True	False	False 11m
openshift-apiserver	4.2.0	True	False	False 11m
openshift-controller-manager	4.2.0	True	False	False 5m943s
openshift-samples	4.2.0	True	False	False 3m55s
operator-lifecycle-manager	4.2.0	True	False	False 11m
operator-lifecycle-manager-catalog	4.2.0	True	False	False 11m
service-ca	4.2.0	True	False	False 11m
service-catalog-apiserver	4.2.0	True	False	False 5m26s
service-catalog-controller-manager	4.2.0	True	False	False 5m25s
storage	4.2.0	True	False	False 5m30s

2. Configure the Operators that are not available.

9.4.13.1. 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.

9.4.13.1.1. 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**.



IMPORTANT

vSphere volumes do not support the **ReadWriteMany** access mode. You must use a different storage backend, such as **NFS**, to configure the registry storage.

- 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 cannot 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
```

3. 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.

4. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

9.4.13.1.2. 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.

9.4.14. 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	4.2.0	True	False	False 10m
cloud-credential	4.2.0	True	False	False 22m

cluster-autoscaler	4.2.0	True	False	False	21m
console	4.2.0	True	False	False	10m
dns	4.2.0	True	False	False	21m
image-registry	4.2.0	True	False	False	16m
ingress	4.2.0	True	False	False	16m
kube-apiserver	4.2.0	True	False	False	19m
kube-controller-manager	4.2.0	True	False	False	18m
kube-scheduler	4.2.0	True	False	False	22m
machine-api	4.2.0	True	False	False	22m
machine-config	4.2.0	True	False	False	18m
marketplace	4.2.0	True	False	False	18m
monitoring	4.2.0	True	False	False	18m
network	4.2.0	True	False	False	16m
node-tuning	4.2.0	True	False	False	21m
openshift-apiserver	4.2.0	True	False	False	21m
openshift-controller-manager	4.2.0	True	False	False	17m
openshift-samples	4.2.0	True	False	False	14m
operator-lifecycle-manager	4.2.0	True	False	False	21m
operator-lifecycle-manager-catalog	4.2.0	True	False	False	21m
service-ca	4.2.0	True	False	False	21m
service-catalog-apiserver	4.2.0	True	False	False	16m
service-catalog-controller-manager	4.2.0	True	False	False	16m
storage	4.2.0	True	False	False	16m

When all of the cluster Operators are **AVAILABLE**, you can complete the installation.

- Monitor for cluster completion:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete ①
INFO Waiting up to 30m0s for the cluster to initialize...
```

- 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.



IMPORTANT

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

- Confirm that the Kubernetes API server is communicating with the Pods.

- 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 3m	apiserver-67b9g	1/1	Running	0
openshift-apiserver 1m	apiserver-ljcmx	1/1	Running	0
openshift-apiserver 2m	apiserver-z25h4	1/1	Running	0
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8 Running 0 5m		1/1		
...				

- 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> ①
```

- ① 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.

4. Register your cluster on the [Cluster registration](#) page.

Next steps

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

CHAPTER 10. GATHERING INSTALLATION LOGS

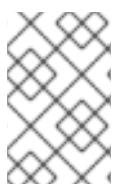
To assist in troubleshooting a failed OpenShift Container Platform installation, you can gather logs from the bootstrap and control plane, or master, machines.

Prerequisites

- You attempted to install an 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.

10.1. 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> \
--bootstrap <bootstrap_address> \
--master "<master_address> <master_address> <master_address>" ② 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 11. INSTALLATION CONFIGURATION

11.1. 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**

11.1.1. 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 nodeselectors.

11.1.2. 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 imagestreams and templates are installed on the cluster.

11.1.3. 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.

11.1.4. 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 4.2, 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 .

11.2. CONFIGURING YOUR FIREWALL

If you use a firewall, you must configure it so that OpenShift Container Platform can access the sites that it requires to function. You must always grant access to some sites, and you grant access to more if you use Red Hat Insights, the Telemetry service, a cloud to host your cluster, and certain build strategies.

11.2.1. Configuring your firewall for OpenShift Container Platform

Before you install OpenShift Container Platform, you must configure your firewall to grant access to the sites that OpenShift Container Platform requires.

Procedure

1. Whitelist the following registry URLs:

URL	Function
registry.redhat.io	Provides core container images
*.quay.io	Provides core container images
sso.redhat.com	The https://cloud.redhat.com/openshift site uses authentication from sso.redhat.com

2. Whitelist any site that provides resources for a language or framework that your builds require.
3. If you do not disable Telemetry, you must grant access to the following URLs to access Red Hat Insights:

URL	Function
cert-api.access.redhat.com	Required for Telemetry
api.access.redhat.com	Required for Telemetry
infogw.api.openshift.com	Required for Telemetry

4. If you use Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP) to host your cluster, you must grant access to the URLs that provide the cloud provider API and DNS for that cloud:

Cloud	URL	Function
AWS	*.amazonaws.com	Required to access AWS services and resources. Review the AWS Service Endpoints in the AWS documentation to determine the exact endpoints to allow for the regions that you use.
GCP	*.googleapis.com	Required to access GCP services and resources. Review Cloud Endpoints in the GCP documentation to determine the endpoints to allow for your APIs.
	accounts.google.com	Required to access your GCP account.
Azure	management.azure.com	Required to access Azure services and resources. Review the Azure REST API Reference in the Azure documentation to determine the endpoints to allow for your APIs.

5. Whitelist the following URLs:

URL	Function
mirror.openshift.com	Required to access mirrored installation content and images
*.cloudfront.net	Required by the Quay CDN to provide edge delivery of Quay.io images
*.apps	Required to access the default cluster routes unless you set an ingress wildcard during installation
quay-registry.s3.amazonaws.com	Required to access Quay image content in AWS

URL	Function
api.openshift.com	Required to check if updates are available for the cluster
art-rhcos-ci.s3.amazonaws.com	Required to download Red Hat Enterprise Linux CoreOS (RHCOS) images
api.openshift.com	Required for your cluster token
cloud.redhat.com/openshift	Required for your cluster token