



OpenShift Container Platform 4.18

Installing on GCP

Installing OpenShift Container Platform on Google Cloud Platform

OpenShift Container Platform 4.18 Installing on GCP

Installing OpenShift Container Platform on Google Cloud Platform

Legal Notice

Copyright © 2025 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

This document describes how to install OpenShift Container Platform on Google Cloud Platform.

Table of Contents

CHAPTER 1. PREPARING TO INSTALL ON GCP	11
1.1. PREREQUISITES	11
1.2. REQUIREMENTS FOR INSTALLING OPENSHIFT CONTAINER PLATFORM ON GCP	11
1.3. CHOOSING A METHOD TO INSTALL OPENSHIFT CONTAINER PLATFORM ON GCP	11
1.3.1. Installing a cluster on installer-provisioned infrastructure	11
1.3.2. Installing a cluster on user-provisioned infrastructure	12
1.4. NEXT STEPS	12
CHAPTER 2. CONFIGURING A GCP PROJECT	13
2.1. CREATING A GCP PROJECT	13
2.2. ENABLING API SERVICES IN GCP	13
2.3. CONFIGURING DNS FOR GCP	14
2.4. GCP ACCOUNT LIMITS	15
2.5. CREATING A SERVICE ACCOUNT IN GCP	16
2.5.1. Required GCP roles	17
2.5.2. Required GCP permissions for installer-provisioned infrastructure	18
2.5.3. Required GCP permissions for shared VPC installations	26
2.5.4. Required GCP permissions for user-provided service accounts	27
2.6. SUPPORTED GCP REGIONS	28
2.7. NEXT STEPS	29
CHAPTER 3. INSTALLING A CLUSTER QUICKLY ON GCP	30
3.1. PREREQUISITES	30
3.2. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM	30
3.3. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS	30
3.4. OBTAINING THE INSTALLATION PROGRAM	32
3.5. DEPLOYING THE CLUSTER	33
3.6. INSTALLING THE OPENSHIFT CLI	36
Installing the OpenShift CLI on Linux	36
Installing the OpenShift CLI on Windows	37
Installing the OpenShift CLI on macOS	37
3.7. LOGGING IN TO THE CLUSTER BY USING THE CLI	38
3.8. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM	38
3.9. NEXT STEPS	39
CHAPTER 4. INSTALLING A CLUSTER ON GCP WITH CUSTOMIZATIONS	40
4.1. PREREQUISITES	40
4.2. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM	40
4.3. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS	40
4.4. OBTAINING THE INSTALLATION PROGRAM	42
4.5. CREATING THE INSTALLATION CONFIGURATION FILE	43
4.5.1. Minimum resource requirements for cluster installation	45
4.5.2. Tested instance types for GCP	46
4.5.3. Tested instance types for GCP on 64-bit ARM infrastructures	46
4.5.4. Using custom machine types	47
4.5.5. Enabling Shielded VMs	47
4.5.6. Enabling Confidential VMs	48
4.5.7. Sample customized install-config.yaml file for GCP	49
4.5.8. Configuring the cluster-wide proxy during installation	52
4.6. MANAGING USER-DEFINED LABELS AND TAGS FOR GCP	54
4.6.1. Configuring user-defined labels and tags for GCP	56
4.6.2. Querying user-defined labels and tags for GCP	57

4.7. INSTALLING THE OPENSHIFT CLI	58
Installing the OpenShift CLI on Linux	58
Installing the OpenShift CLI on Windows	59
Installing the OpenShift CLI on macOS	59
4.8. ALTERNATIVES TO STORING ADMINISTRATOR-LEVEL SECRETS IN THE KUBE-SYSTEM PROJECT	60
4.8.1. Manually creating long-term credentials	60
4.8.2. Configuring a GCP cluster to use short-term credentials	62
4.8.2.1. Configuring the Cloud Credential Operator utility	62
4.8.2.2. Creating GCP resources with the Cloud Credential Operator utility	65
4.8.2.3. Incorporating the Cloud Credential Operator utility manifests	67
4.9. USING THE GCP MARKETPLACE OFFERING	68
4.10. DEPLOYING THE CLUSTER	69
4.11. LOGGING IN TO THE CLUSTER BY USING THE CLI	71
4.12. TELEMETRY ACCESS FOR OPENShift CONTAINER PLATFORM	72
4.13. NEXT STEPS	72
CHAPTER 5. INSTALLING A CLUSTER ON GCP WITH NETWORK CUSTOMIZATIONS	73
5.1. PREREQUISITES	73
5.2. INTERNET ACCESS FOR OPENSIFT CONTAINER PLATFORM	73
5.3. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS	73
5.4. OBTAINING THE INSTALLATION PROGRAM	75
5.5. CREATING THE INSTALLATION CONFIGURATION FILE	76
5.5.1. Minimum resource requirements for cluster installation	78
5.5.2. Tested instance types for GCP	79
5.5.3. Tested instance types for GCP on 64-bit ARM infrastructures	80
5.5.4. Using custom machine types	80
5.5.5. Enabling Shielded VMs	81
5.5.6. Enabling Confidential VMs	81
5.5.7. Sample customized install-config.yaml file for GCP	82
5.5.8. Configuring the cluster-wide proxy during installation	86
5.6. INSTALLING THE OPENSIFT CLI	87
Installing the OpenShift CLI on Linux	87
Installing the OpenShift CLI on Windows	88
Installing the OpenShift CLI on macOS	88
5.7. ALTERNATIVES TO STORING ADMINISTRATOR-LEVEL SECRETS IN THE KUBE-SYSTEM PROJECT	89
5.7.1. Manually creating long-term credentials	89
5.7.2. Configuring a GCP cluster to use short-term credentials	92
5.7.2.1. Configuring the Cloud Credential Operator utility	92
5.7.2.2. Creating GCP resources with the Cloud Credential Operator utility	94
5.7.2.3. Incorporating the Cloud Credential Operator utility manifests	96
5.8. NETWORK CONFIGURATION PHASES	98
5.9. SPECIFYING ADVANCED NETWORK CONFIGURATION	98
5.10. CLUSTER NETWORK OPERATOR CONFIGURATION	99
5.10.1. Cluster Network Operator configuration object	100
defaultNetwork object configuration	101
Configuration for the OVN-Kubernetes network plugin	101
5.11. DEPLOYING THE CLUSTER	106
5.12. LOGGING IN TO THE CLUSTER BY USING THE CLI	108
5.13. TELEMETRY ACCESS FOR OPENSIFT CONTAINER PLATFORM	109
5.14. NEXT STEPS	109
CHAPTER 6. INSTALLING A CLUSTER ON GCP IN A DISCONNECTED ENVIRONMENT	110
6.1. PREREQUISITES	110

6.2. ABOUT INSTALLATIONS IN RESTRICTED NETWORKS	110
6.2.1. Additional limits	111
6.3. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM	111
6.4. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS	111
6.5. CREATING THE INSTALLATION CONFIGURATION FILE	113
6.5.1. Minimum resource requirements for cluster installation	115
6.5.2. Tested instance types for GCP	116
6.5.3. Tested instance types for GCP on 64-bit ARM infrastructures	117
6.5.4. Using custom machine types	118
6.5.5. Enabling Shielded VMs	118
6.5.6. Enabling Confidential VMs	119
6.5.7. Sample customized install-config.yaml file for GCP	120
6.5.8. Create an Ingress Controller with global access on GCP	123
6.5.9. Configuring the cluster-wide proxy during installation	125
6.6. INSTALLING THE OPENSHIFT CLI	126
Installing the OpenShift CLI on Linux	127
Installing the OpenShift CLI on Windows	127
Installing the OpenShift CLI on macOS	128
6.7. ALTERNATIVES TO STORING ADMINISTRATOR-LEVEL SECRETS IN THE KUBE-SYSTEM PROJECT	128
6.7.1. Manually creating long-term credentials	128
6.7.2. Configuring a GCP cluster to use short-term credentials	131
6.7.2.1. Configuring the Cloud Credential Operator utility	131
6.7.2.2. Creating GCP resources with the Cloud Credential Operator utility	134
6.7.2.3. Incorporating the Cloud Credential Operator utility manifests	135
6.8. DEPLOYING THE CLUSTER	137
6.9. LOGGING IN TO THE CLUSTER BY USING THE CLI	138
6.10. DISABLING THE DEFAULT OPERATORHUB CATALOG SOURCES	139
6.11. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM	139
6.12. NEXT STEPS	140
CHAPTER 7. INSTALLING A CLUSTER ON GCP INTO AN EXISTING VPC	141
7.1. PREREQUISITES	141
7.2. ABOUT USING A CUSTOM VPC	141
7.2.1. Requirements for using your VPC	141
7.2.2. VPC validation	141
7.2.3. Division of permissions	142
7.2.4. Isolation between clusters	142
7.3. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM	142
7.4. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS	142
7.5. OBTAINING THE INSTALLATION PROGRAM	144
7.6. CREATING THE INSTALLATION CONFIGURATION FILE	145
7.6.1. Minimum resource requirements for cluster installation	147
7.6.2. Tested instance types for GCP	148
7.6.3. Tested instance types for GCP on 64-bit ARM infrastructures	149
7.6.4. Using custom machine types	149
7.6.5. Enabling Shielded VMs	150
7.6.6. Enabling Confidential VMs	150
7.6.7. Sample customized install-config.yaml file for GCP	151
7.6.8. Create an Ingress Controller with global access on GCP	155
7.6.9. Configuring the cluster-wide proxy during installation	156
7.7. INSTALLING THE OPENSHIFT CLI	158
Installing the OpenShift CLI on Linux	158
Installing the OpenShift CLI on Windows	158

Installing the OpenShift CLI on macOS	159
7.8. ALTERNATIVES TO STORING ADMINISTRATOR-LEVEL SECRETS IN THE KUBE-SYSTEM PROJECT	159
7.8.1. Manually creating long-term credentials	160
7.8.2. Configuring a GCP cluster to use short-term credentials	162
7.8.2.1. Configuring the Cloud Credential Operator utility	162
7.8.2.2. Creating GCP resources with the Cloud Credential Operator utility	165
7.8.2.3. Incorporating the Cloud Credential Operator utility manifests	167
7.9. DEPLOYING THE CLUSTER	168
7.10. LOGGING IN TO THE CLUSTER BY USING THE CLI	170
7.11. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM	170
7.12. NEXT STEPS	171
CHAPTER 8. INSTALLING A CLUSTER ON GCP INTO A SHARED VPC	172
8.1. PREREQUISITES	172
8.2. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM	172
8.3. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS	173
8.4. OBTAINING THE INSTALLATION PROGRAM	174
8.5. CREATING THE INSTALLATION FILES FOR GCP	175
8.5.1. Manually creating the installation configuration file	176
8.5.2. Enabling Shielded VMs	176
8.5.3. Enabling Confidential VMs	177
8.5.4. Sample customized install-config.yaml file for shared VPC installation	178
8.5.5. Configuring the cluster-wide proxy during installation	180
8.6. INSTALLING THE OPENSHIFT CLI	181
Installing the OpenShift CLI on Linux	181
Installing the OpenShift CLI on Windows	182
Installing the OpenShift CLI on macOS	182
8.7. ALTERNATIVES TO STORING ADMINISTRATOR-LEVEL SECRETS IN THE KUBE-SYSTEM PROJECT	183
8.7.1. Manually creating long-term credentials	183
8.7.2. Configuring a GCP cluster to use short-term credentials	186
8.7.2.1. Configuring the Cloud Credential Operator utility	186
8.7.2.2. Creating GCP resources with the Cloud Credential Operator utility	188
8.7.2.3. Incorporating the Cloud Credential Operator utility manifests	190
8.8. DEPLOYING THE CLUSTER	191
8.9. LOGGING IN TO THE CLUSTER BY USING THE CLI	193
8.10. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM	194
8.11. NEXT STEPS	194
CHAPTER 9. INSTALLING A PRIVATE CLUSTER ON GCP	195
9.1. PREREQUISITES	195
9.2. PRIVATE CLUSTERS	195
9.2.1. Private clusters in GCP	195
9.2.1.1. Limitations	196
9.3. ABOUT USING A CUSTOM VPC	196
9.3.1. Requirements for using your VPC	196
9.3.2. Division of permissions	197
9.3.3. Isolation between clusters	197
9.4. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM	198
9.5. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS	198
9.6. OBTAINING THE INSTALLATION PROGRAM	200
9.7. MANUALLY CREATING THE INSTALLATION CONFIGURATION FILE	201
9.7.1. Minimum resource requirements for cluster installation	202
9.7.2. Tested instance types for GCP	203

9.7.3. Tested instance types for GCP on 64-bit ARM infrastructures	204
9.7.4. Using custom machine types	204
9.7.5. Enabling Shielded VMs	205
9.7.6. Enabling Confidential VMs	206
9.7.7. Sample customized install-config.yaml file for GCP	207
9.7.8. Create an Ingress Controller with global access on GCP	210
9.7.9. Configuring the cluster-wide proxy during installation	211
9.8. INSTALLING THE OPENSHIFT CLI	213
Installing the OpenShift CLI on Linux	213
Installing the OpenShift CLI on Windows	214
Installing the OpenShift CLI on macOS	214
9.9. ALTERNATIVES TO STORING ADMINISTRATOR-LEVEL SECRETS IN THE KUBE-SYSTEM PROJECT	215
9.9.1. Manually creating long-term credentials	215
9.9.2. Configuring a GCP cluster to use short-term credentials	217
9.9.2.1. Configuring the Cloud Credential Operator utility	217
9.9.2.2. Creating GCP resources with the Cloud Credential Operator utility	220
9.9.2.3. Incorporating the Cloud Credential Operator utility manifests	222
9.10. DEPLOYING THE CLUSTER	223
9.11. LOGGING IN TO THE CLUSTER BY USING THE CLI	225
9.12. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM	226
9.13. NEXT STEPS	226
CHAPTER 10. INSTALLING A CLUSTER ON USER-PROVISIONED INFRASTRUCTURE IN GCP BY USING DEPLOYMENT MANAGER TEMPLATES	227
10.1. PREREQUISITES	227
10.2. CERTIFICATE SIGNING REQUESTS MANAGEMENT	227
10.3. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM	227
10.4. CONFIGURING YOUR GCP PROJECT	228
10.4.1. Creating a GCP project	228
10.4.2. Enabling API services in GCP	228
10.4.3. Configuring DNS for GCP	229
10.4.4. GCP account limits	230
10.4.5. Creating a service account in GCP	231
10.4.6. Required GCP roles	232
10.4.7. Required GCP permissions for user-provisioned infrastructure	233
10.4.8. Supported GCP regions	242
10.4.9. Installing and configuring CLI tools for GCP	243
10.5. REQUIREMENTS FOR A CLUSTER WITH USER-PROVISIONED INFRASTRUCTURE	244
10.5.1. Required machines for cluster installation	244
10.5.2. Minimum resource requirements for cluster installation	245
10.5.3. Tested instance types for GCP	246
10.5.4. Tested instance types for GCP on 64-bit ARM infrastructures	246
10.5.5. Using custom machine types	247
10.6. CREATING THE INSTALLATION FILES FOR GCP	247
10.6.1. Optional: Creating a separate /var partition	247
10.6.2. Creating the installation configuration file	249
10.6.3. Enabling Shielded VMs	251
10.6.4. Enabling Confidential VMs	252
10.6.5. Configuring the cluster-wide proxy during installation	253
10.6.6. Creating the Kubernetes manifest and Ignition config files	254
10.7. EXPORTING COMMON VARIABLES	257
10.7.1. Extracting the infrastructure name	257
10.7.2. Exporting common variables for Deployment Manager templates	257

10.8. CREATING A VPC IN GCP	258
10.8.1. Deployment Manager template for the VPC	259
10.9. NETWORKING REQUIREMENTS FOR USER-PROVISIONED INFRASTRUCTURE	260
10.9.1. Setting the cluster node hostnames through DHCP	260
10.9.2. Network connectivity requirements	261
10.10. CREATING LOAD BALANCERS IN GCP	262
10.10.1. Deployment Manager template for the external load balancer	264
10.10.2. Deployment Manager template for the internal load balancer	265
10.11. CREATING A PRIVATE DNS ZONE IN GCP	266
10.11.1. Deployment Manager template for the private DNS	267
10.12. CREATING FIREWALL RULES IN GCP	268
10.12.1. Deployment Manager template for firewall rules	269
10.13. CREATING IAM ROLES IN GCP	271
10.13.1. Deployment Manager template for IAM roles	273
10.14. CREATING THE RHCOS CLUSTER IMAGE FOR THE GCP INFRASTRUCTURE	273
10.15. CREATING THE BOOTSTRAP MACHINE IN GCP	274
10.15.1. Deployment Manager template for the bootstrap machine	276
10.16. CREATING THE CONTROL PLANE MACHINES IN GCP	277
10.16.1. Deployment Manager template for control plane machines	279
10.17. WAIT FOR BOOTSTRAP COMPLETION AND REMOVE BOOTSTRAP RESOURCES IN GCP	282
10.18. CREATING ADDITIONAL WORKER MACHINES IN GCP	282
10.18.1. Deployment Manager template for worker machines	285
10.19. INSTALLING THE OPENSHIFT CLI	286
Installing the OpenShift CLI on Linux	286
Installing the OpenShift CLI on Windows	286
Installing the OpenShift CLI on macOS	287
10.20. LOGGING IN TO THE CLUSTER BY USING THE CLI	287
10.21. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES	288
10.22. OPTIONAL: ADDING THE INGRESS DNS RECORDS	291
10.23. COMPLETING A GCP INSTALLATION ON USER-PROVISIONED INFRASTRUCTURE	292
10.24. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM	294
10.25. NEXT STEPS	295
CHAPTER 11. INSTALLING A CLUSTER INTO A SHARED VPC ON GCP USING DEPLOYMENT MANAGER TEMPLATES	296
11.1. PREREQUISITES	296
11.2. CERTIFICATE SIGNING REQUESTS MANAGEMENT	296
11.3. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM	297
11.4. CONFIGURING THE GCP PROJECT THAT HOSTS YOUR CLUSTER	297
11.4.1. Creating a GCP project	297
11.4.2. Enabling API services in GCP	298
11.4.3. GCP account limits	298
11.4.4. Creating a service account in GCP	300
11.4.4.1. Required GCP roles	301
11.4.5. Supported GCP regions	302
11.4.6. Installing and configuring CLI tools for GCP	303
11.5. REQUIREMENTS FOR A CLUSTER WITH USER-PROVISIONED INFRASTRUCTURE	304
11.5.1. Required machines for cluster installation	304
11.5.2. Minimum resource requirements for cluster installation	304
11.5.3. Tested instance types for GCP	306
11.5.4. Using custom machine types	306
11.6. CONFIGURING THE GCP PROJECT THAT HOSTS YOUR SHARED VPC NETWORK	307
11.6.1. Configuring DNS for GCP	308

11.6.2. Creating a VPC in GCP	308
11.6.2.1. Deployment Manager template for the VPC	310
11.7. CREATING THE INSTALLATION FILES FOR GCP	311
11.7.1. Manually creating the installation configuration file	311
11.7.2. Enabling Shielded VMs	312
11.7.3. Enabling Confidential VMs	313
11.7.4. Sample customized install-config.yaml file for GCP	314
11.7.5. Configuring the cluster-wide proxy during installation	316
11.7.6. Creating the Kubernetes manifest and Ignition config files	318
11.8. EXPORTING COMMON VARIABLES	321
11.8.1. Extracting the infrastructure name	321
11.8.2. Exporting common variables for Deployment Manager templates	321
11.9. NETWORKING REQUIREMENTS FOR USER-PROVISIONED INFRASTRUCTURE	322
11.9.1. Setting the cluster node hostnames through DHCP	322
11.9.2. Network connectivity requirements	322
11.10. CREATING LOAD BALANCERS IN GCP	323
11.10.1. Deployment Manager template for the external load balancer	325
11.10.2. Deployment Manager template for the internal load balancer	326
11.11. CREATING A PRIVATE DNS ZONE IN GCP	328
11.11.1. Deployment Manager template for the private DNS	329
11.12. CREATING FIREWALL RULES IN GCP	330
11.12.1. Deployment Manager template for firewall rules	331
11.13. CREATING IAM ROLES IN GCP	333
11.13.1. Deployment Manager template for IAM roles	335
11.14. CREATING THE RHCOSS CLUSTER IMAGE FOR THE GCP INFRASTRUCTURE	336
11.15. CREATING THE BOOTSTRAP MACHINE IN GCP	337
11.15.1. Deployment Manager template for the bootstrap machine	339
11.16. CREATING THE CONTROL PLANE MACHINES IN GCP	340
11.16.1. Deployment Manager template for control plane machines	342
11.17. WAIT FOR BOOTSTRAP COMPLETION AND REMOVE BOOTSTRAP RESOURCES IN GCP	344
11.18. CREATING ADDITIONAL WORKER MACHINES IN GCP	345
11.18.1. Deployment Manager template for worker machines	347
11.19. INSTALLING THE OPENSHIFT CLI	348
Installing the OpenShift CLI on Linux	348
Installing the OpenShift CLI on Windows	349
Installing the OpenShift CLI on macOS	349
11.20. LOGGING IN TO THE CLUSTER BY USING THE CLI	350
11.21. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES	350
11.22. ADDING THE INGRESS DNS RECORDS	353
11.23. ADDING INGRESS FIREWALL RULES	354
11.23.1. Creating cluster-wide firewall rules for a shared VPC in GCP	355
11.24. COMPLETING A GCP INSTALLATION ON USER-PROVISIONED INFRASTRUCTURE	356
11.25. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM	358
11.26. NEXT STEPS	359
CHAPTER 12. INSTALLING A CLUSTER ON GCP IN A DISCONNECTED ENVIRONMENT WITH USER-PROVISIONED INFRASTRUCTURE	360
12.1. PREREQUISITES	360
12.2. ABOUT INSTALLATIONS IN RESTRICTED NETWORKS	360
12.2.1. Additional limits	361
12.3. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM	361
12.4. CONFIGURING YOUR GCP PROJECT	361
12.4.1. Creating a GCP project	361

12.4.2. Enabling API services in GCP	362
12.4.3. Configuring DNS for GCP	363
12.4.4. GCP account limits	363
12.4.5. Creating a service account in GCP	365
12.4.6. Required GCP roles	366
12.4.7. Required GCP permissions for user-provisioned infrastructure	367
12.4.8. Supported GCP regions	375
12.4.9. Installing and configuring CLI tools for GCP	377
12.5. REQUIREMENTS FOR A CLUSTER WITH USER-PROVISIONED INFRASTRUCTURE	377
12.5.1. Required machines for cluster installation	377
12.5.2. Minimum resource requirements for cluster installation	378
12.5.3. Tested instance types for GCP	379
12.5.4. Using custom machine types	380
12.6. CREATING THE INSTALLATION FILES FOR GCP	380
12.6.1. Optional: Creating a separate /var partition	380
12.6.2. Creating the installation configuration file	383
12.6.3. Enabling Shielded VMs	385
12.6.4. Enabling Confidential VMs	386
12.6.5. Configuring the cluster-wide proxy during installation	387
12.6.6. Creating the Kubernetes manifest and Ignition config files	388
12.7. EXPORTING COMMON VARIABLES	391
12.7.1. Extracting the infrastructure name	391
12.7.2. Exporting common variables for Deployment Manager templates	391
12.8. CREATING A VPC IN GCP	392
12.8.1. Deployment Manager template for the VPC	393
12.9. NETWORKING REQUIREMENTS FOR USER-PROVISIONED INFRASTRUCTURE	394
12.9.1. Setting the cluster node hostnames through DHCP	394
12.9.2. Network connectivity requirements	394
12.10. CREATING LOAD BALANCERS IN GCP	395
12.10.1. Deployment Manager template for the external load balancer	397
12.10.2. Deployment Manager template for the internal load balancer	398
12.11. CREATING A PRIVATE DNS ZONE IN GCP	400
12.11.1. Deployment Manager template for the private DNS	401
12.12. CREATING FIREWALL RULES IN GCP	402
12.12.1. Deployment Manager template for firewall rules	403
12.13. CREATING IAM ROLES IN GCP	405
12.13.1. Deployment Manager template for IAM roles	407
12.14. CREATING THE RHCOS CLUSTER IMAGE FOR THE GCP INFRASTRUCTURE	407
12.15. CREATING THE BOOTSTRAP MACHINE IN GCP	408
12.15.1. Deployment Manager template for the bootstrap machine	410
12.16. CREATING THE CONTROL PLANE MACHINES IN GCP	411
12.16.1. Deployment Manager template for control plane machines	413
12.17. WAIT FOR BOOTSTRAP COMPLETION AND REMOVE BOOTSTRAP RESOURCES IN GCP	415
12.18. CREATING ADDITIONAL WORKER MACHINES IN GCP	416
12.18.1. Deployment Manager template for worker machines	418
12.19. LOGGING IN TO THE CLUSTER BY USING THE CLI	419
12.20. DISABLING THE DEFAULT OPERATORHUB CATALOG SOURCES	420
12.21. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES	420
12.22. OPTIONAL: ADDING THE INGRESS DNS RECORDS	423
12.23. COMPLETING A GCP INSTALLATION ON USER-PROVISIONED INFRASTRUCTURE	424
12.24. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM	426
12.25. NEXT STEPS	427

CHAPTER 13. INSTALLING A THREE-NODE CLUSTER ON GCP	428
13.1. CONFIGURING A THREE-NODE CLUSTER	428
13.2. NEXT STEPS	429
CHAPTER 14. INSTALLATION CONFIGURATION PARAMETERS FOR GCP	430
14.1. AVAILABLE INSTALLATION CONFIGURATION PARAMETERS FOR GCP	430
14.1.1. Required configuration parameters	430
14.1.2. Network configuration parameters	431
14.1.3. Optional configuration parameters	433
14.1.4. Additional Google Cloud Platform (GCP) configuration parameters	440
CHAPTER 15. UNINSTALLING A CLUSTER ON GCP	474
15.1. REMOVING A CLUSTER THAT USES INSTALLER-PROVISIONED INFRASTRUCTURE	474
15.2. DELETING GOOGLE CLOUD PLATFORM RESOURCES WITH THE CLOUD CREDENTIAL OPERATOR UTILITY	474
CHAPTER 16. INSTALLING A CLUSTER WITH THE SUPPORT FOR CONFIGURING MULTI-ARCHITECTURE COMPUTE MACHINES	476
16.1. INSTALLING A CLUSTER WITH MULTI-ARCHITECTURE SUPPORT	476

CHAPTER 1. PREPARING TO INSTALL ON GCP

1.1. PREREQUISITES

- You reviewed details about the [OpenShift Container Platform installation and update processes](#).
- You read the documentation on [selecting a cluster installation method and preparing it for users](#).

1.2. REQUIREMENTS FOR INSTALLING OPENSHIFT CONTAINER PLATFORM ON GCP

Before installing OpenShift Container Platform on Google Cloud Platform (GCP), you must create a service account and configure a GCP project. See [Configuring a GCP project](#) for details about creating a project, enabling API services, configuring DNS, GCP account limits, and supported GCP regions.

If the cloud identity and access management (IAM) APIs are not accessible in your environment, or if you do not want to store an administrator-level credential secret in the **kube-system** namespace, see [Manually creating long-term credentials for GCP](#) for other options.

1.3. CHOOSING A METHOD TO INSTALL OPENSHIFT CONTAINER PLATFORM ON GCP

You can install OpenShift Container Platform on installer-provisioned or user-provisioned infrastructure. The default installation type uses installer-provisioned infrastructure, where the installation program provisions the underlying infrastructure for the cluster. You can also install OpenShift Container Platform on infrastructure that you provision. If you do not use infrastructure that the installation program provisions, you must manage and maintain the cluster resources yourself.

See [Installation process](#) for more information about installer-provisioned and user-provisioned installation processes.

1.3.1. Installing a cluster on installer-provisioned infrastructure

You can install a cluster on GCP infrastructure that is provisioned by the OpenShift Container Platform installation program, by using one of the following methods:

- **Installing a cluster quickly on GCP** You can install OpenShift Container Platform on GCP infrastructure that is provisioned by the OpenShift Container Platform installation program. You can install a cluster quickly by using the default configuration options.
- **Installing a customized cluster on GCP** You can install a customized cluster on GCP infrastructure that the installation program provisions. The installation program allows for some customization to be applied at the installation stage. Many other customization options are available [post-installation](#).
- **Installing a cluster on GCP with network customizations** You can customize your OpenShift Container Platform network configuration during installation, so that your cluster can coexist with your existing IP address allocations and adhere to your network requirements.
- **Installing a cluster on GCP in a restricted network** You can install OpenShift Container Platform on GCP on installer-provisioned infrastructure by using an internal mirror of the

installation release content. You can use this method to install a cluster that does not require an active internet connection to obtain the software components. While you can install OpenShift Container Platform by using the mirrored content, your cluster still requires internet access to use the GCP APIs.

- **Installing a cluster into an existing Virtual Private Cloud** You can install OpenShift Container Platform on an existing GCP Virtual Private Cloud (VPC). You can use this installation method if you have constraints set by the guidelines of your company, such as limits on creating new accounts or infrastructure.
- **Installing a private cluster on an existing VPC** You can install a private cluster on an existing GCP VPC. You can use this method to deploy OpenShift Container Platform on an internal network that is not visible to the internet.

1.3.2. Installing a cluster on user-provisioned infrastructure

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

- **Installing a cluster on GCP with user-provisioned infrastructure** You can install OpenShift Container Platform on GCP infrastructure that you provide. You can use the provided Deployment Manager templates to assist with the installation.
- **Installing a cluster with shared VPC on user-provisioned infrastructure in GCP** You can use the provided Deployment Manager templates to create GCP resources in a shared VPC infrastructure.
- **Installing a cluster on GCP in a restricted network with user-provisioned infrastructure** You can install OpenShift Container Platform on GCP in a restricted network with user-provisioned infrastructure. By creating an internal mirror of the installation release content, you can install a cluster that does not require an active internet connection to obtain the software components. You can also use this installation method to ensure that your clusters only use container images that satisfy your organizational controls on external content.

1.4. NEXT STEPS

- [Configuring a GCP project](#)

CHAPTER 2. CONFIGURING A GCP PROJECT

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

2.1. 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.



IMPORTANT

Your GCP project must use the Premium Network Service Tier if you are using installer-provisioned infrastructure. The Standard Network Service Tier is not supported for clusters installed using the installation program. The installation program configures internal load balancing for the **api-int.<cluster_name>. <base_domain>** URL; the Premium Tier is required for internal load balancing.

2.2. 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. You may also enable optional API services which are not required for installation. See [Enabling services](#) in the GCP documentation.

Table 2.1. Required API services

API service	Console service name
Compute Engine API	compute.googleapis.com
Cloud Resource Manager API	cloudresourcemanager.googleapis.com
Google DNS API	dns.googleapis.com
IAM Service Account Credentials API	iamcredentials.googleapis.com

API service	Console service name
Identity and Access Management (IAM) API	iam.googleapis.com
Service Usage API	serviceusage.googleapis.com

Table 2.2. Optional API services

API service	Console service name
Google Cloud APIs	cloudapis.googleapis.com
Service Management API	servicemanagement.googleapis.com
Google Cloud Storage JSON API	storage-api.googleapis.com
Cloud Storage	storage-component.googleapis.com

2.3. 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 migrated your root domain to Google Cloud DNS, migrate your DNS records. See [Migrating to Cloud DNS](#) in the GCP documentation.
6. If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain. This process might include a request to your company's IT department or the division that controls the root domain and DNS services for your company.

2.4. 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 2.3. GCP resources used in a default cluster

Service	Component	Location	Total resources required	Resources removed after bootstrap
Service account	IAM	Global	6	1
Firewall rules	Compute	Global	11	1
Forwarding rules	Compute	Global	2	0
In-use global IP addresses	Compute	Global	4	1
Health checks	Compute	Global	3	0
Images	Compute	Global	1	0
Networks	Compute	Global	2	0
Static IP addresses	Compute	Region	4	1
Routers	Compute	Global	1	0
Routes	Compute	Global	2	0
Subnetworks	Compute	Global	2	0
Target pools	Compute	Global	3	0
CPUs	Compute	Region	28	4

Service	Component	Location	Total resources required	Resources removed after bootstrap
Persistent disk SSD (GB)	Compute	Region	896	128



NOTE

If any of the quotas are insufficient during installation, the installation program displays an error that states both which quota was exceeded and the region.

Be sure to consider your actual cluster size, planned cluster growth, and any usage from other clusters that are associated with your account. The CPU, static IP addresses, and persistent disk SSD (storage) quotas are the ones that are most likely to be insufficient.

If you plan to deploy your cluster in one of the following regions, you will exceed the maximum storage quota and are likely to exceed the CPU quota limit:

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europe-north1**
- **europe-west2**
- **europe-west3**
- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

You can increase resource quotas from the [GCP console](#), but you might need to file a support ticket. Be sure to plan your cluster size early so that you can allow time to resolve the support ticket before you install your OpenShift Container Platform cluster.

2.5. CREATING A SERVICE ACCOUNT IN GCP

OpenShift Container Platform requires a Google Cloud Platform (GCP) service account that provides authentication and authorization to access data in the Google APIs. If you do not have an existing IAM service account that contains the required roles in your project, you must create one.

Prerequisites

- You created a project to host your cluster.

Procedure

1. Create a 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](#).



NOTE

While making the service account an owner of the project is the easiest way to gain the required permissions, it means that service account has complete control over the project. You must determine if the risk that comes from offering that power is acceptable.

3. You can create the service account key in JSON format, or attach the service account to a GCP virtual machine. See [Creating service account keys](#) and [Creating and enabling service accounts for instances](#) in the GCP documentation.



NOTE

If you use a virtual machine with an attached service account to create your cluster, you must set **credentialsMode: Manual** in the **install-config.yaml** file before installation.

2.5.1. Required GCP roles

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. If your organization's security policies require a more restrictive set of permissions, you can create a service account with the following permissions. If you deploy your cluster into an existing virtual private cloud (VPC), the service account does not require certain networking permissions, which are noted in the following lists:

Required roles for the installation program

- Compute Admin
- Role Administrator
- Security Admin
- Service Account Admin
- Service Account Key Admin
- Service Account User
- Storage Admin

Required roles for creating network resources during installation

- DNS Administrator

Required roles for using the Cloud Credential Operator in passthrough mode

- Compute Load Balancer Admin
- Tag User

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

Table 2.4. GCP service account roles

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
	roles/artifactregistry.reader

2.5.2. Required GCP permissions for installer-provisioned infrastructure

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.

If your organization's security policies require a more restrictive set of permissions, you can create [custom roles](#) with the necessary permissions. The following permissions are required for the installer-provisioned infrastructure for creating and deleting the OpenShift Container Platform cluster.

Example 2.1. Required permissions for creating network resources

- **compute.addresses.create**
- **compute.addresses.createInternal**
- **compute.addresses.delete**
- **compute.addresses.get**
- **compute.addresses.list**

- `compute.addresses.use`
- `compute.addresses.useInternal`
- `compute.firewalls.create`
- `compute.firewalls.delete`
- `compute.firewalls.get`
- `compute.firewalls.list`
- `compute.forwardingRules.create`
- `compute.forwardingRules.get`
- `compute.forwardingRules.list`
- `compute.forwardingRules.setLabels`
- `compute.globalAddresses.create`
- `compute.globalAddresses.get`
- `compute.globalAddresses.use`
- `compute.globalForwardingRules.create`
- `compute.globalForwardingRules.get`
- `compute.globalForwardingRules.setLabels`
- `compute.networks.create`
- `compute.networks.get`
- `compute.networks.list`
- `compute.networks.updatePolicy`
- `compute.networks.use`
- `compute.routers.create`
- `compute.routers.get`
- `compute.routers.list`
- `compute.routers.update`
- `compute.routes.list`
- `compute.subnetworks.create`
- `compute.subnetworks.get`
- `compute.subnetworks.list`

- `compute.subnetworks.use`
- `compute.subnetworks.useExternalIIP`

Example 2.2. Required permissions for creating load balancer resources

- `compute.backendServices.create`
- `compute.backendServices.get`
- `compute.backendServices.list`
- `compute.backendServices.update`
- `compute.backendServices.use`
- `compute.regionBackendServices.create`
- `compute.regionBackendServices.get`
- `compute.regionBackendServices.list`
- `compute.regionBackendServices.update`
- `compute.regionBackendServices.use`
- `compute.targetPools.addInstance`
- `compute.targetPools.create`
- `compute.targetPools.get`
- `compute.targetPools.list`
- `compute.targetPools.removeInstance`
- `compute.targetPools.use`
- `compute.targetTcpProxies.create`
- `compute.targetTcpProxies.get`
- `compute.targetTcpProxies.use`

Example 2.3. Required permissions for creating DNS resources

- `dns.changes.create`
- `dns.changes.get`
- `dns.managedZones.create`
- `dns.managedZones.get`
- `dns.managedZones.list`

- `dns.networks.bindPrivateDNSZone`
- `dns.resourceRecordSets.create`
- `dns.resourceRecordSets.list`

Example 2.4. Required permissions for creating Service Account resources

- `iam.serviceAccountKeys.create`
- `iam.serviceAccountKeys.delete`
- `iam.serviceAccountKeys.get`
- `iam.serviceAccountKeys.list`
- `iam.serviceAccounts.actAs`
- `iam.serviceAccounts.create`
- `iam.serviceAccounts.delete`
- `iam.serviceAccounts.get`
- `iam.serviceAccounts.list`
- `resourcemanager.projects.get`
- `resourcemanager.projects.getIamPolicy`
- `resourcemanager.projects.setIamPolicy`

Example 2.5. Required permissions for creating compute resources

- `compute.disks.create`
- `compute.disks.get`
- `compute.disks.list`
- `compute.disks.setLabels`
- `compute.instanceGroups.create`
- `compute.instanceGroups.delete`
- `compute.instanceGroups.get`
- `compute.instanceGroups.list`
- `compute.instanceGroups.update`
- `compute.instanceGroups.use`
- `compute.instances.create`

- `compute.instances.delete`
- `compute.instances.get`
- `compute.instances.list`
- `compute.instances.setLabels`
- `compute.instances.setMetadata`
- `compute.instances.setServiceAccount`
- `compute.instances.setTags`
- `compute.instances.use`
- `compute.machineTypes.get`
- `compute.machineTypes.list`

Example 2.6. Required for creating storage resources

- `storage.buckets.create`
- `storage.buckets.delete`
- `storage.buckets.get`
- `storage.buckets.list`
- `storage.objects.create`
- `storage.objects.delete`
- `storage.objects.get`
- `storage.objects.list`

Example 2.7. Required permissions for creating health check resources

- `compute.healthChecks.create`
- `compute.healthChecks.get`
- `compute.healthChecks.list`
- `compute.healthChecks.useReadOnly`
- `compute.httpHealthChecks.create`
- `compute.httpHealthChecks.get`
- `compute.httpHealthChecks.list`
- `compute.httpHealthChecks.useReadOnly`

- `compute.regionHealthChecks.create`
- `compute.regionHealthChecks.get`
- `compute.regionHealthChecks.useReadOnly`

Example 2.8. Required permissions to get GCP zone and region related information

- `compute.globalOperations.get`
- `compute.regionOperations.get`
- `compute.regions.get`
- `compute.regions.list`
- `compute.zoneOperations.get`
- `compute.zones.get`
- `compute.zones.list`

Example 2.9. Required permissions for checking services and quotas

- `monitoring.timeSeries.list`
- `serviceusage.quotas.get`
- `serviceusage.services.list`

Example 2.10. Required IAM permissions for installation

- `iam.roles.create`
- `iam.roles.get`
- `iam.roles.update`

Example 2.11. Required permissions when authenticating without a service account key

- `iam.serviceAccounts.signBlob`

Example 2.12. Optional Images permissions for installation

- `compute.images.list`

Example 2.13. Optional permission for running gather bootstrap

- `compute.instances.getSerialPortOutput`

Example 2.14. Required permissions for deleting network resources

- **compute.addresses.delete**
- **compute.addresses.deleteInternal**
- **compute.addresses.list**
- **compute.addresses.setLabels**
- **compute.firewalls.delete**
- **compute.firewalls.list**
- **compute.forwardingRules.delete**
- **compute.forwardingRules.list**
- **compute.globalAddresses.delete**
- **compute.globalAddresses.list**
- **compute.globalForwardingRules.delete**
- **compute.globalForwardingRules.list**
- **compute.networks.delete**
- **compute.networks.list**
- **compute.networks.updatePolicy**
- **compute.routers.delete**
- **compute.routers.list**
- **compute.routes.list**
- **compute.subnetworks.delete**
- **compute.subnetworks.list**

Example 2.15. Required permissions for deleting load balancer resources

- **compute.backendServices.delete**
- **compute.backendServices.list**
- **compute.regionBackendServices.delete**
- **compute.regionBackendServices.list**
- **compute.targetPools.delete**
- **compute.targetPools.list**

- `compute.targetTcpProxies.delete`
- `compute.targetTcpProxies.list`

Example 2.16. Required permissions for deleting DNS resources

- `dns.changes.create`
- `dns.managedZones.delete`
- `dns.managedZones.get`
- `dns.managedZones.list`
- `dns.resourceRecordSets.delete`
- `dns.resourceRecordSets.list`

Example 2.17. Required permissions for deleting Service Account resources

- `iam.serviceAccounts.delete`
- `iam.serviceAccounts.get`
- `iam.serviceAccounts.list`
- `resourcemanager.projects.getIamPolicy`
- `resourcemanager.projects.setIamPolicy`

Example 2.18. Required permissions for deleting compute resources

- `compute.disks.delete`
- `compute.disks.list`
- `compute.instanceGroups.delete`
- `compute.instanceGroups.list`
- `compute.instances.delete`
- `compute.instances.list`
- `compute.instances.stop`
- `compute.machineTypes.list`

Example 2.19. Required for deleting storage resources

- `storage.buckets.delete`

- **storage.buckets.getIamPolicy**
- **storage.buckets.list**
- **storage.objects.delete**
- **storage.objects.list**

Example 2.20. Required permissions for deleting health check resources

- **compute.healthChecks.delete**
- **compute.healthChecks.list**
- **compute.httpHealthChecks.delete**
- **compute.httpHealthChecks.list**
- **compute.regionHealthChecks.delete**
- **compute.regionHealthChecks.list**

Example 2.21. Required Images permissions for deletion

- **compute.images.list**

2.5.3. Required GCP permissions for shared VPC installations

When you are installing a cluster to a [Shared VPC](#), you must configure the service account for both the host project and the service project. If you are not installing to a shared VPC, you can skip this section.

You must apply the minimum roles required for a standard installation as listed above, to the service project.



IMPORTANT

You can use granular permissions for a Cloud Credential Operator that operates in either manual or mint credentials mode. You cannot use granular permissions in passthrough credentials mode.

Ensure that the host project applies one of the following configurations to the service account:

Example 2.22. Required permissions for creating firewalls in the host project

- **projects/<host-project>/roles/dns.networks.bindPrivateDNSZone**
- **roles/compute.networkAdmin**
- **roles/compute.securityAdmin**

Example 2.23. Required permissions for deleting firewalls in the host project

- **compute.firewalls.delete**
- **compute.networks.updatePolicy**

Example 2.24. Required minimal permissions

- **projects/<host-project>/roles/dns.networks.bindPrivateDNSZone**
- **roles/compute.networkUser**

If you do not supply a service account for control plane nodes in the **install-config.yaml** file, please grant the below permissions to the service account in the host project. If you do not supply a service account for compute nodes in the **install-config.yaml** file, please grant the below permissions to the service account in the host project for cluster destruction.

- **resourcemanager.projects.getIamPolicy**
- **resourcemanager.projects.setIamPolicy**

2.5.4. Required GCP permissions for user-provided service accounts

When you are installing a cluster, the compute and control plane nodes require their own service accounts. By default, the installation program creates a service account for the control plane and compute nodes. The service account that the installation program uses requires the roles and permissions that are listed in the *Creating a service account in GCP* section, as well as the **resourcemanager.projects.getIamPolicy** and **resourcemanager.projects.setIamPolicy** permissions. These permissions should be applied to the service account in the host project. If this approach does not meet the security requirements of your organization, you can provide a service account email address for the control plane or compute nodes in the **install-config.yaml** file. For more information, see the *Installation configuration parameters for GCP* page. If you provide a service account for control plane nodes during an installation into a shared VPC, you must grant that service account the **roles/compute.networkUser** role in the host project. If you want the installation program to automatically create firewall rules when you supply the control plane service account, you must grant that service account the **roles/compute.networkAdmin** and **roles/compute.securityAdmin** roles in the host project. If you only supply the **roles/compute.networkUser** role, you must create the firewall rules manually.


IMPORTANT

The following roles are required for user-provided service accounts for control plane and compute nodes respectively.

Example 2.25. Required roles for control plane nodes

- **roles/compute.instanceAdmin**
- **roles/compute.networkAdmin**
- **roles/compute.securityAdmin**
- **roles/storage.admin**

Example 2.26. Required roles for compute nodes

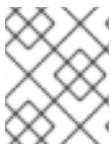
- **roles/compute.viewer**
- **roles/storage.admin**
- **roles/artifactregistry.reader**

2.6. SUPPORTED GCP REGIONS

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

- **africa-south1** (Johannesburg, South Africa)
- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-northeast3** (Seoul, South Korea)
- **asia-south1** (Mumbai, India)
- **asia-south2** (Delhi, India)
- **asia-southeast1** (Jurong West, Singapore)
- **asia-southeast2** (Jakarta, Indonesia)
- **australia-southeast1** (Sydney, Australia)
- **australia-southeast2** (Melbourne, Australia)
- **europe-central2** (Warsaw, Poland)
- **europe-north1** (Hamina, Finland)
- **europe-southwest1** (Madrid, Spain)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)
- **europe-west6** (Zürich, Switzerland)
- **europe-west8** (Milan, Italy)

- **europe-west9** (Paris, France)
- **europe-west12** (Turin, Italy)
- **me-central1** (Doha, Qatar, Middle East)
- **me-central2** (Dammam, Saudi Arabia, Middle East)
- **me-west1** (Tel Aviv, Israel)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **northamerica-northeast2** (Toronto, Ontario, Canada)
- **southamerica-east1** (São Paulo, Brazil)
- **southamerica-west1** (Santiago, Chile)
- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-east5** (Columbus, Ohio)
- **us-south1** (Dallas, Texas)
- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)
- **us-west3** (Salt Lake City, Utah, USA)
- **us-west4** (Las Vegas, Nevada, USA)



NOTE

To determine which machine type instances are available by region and zone, see the Google [documentation](#).

2.7. NEXT STEPS

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

CHAPTER 3. INSTALLING A CLUSTER QUICKLY ON GCP

In OpenShift Container Platform version 4.18, you can install a cluster on Google Cloud Platform (GCP) that uses the default configuration options.

3.1. PREREQUISITES

- You reviewed details about the [OpenShift Container Platform installation and update processes](#).
- You read the documentation on [selecting a cluster installation method and preparing it for users](#).
- You [configured a GCP project](#) to host the cluster.
- If you use a firewall, you [configured it to allow the sites](#) that your cluster requires access to.

3.2. INTERNET ACCESS FOR OPENSIFT CONTAINER PLATFORM

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

You must have internet access to:

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



IMPORTANT

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

3.3. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS

During an OpenShift Container Platform installation, you can provide an SSH public key to the installation program. The key is passed to the Red Hat Enterprise Linux CoreOS (RHCOS) nodes through their Ignition config files and is used to authenticate SSH access to the nodes. The key is added to the `~/.ssh/authorized_keys` list for the `core` user on each node, which enables password-less authentication.

After the key is passed to the nodes, you can use the key pair to SSH in to the RHCOS nodes as the user `core`. To access the nodes through SSH, the private key identity must be managed by SSH for your local user.

If you want to SSH in to your cluster nodes to perform installation debugging or disaster recovery, you must provide the SSH public key during the installation process. The `./openshift-install gather` command also requires the SSH public key to be in place on the cluster nodes.



IMPORTANT

Do not skip this procedure in production environments, where disaster recovery and debugging is required.



NOTE

You must use a local key, not one that you configured with platform-specific approaches.

Procedure

1. If you do not have an existing SSH key pair on your local machine to use for authentication onto your cluster nodes, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N " -f <path>/<file_name> ①
```

- 1 Specify the path and file name, such as `~/.ssh/id_ed25519`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the `x86_64`, `ppc64le`, and `s390x` architectures, do not create a key that uses the `ed25519` algorithm. Instead, create a key that uses the `rsa` or `ecdsa` algorithm.

2. View the public SSH key:

```
$ cat <path>/<file_name>.pub
```

For example, run the following to view the `~/.ssh/id_ed25519.pub` public key:

```
$ cat ~/.ssh/id_ed25519.pub
```

3. Add the SSH private key identity to the SSH agent for your local user, if it has not already been added. SSH agent management of the key is required for password-less SSH authentication onto your cluster nodes, or if you want to use the `./openshift-install gather` command.



NOTE

On some distributions, default SSH private key identities such as `~/.ssh/id_rsa` and `~/.ssh/id_dsa` are managed automatically.

- a. If the `ssh-agent` process is not already running for your local user, start it as a background task:



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

Example output

```
Agent pid 31874
```



NOTE

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

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

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

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_ed25519`

Example output

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

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

3.4. OBTAINING THE INSTALLATION PROGRAM

Before you install OpenShift Container Platform, download the installation file on the host you are using for installation.

Prerequisites

- You have a computer that runs Linux or macOS, with 500 MB of local disk space.

Procedure

- Go to the [Cluster Type](#) page on the Red Hat Hybrid Cloud Console. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

TIP

You can also [download the binaries for a specific OpenShift Container Platform release](#).

- Select your infrastructure provider from the **Run it yourself** section of the page.
- Select your host operating system and architecture from the dropdown menus under **OpenShift Installer** and click **Download Installer**.

4. Place the downloaded file in the directory where you want to 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 the installation program and the files that the installation program creates after you finish installing the cluster. Both of the files are required to delete the cluster.
- Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. To remove your cluster, complete the OpenShift Container Platform uninstallation procedures for your specific cloud provider.

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

```
$ tar -xvf openshift-install-linux.tar.gz
```

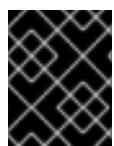
6. Download your installation [pull secret from Red Hat OpenShift Cluster Manager](#). This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

TIP

Alternatively, you can retrieve the installation program from the [Red Hat Customer Portal](#), where you can specify a version of the installation program to download. However, you must have an active subscription to access this page.

3.5. DEPLOYING THE CLUSTER

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- You have verified that the cloud provider account on your host has the correct permissions to deploy the cluster. An account with incorrect permissions causes the installation process to fail with an error message that displays the missing permissions.

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

2. Change to the directory that contains the installation program and initialize the cluster deployment:

```
$ ./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**.

When specifying the directory:

- Verify that the directory has the **execute** permission. This permission is required to run Terraform binaries under the installation directory.
- Use an empty directory. Some installation assets, such as bootstrap X.509 certificates, have short expiration intervals, therefore 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.

3. 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, 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 host, 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 from Red Hat OpenShift Cluster Manager](#).
4. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.
- If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
 - If you included the **Service Account Key Admin** role, you can remove it.

Verification

When the cluster deployment completes successfully:

- The terminal displays directions for accessing your cluster, including a link to the web console and credentials for the **kubeadmin** user.
- Credential information also outputs to `<installation_directory>/openshift_install.log`.



IMPORTANT

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

Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

3.6. INSTALLING THE OPENSHIFT CLI

You can install the OpenShift CLI (**oc**) to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.18. Download and install the new version of **oc**.

Installing the OpenShift CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** drop-down list.
3. Select the appropriate version from the **Version** drop-down list.
4. Click **Download Now** next to the **OpenShift v4.18 Linux Clients** entry and save the file.
5. Unpack the archive:

```
$ tar xvf <file>
```

6. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

Verification

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

```
$ oc <command>
```

Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.18 Windows Client** entry and save the file.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.

To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

Verification

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

```
C:\> oc <command>
```

Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.18 macOS Clients** entry and save the file.



NOTE

For macOS arm64, choose the **OpenShift v4.18 macOS arm64 Client** entry.

4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.

To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

Verification

- Verify your installation by using an **oc** command:

```
$ oc <command>
```

3.7. LOGGING IN TO THE CLUSTER BY USING THE CLI

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

- You deployed an OpenShift Container Platform cluster.
- You installed 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
```

Example output

```
system:admin
```

Additional resources

- See [Accessing the web console](#) for more details about accessing and understanding the OpenShift Container Platform web console.

3.8. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.18, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to [OpenShift Cluster Manager](#).

After you confirm that your [OpenShift Cluster Manager](#) inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See [About remote health monitoring](#) for more information about the Telemetry service

3.9. NEXT STEPS

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

CHAPTER 4. INSTALLING A CLUSTER ON GCP WITH CUSTOMIZATIONS

In OpenShift Container Platform version 4.18, 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.

4.1. PREREQUISITES

- You reviewed details about the [OpenShift Container Platform installation and update processes](#).
- You read the documentation on [selecting a cluster installation method and preparing it for users](#).
- You [configured a GCP project](#) to host the cluster.
- If you use a firewall, you [configured it to allow the sites](#) that your cluster requires access to.

4.2. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM

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

You must have internet access to:

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



IMPORTANT

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

4.3. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS

During an OpenShift Container Platform installation, you can provide an SSH public key to the installation program. The key is passed to the Red Hat Enterprise Linux CoreOS (RHCOS) nodes through their Ignition config files and is used to authenticate SSH access to the nodes. The key is added to the `~/.ssh/authorized_keys` list for the **core** user on each node, which enables password-less authentication.

After the key is passed to the nodes, you can use the key pair to SSH in to the RHCOS nodes as the user **core**. To access the nodes through SSH, the private key identity must be managed by SSH for your local user.

If you want to SSH in to your cluster nodes to perform installation debugging or disaster recovery, you must provide the SSH public key during the installation process. The `./openshift-install gather` command also requires the SSH public key to be in place on the cluster nodes.



IMPORTANT

Do not skip this procedure in production environments, where disaster recovery and debugging is required.



NOTE

You must use a local key, not one that you configured with platform-specific approaches.

Procedure

1. If you do not have an existing SSH key pair on your local machine to use for authentication onto your cluster nodes, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" -f <path>/<file_name> ①
```

1. Specify the path and file name, such as `~/.ssh/id_ed25519`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the **x86_64**, **ppc64le**, and **s390x** architectures, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. View the public SSH key:

```
$ cat <path>/<file_name>.pub
```

For example, run the following to view the `~/.ssh/id_ed25519.pub` public key:

```
$ cat ~/.ssh/id_ed25519.pub
```

3. Add the SSH private key identity to the SSH agent for your local user, if it has not already been added. SSH agent management of the key is required for password-less SSH authentication onto your cluster nodes, or if you want to use the `./openshift-install gather` command.



NOTE

On some distributions, default SSH private key identities such as `~/.ssh/id_rsa` and `~/.ssh/id_dsa` are managed automatically.

- a. If the **ssh-agent** process is not already running for your local user, start it as a background task:

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

Example output

```
Agent pid 31874
```



NOTE

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

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

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

- ① Specify the path and file name for your SSH private key, such as `~/.ssh/id_ed25519`

Example output

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

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

4.4. OBTAINING THE INSTALLATION PROGRAM

Before you install OpenShift Container Platform, download the installation file on the host you are using for installation.

Prerequisites

- You have a computer that runs Linux or macOS, with 500 MB of local disk space.

Procedure

- Go to the [Cluster Type](#) page on the Red Hat Hybrid Cloud Console. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

TIP

You can also [download the binaries for a specific OpenShift Container Platform release](#).

- Select your infrastructure provider from the **Run it yourself** section of the page.
- Select your host operating system and architecture from the dropdown menus under **OpenShift Installer** and click **Download Installer**.

4. Place the downloaded file in the directory where you want to 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 the installation program and the files that the installation program creates after you finish installing the cluster. Both of the files are required to delete the cluster.
- Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. To remove your cluster, complete the OpenShift Container Platform uninstallation procedures for your specific cloud provider.

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

```
$ tar -xvf openshift-install-linux.tar.gz
```

6. Download your installation [pull secret from Red Hat OpenShift Cluster Manager](#). 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.

TIP

Alternatively, you can retrieve the installation program from the [Red Hat Customer Portal](#), where you can specify a version of the installation program to download. However, you must have an active subscription to access this page.

4.5. CREATING THE INSTALLATION CONFIGURATION FILE

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP).

Prerequisites

- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- Configure a GCP account.

Procedure

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

- a. Change to the directory that contains the installation program and 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.

When specifying the directory:

- Verify that the directory has the **execute** permission. This permission is required to run Terraform binaries under the installation directory.
- Use an empty directory. Some installation assets, such as bootstrap X.509 certificates, have short expiration intervals, therefore 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:

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

- Select **gcp** as the platform to target.
- 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.
- Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
- Select the region to deploy the cluster to.
- Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
- Enter a descriptive name for your cluster.

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



NOTE

If you are installing a three-node cluster, be sure to set the **compute.replicas** parameter to **0**. This ensures that the cluster's control planes are schedulable. For more information, see "Installing a three-node cluster on GCP".

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.

Additional resources

- Installation configuration parameters for GCP

4.5.1. Minimum resource requirements for cluster installation

Each cluster machine must meet the following minimum requirements:

Table 4.1. Minimum resource requirements

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

1. One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.
2. OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.
3. As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.



NOTE

For OpenShift Container Platform version 4.18, RHCOS is based on RHEL version 9.4, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see [Architectures](#) (RHEL documentation).

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

Additional resources

- [Optimizing storage](#)

4.5.2. Tested instance types for GCP

The following Google Cloud Platform instance types have been tested with OpenShift Container Platform.



NOTE

Not all instance types are available in all regions and zones. For a detailed breakdown of which instance types are available in which zones, see [regions and zones](#) (Google documentation).

Some instance types require the use of Hyperdisk storage. If you use an instance type that requires Hyperdisk storage, all of the nodes in your cluster must support Hyperdisk storage, and you must change the default storage class to use Hyperdisk storage. For more information, see [machine series support for Hyperdisk](#) (Google documentation). For instructions on modifying storage classes, see the "GCE PersistentDisk (gcePD) object definition" section in the Dynamic Provisioning page in *Storage*.

Example 4.1. Machine series

- A2
- A3
- C2
- C2D
- C3
- C3D
- C4
- E2
- M1
- N1
- N2
- N2D
- N4
- Tau T2D

4.5.3. Tested instance types for GCP on 64-bit ARM infrastructures

The following Google Cloud Platform (GCP) 64-bit ARM instance types have been tested with OpenShift Container Platform.

Example 4.2. Machine series for 64-bit ARM machines

- C4A
- Tau T2A

4.5.4. Using custom machine types

Using a custom machine type to install a OpenShift Container Platform cluster is supported.

Consider the following when using a custom machine type:

- Similar to predefined instance types, custom machine types must meet the minimum resource requirements for control plane and compute machines. For more information, see "Minimum resource requirements for cluster installation".
- The name of the custom machine type must adhere to the following syntax:
custom-<number_of_cpus>-<amount_of_memory_in_mb>

For example, **custom-6-20480**.

As part of the installation process, you specify the custom machine type in the **install-config.yaml** file.

Sample **install-config.yaml** file with a custom machine type

```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    gcp:
      type: custom-6-20480
  replicas: 3
```

4.5.5. Enabling Shielded VMs

You can use Shielded VMs when installing your cluster. Shielded VMs have extra security features including secure boot, firmware and integrity monitoring, and rootkit detection. For more information, see Google's documentation on [Shielded VMs](#).



NOTE

Shielded VMs are currently not supported on clusters with 64-bit ARM infrastructures.

Procedure

- Use a text editor to edit the **install-config.yaml** file prior to deploying your cluster and add one of the following stanzas:

- To use shielded VMs for only control plane machines:

```
controlPlane:  
  platform:  
    gcp:  
      secureBoot: Enabled
```

- To use shielded VMs for only compute machines:

```
compute:  
  - platform:  
    gcp:  
      secureBoot: Enabled
```

- To use shielded VMs for all machines:

```
platform:  
  gcp:  
    defaultMachinePlatform:  
      secureBoot: Enabled
```

4.5.6. Enabling Confidential VMs

You can use Confidential VMs when installing your cluster. Confidential VMs encrypt data while it is being processed. For more information, see Google's documentation on [Confidential Computing](#). You can enable Confidential VMs and Shielded VMs at the same time, although they are not dependent on each other.



NOTE

Confidential VMs are currently not supported on 64-bit ARM architectures.

Procedure

- Use a text editor to edit the **install-config.yaml** file prior to deploying your cluster and add one of the following stanzas:

- To use confidential VMs for only control plane machines:

```
controlPlane:  
  platform:  
    gcp:  
      confidentialCompute: Enabled ①  
      type: n2d-standard-8 ②  
      onHostMaintenance: Terminate ③
```

- - 1 Enable confidential VMs.
 - 2 Specify a machine type that supports Confidential VMs. Confidential VMs require the N2D or C2D series of machine types. For more information on supported machine types, see [Supported operating systems and machine types](#).
 - 3 Specify the behavior of the VM during a host maintenance event, such as a hardware or software update. For a machine that uses Confidential VM, this value must be set to **Terminate**, which stops the VM. Confidential VMs do not support live VM migration.

b. To use confidential VMs for only compute machines:

```
compute:
- platform:
  gcp:
    confidentialCompute: Enabled
    type: n2d-standard-8
    onHostMaintenance: Terminate
```

c. To use confidential VMs for all machines:

```
platform:
gcp:
defaultMachinePlatform:
  confidentialCompute: Enabled
  type: n2d-standard-8
  onHostMaintenance: Terminate
```

4.5.7. 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
credentialsMode: Mint ②
controlPlane: ③ ④
hyperthreading: Enabled ⑤
name: master
platform:
gcp:
  type: n2-standard-4
  zones:
    - us-central1-a
    - us-central1-c
  osDisk:
```

```

diskType: pd-ssd
diskSizeGB: 1024
encryptionKey: 6
kmsKey:
  name: worker-key
  keyRing: test-machine-keys
  location: global
  projectID: project-id
tags: 7
- control-plane-tag1
- control-plane-tag2
osImage: 8
  project: example-project-name
  name: example-image-name
replicas: 3
compute: 9 10
- hyperthreading: Enabled 11
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: 12
        kmsKey:
          name: worker-key
          keyRing: test-machine-keys
          location: global
          projectID: project-id
        tags: 13
        - compute-tag1
        - compute-tag2
      osImage: 14
        project: example-project-name
        name: example-image-name
    replicas: 3
    metadata:
      name: test-cluster 15
    networking:
      clusterNetwork:
        - cidr: 10.128.0.0/14
          hostPrefix: 23
      machineNetwork:
        - cidr: 10.0.0.0/16
      networkType: OVNKubernetes 16
      serviceNetwork:
        - 172.30.0.0/16
    platform:
      gcp:
        projectID: openshift-production 17
        region: us-central1 18

```

```

defaultMachinePlatform:
tags: 19
- global-tag1
- global-tag2
osImage: 20
  project: example-project-name
  name: example-image-name
pullSecret: '{"auths": ...}' 21
tips: false 22
sshKey: ssh-ed25519 AAAA... 23

```

1 15 17 18 21 Required. The installation program prompts you for this value.

2 Optional: Add this parameter to force the Cloud Credential Operator (CCO) to use the specified mode. By default, the CCO uses the root credentials in the **kube-system** namespace to dynamically try to determine the capabilities of the credentials. For details about CCO modes, see the "About the Cloud Credential Operator" section in the *Authentication and authorization* guide.

3 9 If you do not provide these parameters and values, the installation program provides the default value.

4 10 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. Only one control plane pool is used.

5 11 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.

6 12 Optional: The custom encryption key section to encrypt both virtual machines and persistent volumes. Your default compute service account must have the permissions granted to use your KMS key and have the correct IAM role assigned. The default service account name follows the **service-<project_number>@compute-system.iam.gserviceaccount.com** pattern. For more information about granting the correct permissions for your service account, see "Machine management" → "Creating compute machine sets" → "Creating a compute machine set on GCP".

7 13 19 Optional: A set of network tags to apply to the control plane or compute machine sets. The **platform.gcp.defaultMachinePlatform.tags** parameter will apply to both control plane and compute machines. If the **compute.platform.gcp.tags** or **controlPlane.platform.gcp.tags** parameters are set, they override the **platform.gcp.defaultMachinePlatform.tags** parameter.

8 14 20 Optional: A custom Red Hat Enterprise Linux CoreOS (RHCOS) that should be used to boot control plane and compute machines. The **project** and **name** parameters under **platform.gcp.defaultMachinePlatform.osImage** apply to both control plane and compute machines. If the **project** and **name** parameters under **controlPlane.platform.gcp.osImage** or **compute.platform.gcp.osImage** are set, they override the

platform.gcp.defaultMachinePlatform.osImage parameters.

- 16 The cluster network plugin to install. The default value **OVNKubernetes** is the only supported value.
- 22 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.



IMPORTANT

When running Red Hat Enterprise Linux (RHEL) or Red Hat Enterprise Linux CoreOS (RHCOS) booted in FIPS mode, OpenShift Container Platform core components use the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the x86_64, ppc64le, and s390x architectures.

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

Additional resources

- [Enabling customer-managed encryption keys for a compute machine set](#)

4.5.8. 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

- You have an existing **install-config.yaml** file.
- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (169.254.169.254).

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: https://<username>:<pswd>@<ip>:<port> ②
  noProxy: example.com ③
  additionalTrustBundle: | ④
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> ⑤
```

- 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.
- 3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with . to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use * to bypass the proxy for all destinations.
- 4 If provided, the installation program generates a config map 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** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **trustedCA** field of the **Proxy** object. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.
- 5 Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.



NOTE

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

**NOTE**

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

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**.

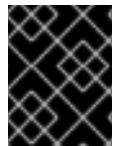
**NOTE**

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

4.6. MANAGING USER-DEFINED LABELS AND TAGS FOR GCP

Google Cloud Platform (GCP) provides labels and tags that help to identify and organize the resources created for a specific OpenShift Container Platform cluster, making them easier to manage.

You can define labels and tags for each GCP resource only during OpenShift Container Platform cluster installation.

**IMPORTANT**

User-defined labels and tags are not supported for OpenShift Container Platform clusters upgraded to OpenShift Container Platform 4.18.

**NOTE**

You cannot update the tags that are already added. Also, a new tag-supported resource creation fails if the configured tag keys or tag values are deleted.

User-defined labels

User-defined labels and OpenShift Container Platform specific labels are applied only to resources created by OpenShift Container Platform installation program and its core components such as:

- GCP filestore CSI Driver Operator
- GCP PD CSI Driver Operator
- Image Registry Operator
- Machine API provider for GCP

User-defined labels are not attached to the resources created by any other Operators or the Kubernetes in-tree components.

User-defined labels and OpenShift Container Platform labels are available on the following GCP resources:

- Compute disk
- Compute forwarding rule
- Compute image
- Compute instance
- DNS managed zone
- Filestore backup
- Filestore instance
- Storage bucket

Limitations to user-defined labels

- Labels for **ComputeAddress** are supported in the GCP beta version. OpenShift Container Platform does not add labels to the resource.

User-defined tags

User-defined tags are applied only to resources created by OpenShift Container Platform installation program and its core components, such as the following resources:

- GCP FileStore CSI Driver Operator
- GCP PD CSI Driver Operator
- Image Registry Operator
- Machine API provider for GCP

User-defined tags are not attached to the resources created by any other Operators or the Kubernetes in-tree components.

User-defined tags are available on the following GCP resources:

- Compute disk
- Compute instance
- Filestore backup
- Filestore instance
- Storage bucket

Limitations to the user-defined tags

- Tags must not be restricted to particular service accounts, because Operators create and use service accounts with minimal roles.
- OpenShift Container Platform does not create any key and value resources of the tag.
- OpenShift Container Platform specific tags are not added to any resource.

Additional resources

- For more information about identifying the **OrganizationID**, see: [OrganizationID](#)
- For more information about identifying the **ProjectID**, see: [ProjectID](#)
- For more information about labels, see [Labels Overview](#).
- For more information about tags, see [Tags Overview](#).

4.6.1. Configuring user-defined labels and tags for GCP

Prerequisites

- The installation program requires that a service account includes a **TagUser** role, so that the program can create the OpenShift Container Platform cluster with defined tags at both organization and project levels.

Procedure

- Update the **install-config.yaml** file to define the list of desired labels and tags.



NOTE

Labels and tags are defined during the **install-config.yaml** creation phase, and cannot be modified or updated with new labels and tags after cluster creation.

Sample **install-config.yaml** file

```
apiVersion: v1
featureSet: TechPreviewNoUpgrade
platform:
  gcp:
    userLabels: ①
      - key: <label_key>②
        value: <label_value>③
    userTags: ④
      - parentID: <OrganizationID/ProjectID>⑤
        key: <tag_key_short_name>
        value: <tag_value_short_name>
```

- ① Adds keys and values as labels to the resources created on GCP.
- ② Defines the label name.
- ③ Defines the label content.
- ④ Adds keys and values as tags to the resources created on GCP.
- ⑤ The ID of the hierarchical resource where the tags are defined, at the organization or the project level.

The following are the requirements for user-defined labels:

- A label key and value must have a minimum of 1 character and can have a maximum of 63 characters.
- A label key and value must contain only lowercase letters, numeric characters, underscore (_), and dash (-).
- A label key must start with a lowercase letter.
- You can configure a maximum of 32 labels per resource. Each resource can have a maximum of 64 labels, and 32 labels are reserved for internal use by OpenShift Container Platform.

The following are the requirements for user-defined tags:

- Tag key and tag value must already exist. OpenShift Container Platform does not create the key and the value.
- A tag **parentID** can be either **OrganizationID** or **ProjectID**:
 - **OrganizationID** must consist of decimal numbers without leading zeros.
 - **ProjectID** must be 6 to 30 characters in length, that includes only lowercase letters, numbers, and hyphens.
 - **ProjectID** must start with a letter, and cannot end with a hyphen.
- A tag key must contain only uppercase and lowercase alphanumeric characters, hyphen (-), underscore (_), and period (.).
- A tag value must contain only uppercase and lowercase alphanumeric characters, hyphen (-), underscore (_), period (.), at sign (@), percent sign (%), equals sign (=), plus (+), colon (:), comma (,), asterisk (*), pound sign (\$), ampersand (&), parentheses (()), square braces ([]), curly braces ({}), and space.
- A tag key and value must begin and end with an alphanumeric character.
- Tag value must be one of the pre-defined values for the key.
- You can configure a maximum of 50 tags.
- There should be no tag key defined with the same value as any of the existing tag keys that will be inherited from the parent resource.

4.6.2. Querying user-defined labels and tags for GCP

After creating the OpenShift Container Platform cluster, you can access the list of the labels and tags defined for the GCP resources in the **infrastructures.config.openshift.io/cluster** object as shown in the following sample **infrastructure.yaml** file.

Sample **infrastructure.yaml** file

```
apiVersion: config.openshift.io/v1
kind: Infrastructure
metadata:
  name: cluster
spec:
  platformSpec:
    type: GCP
```

```

status:
infrastructureName: <cluster_id> ①
platform: GCP
platformStatus:
gcp:
  resourceLabels:
    - key: <label_key>
      value: <label_value>
  resourceTags:
    - key: <tag_key_short_name>
      parentID: <OrganizationID/ProjectID>
      value: <tag_value_short_name>
type: GCP

```

- ① The cluster ID that is generated during cluster installation.

Along with the user-defined labels, resources have a label defined by the OpenShift Container Platform. The format of the OpenShift Container Platform labels is **kubernetes-io-cluster-<cluster_id>:owned**.

4.7. INSTALLING THE OPENSHIFT CLI

You can install the OpenShift CLI (**oc**) to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.18. Download and install the new version of **oc**.

Installing the OpenShift CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** drop-down list.
3. Select the appropriate version from the **Version** drop-down list.
4. Click **Download Now** next to the **OpenShift v4.18 Linux Clients** entry and save the file.
5. Unpack the archive:

```
$ tar xvf <file>
```

6. Place the **oc** binary in a directory that is on your **PATH**. To check your **PATH**, execute the following command:

```
$ echo $PATH
```

Verification

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

```
$ oc <command>
```

Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

- Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
- Select the appropriate version from the **Version** drop-down list.
- Click **Download Now** next to the **OpenShift v4.18 Windows Client** entry and save the file.
- Unzip the archive with a ZIP program.
- Move the **oc** binary to a directory that is on your **PATH**.

To check your **PATH**, open the command prompt and execute the following command:

```
C:> path
```

Verification

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

```
C:> oc <command>
```

Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

- Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
- Select the appropriate version from the **Version** drop-down list.
- Click **Download Now** next to the **OpenShift v4.18 macOS Clients** entry and save the file.



NOTE

For macOS arm64, choose the **OpenShift v4.18 macOS arm64 Client** entry.

- Unpack and unzip the archive.

- Move the **oc** binary to a directory on your **PATH**.

To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

Verification

- Verify your installation by using an **oc** command:

```
$ oc <command>
```

4.8. ALTERNATIVES TO STORING ADMINISTRATOR-LEVEL SECRETS IN THE KUBE-SYSTEM PROJECT

By default, administrator secrets are stored in the **kube-system** project. If you configured the **credentialsMode** parameter in the **install-config.yaml** file to **Manual**, you must use one of the following alternatives:

- To manage long-term cloud credentials manually, follow the procedure in [Manually creating long-term credentials](#).
- To implement short-term credentials that are managed outside the cluster for individual components, follow the procedures in [Configuring a GCP cluster to use short-term credentials](#).

4.8.1. Manually creating long-term credentials

The Cloud Credential Operator (CCO) can be put into manual mode prior to installation in environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace.

Procedure

- Add the following granular permissions to the GCP account that the installation program uses:

Example 4.3. Required GCP permissions

- compute.machineTypes.list
- compute.regions.list
- compute.zones.list
- dns.changes.create
- dns.changes.get
- dns.managedZones.create
- dns.managedZones.delete
- dns.managedZones.get
- dns.managedZones.list
- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.delete

- dns.resourceRecordSets.list
2. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

Sample configuration file snippet

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

3. If you have not previously created installation manifest files, do so by running the following command:

```
$ openshift-install create manifests --dir <installation_directory>
```

where **<installation_directory>** is the directory in which the installation program creates files.

4. Set a **\$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

5. Extract the list of **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
--from=$RELEASE_IMAGE \
--credentials-requests \
--included \① \
--install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \② \
--to=<path_to_directory_for_credentials_requests> \③
```

- ① The **--included** parameter includes only the manifests that your specific cluster configuration requires.
- ② Specify the location of the **install-config.yaml** file.
- ③ Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

This command creates a YAML file for each **CredentialsRequest** object.

Sample CredentialsRequest object

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
...
```

```

spec:
providerSpec:
  apiVersion: cloudcredential.openshift.io/v1
  kind: GCPPowerSpec
  predefinedRoles:
    - roles/storage.admin
    - roles/iam.serviceAccountUser
  skipServiceCheck: true
...

```

6. Create YAML files for secrets in the **openshift-install** manifests directory that you generated previously. The secrets must be stored using the namespace and secret name defined in the **spec.secretRef** for each **CredentialsRequest** object.

Sample CredentialsRequest object with secrets

```

apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
...
spec:
providerSpec:
  apiVersion: cloudcredential.openshift.io/v1
...
secretRef:
  name: <component_secret>
  namespace: <component_namespace>
...

```

Sample Secret object

```

apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  service_account.json: <base64_encoded_gcp_service_account_file>

```



IMPORTANT

Before upgrading a cluster that uses manually maintained credentials, you must ensure that the CCO is in an upgradeable state.

4.8.2. Configuring a GCP cluster to use short-term credentials

To install a cluster that is configured to use GCP Workload Identity, you must configure the CCO utility and create the required GCP resources for your cluster.

4.8.2.1. Configuring the Cloud Credential Operator utility

To create and manage cloud credentials from outside of the cluster when the Cloud Credential Operator (CCO) is operating in manual mode, extract and prepare the CCO utility (**ccctl**) binary.



NOTE

The **ccctl** utility is a Linux binary that must run in a Linux environment.

Prerequisites

- You have access to an OpenShift Container Platform account with cluster administrator access.
- You have installed the OpenShift CLI (**oc**).
- You have added one of the following authentication options to the GCP account that the **ccctl** utility uses:
 - The **IAM Workload Identity Pool Adminrole**
 - The following granular permissions:
 - **compute.projects.get**
 - **iam.googleapis.com/workloadIdentityPoolProviders.create**
 - **iam.googleapis.com/workloadIdentityPoolProviders.get**
 - **iam.googleapis.com/workloadIdentityPools.create**
 - **iam.googleapis.com/workloadIdentityPools.delete**
 - **iam.googleapis.com/workloadIdentityPools.get**
 - **iam.googleapis.com/workloadIdentityPools.undelete**
 - **iam.roles.create**
 - **iam.roles.delete**
 - **iam.roles.list**
 - **iam.roles.undelete**
 - **iam.roles.update**
 - **iam.serviceAccounts.create**
 - **iam.serviceAccounts.delete**
 - **iam.serviceAccounts.getIamPolicy**
 - **iam.serviceAccounts.list**
 - **iam.serviceAccounts.setIamPolicy**
 - **iam.workloadIdentityPoolProviders.get**
 - **iam.workloadIdentityPools.delete**

- **resourcemanager.projects.get**
- **resourcemanager.projects.getIamPolicy**
- **resourcemanager.projects.setIamPolicy**
- **storage.buckets.create**
- **storage.buckets.delete**
- **storage.buckets.get**
- **storage.buckets.getIamPolicy**
- **storage.buckets.setIamPolicy**
- **storage.objects.create**
- **storage.objects.delete**
- **storage.objects.list**

Procedure

1. Set a variable for the OpenShift Container Platform release image by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Obtain the CCO container image from the OpenShift Container Platform release image by running the following command:

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator' $RELEASE_IMAGE -a ~/.pull-secret)
```



NOTE

Ensure that the architecture of the **\$RELEASE_IMAGE** matches the architecture of the environment in which you will use the **ccctl** tool.

3. Extract the **ccctl** binary from the CCO container image within the OpenShift Container Platform release image by running the following command:

```
$ oc image extract $CCO_IMAGE \
--file="/usr/bin/ccctl.<rhel_version>" \①
-a ~/.pull-secret
```

- ① For **<rhel_version>**, specify the value that corresponds to the version of Red Hat Enterprise Linux (RHEL) that the host uses. If no value is specified, **ccctl.rhel8** is used by default. The following values are valid:

- **rhel8**: Specify this value for hosts that use RHEL 8.
- **rhel9**: Specify this value for hosts that use RHEL 9.

4. Change the permissions to make **ccocctl** executable by running the following command:

```
$ chmod 775 ccocctl.<rhel_version>
```

Verification

- To verify that **ccocctl** is ready to use, display the help file. Use a relative file name when you run the command, for example:

```
$ ./ccocctl.rhel9
```

Example output

OpenShift credentials provisioning tool

Usage:

```
ccocctl [command]
```

Available Commands:

aws	Manage credentials objects for AWS cloud
azure	Manage credentials objects for Azure
gcp	Manage credentials objects for Google cloud
help	Help about any command
ibmcloud	Manage credentials objects for {ibm-cloud-title}
nutanix	Manage credentials objects for Nutanix

Flags:

```
-h, --help help for ccocctl
```

Use "ccocctl [command] --help" for more information about a command.

4.8.2.2. Creating GCP resources with the Cloud Credential Operator utility

You can use the **ccocctl gcp create-all** command to automate the creation of GCP resources.



NOTE

By default, **ccocctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccocctl_output_dir>** to refer to this directory.

Prerequisites

You must have:

- Extracted and prepared the **ccocctl** binary.

Procedure

- Set a **\$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
--from=$RELEASE_IMAGE \
--credentials-requests \
--included \①
--install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \②
--to=<path_to_directory_for_credentials_requests> \③
```

- ① The **--included** parameter includes only the manifests that your specific cluster configuration requires.
- ② Specify the location of the **install-config.yaml** file.
- ③ Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.



NOTE

This command might take a few moments to run.

3. Use the **ccctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccctl gcp create-all \
--name=<name> \①
--region=<gcp_region> \②
--project=<gcp_project_id> \③
--credentials-requests-dir=<path_to_credentials_requests_directory> \④
```

- ① Specify the user-defined name for all created GCP resources used for tracking. If you plan to install the GCP Filestore Container Storage Interface (CSI) Driver Operator, retain this value.
- ② Specify the GCP region in which cloud resources will be created.
- ③ Specify the GCP project ID in which cloud resources will be created.
- ④ Specify the directory containing the files of **CredentialsRequest** manifests to create GCP service accounts.



NOTE

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccctl_output_dir>/manifests** directory:

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

Example output

```
cluster-authentication-02-config.yaml
openshift-cloud-controller-manager-gcp-ccm-cloud-credentials-credentials.yaml
openshift-cloud-credential-operator-cloud-credential-operator-gcp-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capg-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-gcp-pd-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-gcp-cloud-credentials-credentials.yaml
```

You can verify that the IAM service accounts are created by querying GCP. For more information, refer to GCP documentation on listing IAM service accounts.

4.8.2.3. Incorporating the Cloud Credential Operator utility manifests

To implement short-term security credentials managed outside the cluster for individual components, you must move the manifest files that the Cloud Credential Operator utility (**ccoctl**) created to the correct directories for the installation program.

Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have configured the Cloud Credential Operator utility (**ccoctl**).
- You have created the cloud provider resources that are required for your cluster with the **ccoctl** utility.

Procedure

1. Add the following granular permissions to the GCP account that the installation program uses:

Example 4.4. Required GCP permissions

- compute.machineTypes.list
- compute.regions.list
- compute.zones.list
- dns.changes.create
- dns.changes.get
- dns.managedZones.create
- dns.managedZones.delete
- dns.managedZones.get
- dns.managedZones.list

- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

2. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

Sample configuration file snippet

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

3. If you have not previously created installation manifest files, do so by running the following command:

```
$ openshift-install create manifests --dir <installation_directory>
```

where **<installation_directory>** is the directory in which the installation program creates files.

4. Copy the manifests that the **ccctl** utility generated to the **manifests** directory that the installation program created by running the following command:

```
$ cp /<path_to_ccctl_output_dir>/manifests/* ./manifests/
```

5. Copy the **tls** directory that contains the private key to the installation directory:

```
$ cp -a /<path_to_ccctl_output_dir>/tls .
```

4.9. USING THE GCP MARKETPLACE OFFERING

Using the GCP Marketplace offering lets you deploy an OpenShift Container Platform cluster, which is billed on pay-per-use basis (hourly, per core) through GCP, while still being supported directly by Red Hat.

By default, the installation program downloads and installs the Red Hat Enterprise Linux CoreOS (RHCOS) image that is used to deploy compute machines. To deploy an OpenShift Container Platform cluster using an RHCOS image from the GCP Marketplace, override the default behavior by modifying the **install-config.yaml** file to reference the location of GCP Marketplace offer.



NOTE

You should only modify the RHCOS image for compute machines to use a GCP Marketplace image. Control plane machines and infrastructure nodes do not require an OpenShift Container Platform subscription and use the public RHCOS default image by default, which does not incur subscription costs on your GCP bill. Therefore, you should not modify the cluster default boot image or the control plane boot images. Applying the GCP Marketplace image to them will incur additional licensing costs that cannot be recovered.

Prerequisites

- You have an existing **install-config.yaml** file.

Procedure

1. Edit the **compute.platform.gcp.osImage** parameters to specify the location of the GCP Marketplace image:

- Set the **project** parameter to **redhat-marketplace-public**
- Set the **name** parameter to one of the following offers:

OpenShift Container Platform

redhat-coreos-ocp-413-x86-64-202305021736

OpenShift Platform Plus

redhat-coreos-opp-413-x86-64-202305021736

OpenShift Kubernetes Engine

redhat-coreos-oke-413-x86-64-202305021736

2. Save the file and reference it when deploying the cluster.

Sample **install-config.yaml** file that specifies a GCP Marketplace image for compute machines

```
apiVersion: v1
baseDomain: example.com
controlPlane:
# ...
compute:
platform:
gcp:
osImage:
  project: redhat-marketplace-public
  name: redhat-coreos-ocp-413-x86-64-202305021736
# ...
```

4.10. DEPLOYING THE CLUSTER

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- You have verified that the cloud provider account on your host has the correct permissions to deploy the cluster. An account with incorrect permissions causes the installation process to fail with an error message that displays the missing permissions.

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

2. Change to the directory that contains the installation program and initialize the cluster deployment:

```
$ ./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**.

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.

- If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
- If you included the **Service Account Key Admin** role, you can remove it.

Verification

When the cluster deployment completes successfully:

- The terminal displays directions for accessing your cluster, including a link to the web console and credentials for the **kubeadmin** user.
- Credential information also outputs to **<installation_directory>/openshift_install.log**.



IMPORTANT

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

Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

4.11. LOGGING IN TO THE CLUSTER BY USING THE CLI

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

- You deployed an OpenShift Container Platform cluster.
- You installed 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
```

Example output

```
system:admin
```

Additional resources

- See [Accessing the web console](#) for more details about accessing and understanding the OpenShift Container Platform web console.

4.12. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.18, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to [OpenShift Cluster Manager](#).

After you confirm that your [OpenShift Cluster Manager](#) inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See [About remote health monitoring](#) for more information about the Telemetry service

4.13. NEXT STEPS

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

CHAPTER 5. INSTALLING A CLUSTER ON GCP WITH NETWORK CUSTOMIZATIONS

In OpenShift Container Platform version 4.18, you can install a cluster with a customized network configuration on infrastructure that the installation program provisions on Google Cloud Platform (GCP). 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. To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

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

5.1. PREREQUISITES

- You reviewed details about the [OpenShift Container Platform installation and update processes](#).
- You read the documentation on [selecting a cluster installation method and preparing it for users](#).
- You [configured a GCP project](#) to host the cluster.
- If you use a firewall, you [configured it to allow the sites](#) that your cluster requires access to.

5.2. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM

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

You must have internet access to:

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



IMPORTANT

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

5.3. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS

During an OpenShift Container Platform installation, you can provide an SSH public key to the installation program. The key is passed to the Red Hat Enterprise Linux CoreOS (RHCOS) nodes through their Ignition config files and is used to authenticate SSH access to the nodes. The key is added

to the `~/.ssh/authorized_keys` list for the **core** user on each node, which enables password-less authentication.

After the key is passed to the nodes, you can use the key pair to SSH in to the RHCOS nodes as the user **core**. To access the nodes through SSH, the private key identity must be managed by SSH for your local user.

If you want to SSH in to your cluster nodes to perform installation debugging or disaster recovery, you must provide the SSH public key during the installation process. The `./openshift-install gather` command also requires the SSH public key to be in place on the cluster nodes.



IMPORTANT

Do not skip this procedure in production environments, where disaster recovery and debugging is required.



NOTE

You must use a local key, not one that you configured with platform-specific approaches.

Procedure

1. If you do not have an existing SSH key pair on your local machine to use for authentication onto your cluster nodes, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" -f <path>/<file_name>
```

①

- 1 Specify the path and file name, such as `~/.ssh/id_ed25519`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the **x86_64**, **ppc64le**, and **s390x** architectures, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. View the public SSH key:

```
$ cat <path>/<file_name>.pub
```

For example, run the following to view the `~/.ssh/id_ed25519.pub` public key:

```
$ cat ~/.ssh/id_ed25519.pub
```

3. Add the SSH private key identity to the SSH agent for your local user, if it has not already been added. SSH agent management of the key is required for password-less SSH authentication onto your cluster nodes, or if you want to use the `./openshift-install gather` command.



NOTE

On some distributions, default SSH private key identities such as `~/.ssh/id_rsa` and `~/.ssh/id_dsa` are managed automatically.

- a. If the **ssh-agent** process is not already running for your local user, start it as a background task:

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

Example output

```
Agent pid 31874
```



NOTE

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

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

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

- ① Specify the path and file name for your SSH private key, such as `~/.ssh/id_ed25519`

Example output

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

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

5.4. OBTAINING THE INSTALLATION PROGRAM

Before you install OpenShift Container Platform, download the installation file on the host you are using for installation.

Prerequisites

- You have a computer that runs Linux or macOS, with 500 MB of local disk space.

Procedure

1. Go to the [Cluster Type](#) page on the Red Hat Hybrid Cloud Console. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

TIP

You can also [download the binaries for a specific OpenShift Container Platform release](#).

2. Select your infrastructure provider from the **Run it yourself** section of the page.
3. Select your host operating system and architecture from the dropdown menus under **OpenShift Installer** and click **Download Installer**.
4. Place the downloaded file in the directory where you want to 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 the installation program and the files that the installation program creates after you finish installing the cluster. Both of the files are required to delete the cluster.
- Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. To remove your cluster, complete the OpenShift Container Platform uninstallation procedures for your specific cloud provider.

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

```
$ tar -xvf openshift-install-linux.tar.gz
```

6. Download your installation [pull secret from Red Hat OpenShift Cluster Manager](#). 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.

TIP

Alternatively, you can retrieve the installation program from the [Red Hat Customer Portal](#), where you can specify a version of the installation program to download. However, you must have an active subscription to access this page.

5.5. CREATING THE INSTALLATION CONFIGURATION FILE

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP).

Prerequisites

- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- Configure a GCP account.

Procedure

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

- Change to the directory that contains the installation program and 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.

When specifying the directory:

- Verify that the directory has the **execute** permission. This permission is required to run Terraform binaries under the installation directory.
- Use an empty directory. Some installation assets, such as bootstrap X.509 certificates, have short expiration intervals, therefore 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, specify an SSH key that your **ssh-agent** process uses.

- Select **gcp** as the platform to target.
- 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.
- Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
- Select the region to deploy the cluster to.
- Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
- Enter a descriptive name for your cluster.

- 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.

Additional resources

- [Installation configuration parameters for GCP](#)

5.5.1. Minimum resource requirements for cluster installation

Each cluster machine must meet the following minimum requirements:

Table 5.1. Minimum resource requirements

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

1. One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.
2. OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.
3. As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.



NOTE

For OpenShift Container Platform version 4.18, RHCOS is based on RHEL version 9.4, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see [Architectures](#) (RHEL documentation).

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

Additional resources

- [Optimizing storage](#)

5.5.2. Tested instance types for GCP

The following Google Cloud Platform instance types have been tested with OpenShift Container Platform.



NOTE

Not all instance types are available in all regions and zones. For a detailed breakdown of which instance types are available in which zones, see [regions and zones](#) (Google documentation).

Some instance types require the use of Hyperdisk storage. If you use an instance type that requires Hyperdisk storage, all of the nodes in your cluster must support Hyperdisk storage, and you must change the default storage class to use Hyperdisk storage. For more information, see [machine series support for Hyperdisk](#) (Google documentation). For instructions on modifying storage classes, see the "GCE PersistentDisk (gcePD) object definition" section in the Dynamic Provisioning page in [Storage](#).

Example 5.1. Machine series

- **A2**
- **A3**
- **C2**
- **C2D**
- **C3**
- **C3D**

- C4
- E2
- M1
- N1
- N2
- N2D
- N4
- Tau T2D

5.5.3. Tested instance types for GCP on 64-bit ARM infrastructures

The following Google Cloud Platform (GCP) 64-bit ARM instance types have been tested with OpenShift Container Platform.

Example 5.2. Machine series for 64-bit ARM machines

- C4A
- Tau T2A

5.5.4. Using custom machine types

Using a custom machine type to install a OpenShift Container Platform cluster is supported.

Consider the following when using a custom machine type:

- Similar to predefined instance types, custom machine types must meet the minimum resource requirements for control plane and compute machines. For more information, see "Minimum resource requirements for cluster installation".
- The name of the custom machine type must adhere to the following syntax:
custom-<number_of_cpus>-<amount_of_memory_in_mb>

For example, **custom-6-20480**.

As part of the installation process, you specify the custom machine type in the **install-config.yaml** file.

Sample **install-config.yaml** file with a custom machine type

```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
```

```

replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    gcp:
      type: custom-6-20480
replicas: 3

```

5.5.5. Enabling Shielded VMs

You can use Shielded VMs when installing your cluster. Shielded VMs have extra security features including secure boot, firmware and integrity monitoring, and rootkit detection. For more information, see Google's documentation on [Shielded VMs](#).



NOTE

Shielded VMs are currently not supported on clusters with 64-bit ARM infrastructures.

Procedure

- Use a text editor to edit the **install-config.yaml** file prior to deploying your cluster and add one of the following stanzas:
 - To use shielded VMs for only control plane machines:

```

controlPlane:
  platform:
    gcp:
      secureBoot: Enabled

```

- To use shielded VMs for only compute machines:

```

compute:
  - platform:
    gcp:
      secureBoot: Enabled

```

- To use shielded VMs for all machines:

```

platform:
  gcp:
    defaultMachinePlatform:
      secureBoot: Enabled

```

5.5.6. Enabling Confidential VMs

You can use Confidential VMs when installing your cluster. Confidential VMs encrypt data while it is being processed. For more information, see Google's documentation on [Confidential Computing](#). You can enable Confidential VMs and Shielded VMs at the same time, although they are not dependent on each other.

**NOTE**

Confidential VMs are currently not supported on 64-bit ARM architectures.

Procedure

- Use a text editor to edit the **install-config.yaml** file prior to deploying your cluster and add one of the following stanzas:

- To use confidential VMs for only control plane machines:

```
controlPlane:  
  platform:  
    gcp:  
      confidentialCompute: Enabled ①  
      type: n2d-standard-8 ②  
      onHostMaintenance: Terminate ③
```

- ① Enable confidential VMs.
- ② Specify a machine type that supports Confidential VMs. Confidential VMs require the N2D or C2D series of machine types. For more information on supported machine types, see [Supported operating systems and machine types](#).
- ③ Specify the behavior of the VM during a host maintenance event, such as a hardware or software update. For a machine that uses Confidential VM, this value must be set to **Terminate**, which stops the VM. Confidential VMs do not support live VM migration.

- To use confidential VMs for only compute machines:

```
compute:  
  - platform:  
    gcp:  
      confidentialCompute: Enabled  
      type: n2d-standard-8  
      onHostMaintenance: Terminate
```

- To use confidential VMs for all machines:

```
platform:  
  gcp:  
    defaultMachinePlatform:  
      confidentialCompute: Enabled  
      type: n2d-standard-8  
      onHostMaintenance: Terminate
```

5.5.7. 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
credentialsMode: Mint 2
controlPlane: 3 4
hyperthreading: Enabled 5
name: master
platform:
  gcp:
    type: n2-standard-4
    zones:
      - us-central1-a
      - us-central1-c
    osDisk:
      diskType: pd-ssd
      diskSizeGB: 1024
      encryptionKey: 6
      kmsKey:
        name: worker-key
        keyRing: test-machine-keys
        location: global
        projectId: project-id
    tags: 7
      - control-plane-tag1
      - control-plane-tag2
    osImage: 8
      project: example-project-name
      name: example-image-name
replicas: 3
compute: 9 10
- hyperthreading: Enabled 11
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: 12
        kmsKey:
          name: worker-key
          keyRing: test-machine-keys
          location: global
          projectId: project-id
    tags: 13
      - compute-tag1
      - compute-tag2

```

```

osImage: ⑯
  project: example-project-name
  name: example-image-name
replicas: 3
metadata:
  name: test-cluster ⑰
networking: ⑯
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  machineNetwork:
    - cidr: 10.0.0.0/16
  networkType: OVNKubernetes ⑯
  serviceNetwork:
    - 172.30.0.0/16
platform:
  gcp:
    projectId: openshift-production ⑯
    region: us-central1 ⑯
    defaultMachinePlatform:
      tags: ⑯
        - global-tag1
        - global-tag2
      osImage: ⑯
        project: example-project-name
        name: example-image-name
    pullSecret: '{"auths": ...}' ⑯
    fips: false ⑯
    sshKey: ssh-ed25519 AAAA... ⑯

```

① ⑮ ⑯ ⑯ ⑯ Required. The installation program prompts you for this value.

② Optional: Add this parameter to force the Cloud Credential Operator (CCO) to use the specified mode. By default, the CCO uses the root credentials in the **kube-system** namespace to dynamically try to determine the capabilities of the credentials. For details about CCO modes, see the "About the Cloud Credential Operator" section in the *Authentication and authorization* guide.

③ ⑨ ⑯ If you do not provide these parameters and values, the installation program provides the default value.

④ ⑩ 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. Only one control plane pool is used.

⑤ ⑪ 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.

6 12 Optional: The custom encryption key section to encrypt both virtual machines and persistent volumes. Your default compute service account must have the permissions granted to use your KMS key and have the correct IAM role assigned. The default service account name follows the **service-<project_number>@compute-system.iam.gserviceaccount.com** pattern. For more information about granting the correct permissions for your service account, see "Machine management" → "Creating compute machine sets" → "Creating a compute machine set on GCP".

7 13 20 Optional: A set of network tags to apply to the control plane or compute machine sets. The **platform.gcp.defaultMachinePlatform.tags** parameter will apply to both control plane and compute machines. If the **compute.platform.gcp.tags** or **controlPlane.platform.gcp.tags** parameters are set, they override the **platform.gcp.defaultMachinePlatform.tags** parameter.

8 14 21 Optional: A custom Red Hat Enterprise Linux CoreOS (RHCOS) that should be used to boot control plane and compute machines. The **project** and **name** parameters under **platform.gcp.defaultMachinePlatform.osImage** apply to both control plane and compute machines. If the **project** and **name** parameters under **controlPlane.platform.gcp.osImage** or **compute.platform.gcp.osImage** are set, they override the **platform.gcp.defaultMachinePlatform.osImage** parameters.

17 The cluster network plugin to install. The default value **OVNKubernetes** is the only supported value.

23 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.



IMPORTANT

When running Red Hat Enterprise Linux (RHEL) or Red Hat Enterprise Linux CoreOS (RHCOS) booted in FIPS mode, OpenShift Container Platform core components use the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the x86_64, ppc64le, and s390x architectures.

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

Additional resources

- [Enabling customer-managed encryption keys for a compute machine set](#)

5.5.8. 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

- You have an existing **install-config.yaml** file.
- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

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: https://<username>:<pswd>@<ip>:<port> ②
  noProxy: example.com ③
  additionalTrustBundle: | ④
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> ⑤
```

- 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.
- 3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass the proxy for all destinations.
- 4 If provided, the installation program generates a config map 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** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **trustedCA** field of the **Proxy** object. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

- 5 Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.



NOTE

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



NOTE

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

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.

5.6. INSTALLING THE OPENSHIFT CLI

You can install the OpenShift CLI (**oc**) to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.18. Download and install the new version of **oc**.

Installing the OpenShift CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** drop-down list.

3. Select the appropriate version from the **Version** drop-down list.
4. Click **Download Now** next to the **OpenShift v4.18 Linux Clients** entry and save the file.
5. Unpack the archive:

```
$ tar xvf <file>
```

6. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

Verification

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

```
$ oc <command>
```

Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.18 Windows Client** entry and save the file.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

Verification

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

```
C:\> oc <command>
```

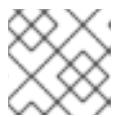
Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.

3. Click **Download Now** next to the **OpenShift v4.18 macOS Clients** entry and save the file.

**NOTE**

For macOS arm64, choose the **OpenShift v4.18 macOS arm64 Client** entry.

4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

Verification

- Verify your installation by using an **oc** command:

```
$ oc <command>
```

5.7. ALTERNATIVES TO STORING ADMINISTRATOR-LEVEL SECRETS IN THE KUBE-SYSTEM PROJECT

By default, administrator secrets are stored in the **kube-system** project. If you configured the **credentialsMode** parameter in the **install-config.yaml** file to **Manual**, you must use one of the following alternatives:

- To manage long-term cloud credentials manually, follow the procedure in [Manually creating long-term credentials](#).
- To implement short-term credentials that are managed outside the cluster for individual components, follow the procedures in [Configuring a GCP cluster to use short-term credentials](#).

5.7.1. Manually creating long-term credentials

The Cloud Credential Operator (CCO) can be put into manual mode prior to installation in environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace.

Procedure

1. Add the following granular permissions to the GCP account that the installation program uses:

Example 5.3. Required GCP permissions

- compute.machineTypes.list
- compute.regions.list
- compute.zones.list
- dns.changes.create
- dns.changes.get

- dns.managedZones.create
- dns.managedZones.delete
- dns.managedZones.get
- dns.managedZones.list
- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

2. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

Sample configuration file snippet

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

3. If you have not previously created installation manifest files, do so by running the following command:

```
$ openshift-install create manifests --dir <installation_directory>
```

where **<installation_directory>** is the directory in which the installation program creates files.

4. Set a **\$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

5. Extract the list of **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
--from=$RELEASE_IMAGE \
--credentials-requests \
--included ① \
--install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
--to=<path_to_directory_for_credentials_requests> ② ③
```

① The **--included** parameter includes only the manifests that your specific cluster configuration requires.

② Specify the location of the **install-config.yaml** file.

- 3** Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

This command creates a YAML file for each **CredentialsRequest** object.

Sample CredentialsRequest object

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: GCPProviderSpec
    predefinedRoles:
      - roles/storage.admin
      - roles/iam.serviceAccountUser
    skipServiceCheck: true
...
```

6. Create YAML files for secrets in the **openshift-install** manifests directory that you generated previously. The secrets must be stored using the namespace and secret name defined in the **spec.secretRef** for each **CredentialsRequest** object.

Sample CredentialsRequest object with secrets

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
...
```

Sample Secret object

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  service_account.json: <base64_encoded_gcp_service_account_file>
```



IMPORTANT

Before upgrading a cluster that uses manually maintained credentials, you must ensure that the CCO is in an upgradeable state.

5.7.2. Configuring a GCP cluster to use short-term credentials

To install a cluster that is configured to use GCP Workload Identity, you must configure the CCO utility and create the required GCP resources for your cluster.

5.7.2.1. Configuring the Cloud Credential Operator utility

To create and manage cloud credentials from outside of the cluster when the Cloud Credential Operator (CCO) is operating in manual mode, extract and prepare the CCO utility (**ccctl**) binary.



NOTE

The **ccctl** utility is a Linux binary that must run in a Linux environment.

Prerequisites

- You have access to an OpenShift Container Platform account with cluster administrator access.
- You have installed the OpenShift CLI (**oc**).
- You have added one of the following authentication options to the GCP account that the **ccctl** utility uses:
 - The **IAM Workload Identity Pool Adminrole**
 - The following granular permissions:
 - **compute.projects.get**
 - **iam.googleapis.com/workloadIdentityPoolProviders.create**
 - **iam.googleapis.com/workloadIdentityPoolProviders.get**
 - **iam.googleapis.com/workloadIdentityPools.create**
 - **iam.googleapis.com/workloadIdentityPools.delete**
 - **iam.googleapis.com/workloadIdentityPools.get**
 - **iam.googleapis.com/workloadIdentityPools.undelete**
 - **iam.roles.create**
 - **iam.roles.delete**
 - **iam.roles.list**
 - **iam.roles.undelete**
 - **iam.roles.update**

- **iam.serviceAccounts.create**
- **iam.serviceAccounts.delete**
- **iam.serviceAccounts.getIamPolicy**
- **iam.serviceAccounts.list**
- **iam.serviceAccounts.setIamPolicy**
- **iam.workloadIdentityPoolProviders.get**
- **iam.workloadIdentityPools.delete**
- **resourcemanager.projects.get**
- **resourcemanager.projects.getIamPolicy**
- **resourcemanager.projects.setIamPolicy**
- **storage.buckets.create**
- **storage.buckets.delete**
- **storage.buckets.get**
- **storage.buckets.getIamPolicy**
- **storage.buckets.setIamPolicy**
- **storage.objects.create**
- **storage.objects.delete**
- **storage.objects.list**

Procedure

1. Set a variable for the OpenShift Container Platform release image by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Obtain the CCO container image from the OpenShift Container Platform release image by running the following command:

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'  
$RELEASE_IMAGE -a ~/pull-secret)
```



NOTE

Ensure that the architecture of the **\$RELEASE_IMAGE** matches the architecture of the environment in which you will use the **ccctl** tool.

3. Extract the **ccocctl** binary from the CCO container image within the OpenShift Container Platform release image by running the following command:

```
$ oc image extract $CCO_IMAGE \
--file="/usr/bin/ccocctl.<rhel_version>" \①
-a ~/.pull-secret
```

- 1 For **<rhel_version>**, specify the value that corresponds to the version of Red Hat Enterprise Linux (RHEL) that the host uses. If no value is specified, **ccocctl.rhel8** is used by default. The following values are valid:

- **rhel8**: Specify this value for hosts that use RHEL 8.
- **rhel9**: Specify this value for hosts that use RHEL 9.

4. Change the permissions to make **ccocctl** executable by running the following command:

```
$ chmod 775 ccocctl.<rhel_version>
```

Verification

- To verify that **ccocctl** is ready to use, display the help file. Use a relative file name when you run the command, for example:

```
$ ./ccocctl.rhel9
```

Example output

```
OpenShift credentials provisioning tool

Usage:
  ccocctl [command]

Available Commands:
  aws      Manage credentials objects for AWS cloud
  azure    Manage credentials objects for Azure
  gcp      Manage credentials objects for Google cloud
  help     Help about any command
  ibmcloud Manage credentials objects for {ibm-cloud-title}
  nutanix   Manage credentials objects for Nutanix

Flags:
  -h, --help  help for ccocctl

Use "ccocctl [command] --help" for more information about a command.
```

5.7.2.2. Creating GCP resources with the Cloud Credential Operator utility

You can use the **ccocctl gcp create-all** command to automate the creation of GCP resources.



NOTE

By default, **ccctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccctl_output_dir>** to refer to this directory.

Prerequisites

You must have:

- Extracted and prepared the **ccctl** binary.

Procedure

- Set a **\$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

- Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
--from=$RELEASE_IMAGE \
--credentials-requests \
--included \① \
--install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \② \
--to=<path_to_directory_for_credentials_requests> \③
```

1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

2 Specify the location of the **install-config.yaml** file.

3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.



NOTE

This command might take a few moments to run.

- Use the **ccctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccctl gcp create-all \
--name=<name> \① \
--region=<gcp_region> \② \
--project=<gcp_project_id> \③ \
--credentials-requests-dir=<path_to_credentials_requests_directory> \④
```

- 1** Specify the user-defined name for all created GCP resources used for tracking. If you plan to install the GCP Filestore Container Storage Interface (CSI) Driver Operator, retain this value.
- 2** Specify the GCP region in which cloud resources will be created.
- 3** Specify the GCP project ID in which cloud resources will be created.
- 4** Specify the directory containing the files of **CredentialsRequest** manifests to create GCP service accounts.

**NOTE**

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

Example output

```
cluster-authentication-02-config.yaml
openshift-cloud-controller-manager-gcp-ccm-cloud-credentials-credentials.yaml
openshift-cloud-credential-operator-cloud-credential-operator-gcp-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capg-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-gcp-pd-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-gcp-cloud-credentials-credentials.yaml
```

You can verify that the IAM service accounts are created by querying GCP. For more information, refer to GCP documentation on listing IAM service accounts.

5.7.2.3. Incorporating the Cloud Credential Operator utility manifests

To implement short-term security credentials managed outside the cluster for individual components, you must move the manifest files that the Cloud Credential Operator utility (**ccoctl**) created to the correct directories for the installation program.

Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have configured the Cloud Credential Operator utility (**ccoctl**).
- You have created the cloud provider resources that are required for your cluster with the **ccoctl** utility.

Procedure

1. Add the following granular permissions to the GCP account that the installation program uses:

Example 5.4. Required GCP permissions

- compute.machineTypes.list
- compute.regions.list
- compute.zones.list
- dns.changes.create
- dns.changes.get
- dns.managedZones.create
- dns.managedZones.delete
- dns.managedZones.get
- dns.managedZones.list
- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

2. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

Sample configuration file snippet

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

3. If you have not previously created installation manifest files, do so by running the following command:

```
$ openshift-install create manifests --dir <installation_directory>
```

where **<installation_directory>** is the directory in which the installation program creates files.

4. Copy the manifests that the **ccctl** utility generated to the **manifests** directory that the installation program created by running the following command:

```
$ cp /<path_to_ccctl_output_dir>/manifests/* ./manifests/
```

5. Copy the **tls** directory that contains the private key to the installation directory:

■

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

5.8. NETWORK CONFIGURATION PHASES

There are two phases prior to OpenShift Container Platform installation where you can customize the network configuration.

Phase 1

You can customize the following network-related fields in the **install-config.yaml** file before you create the manifest files:

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**
- **nodeNetworking**

For more information, see "Installation configuration parameters".



NOTE

Set the **networking.machineNetwork** to match the Classless Inter-Domain Routing (CIDR) where the preferred subnet is located.



IMPORTANT

The CIDR range **172.17.0.0/16** is reserved by **libVirt**. You cannot use any other CIDR range that overlaps with the **172.17.0.0/16** CIDR range for networks in your cluster.

Phase 2

After creating the manifest files by running **openshift-install create manifests**, you can define a customized Cluster Network Operator manifest with only the fields you want to modify. You can use the manifest to specify an advanced network configuration.

During phase 2, you cannot override the values that you specified in phase 1 in the **install-config.yaml** file. However, you can customize the network plugin during phase 2.

5.9. SPECIFYING ADVANCED NETWORK CONFIGURATION

You can use advanced network configuration for your network plugin to integrate your cluster into your existing network environment.

You can specify advanced network configuration only before you install the cluster.



IMPORTANT

Customizing your network configuration by modifying the OpenShift Container Platform manifest files created by the installation program is not supported. Applying a manifest file that you create, as in the following procedure, is supported.

Prerequisites

- You have created the **install-config.yaml** file and completed any modifications to it.

Procedure

1. Change to the directory that contains the installation program and create the manifests:

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

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

2. Create a stub manifest file for the advanced network configuration that is named **cluster-network-03-config.yaml** in the **<installation_directory>/manifests**/ directory:

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

3. Specify the advanced network configuration for your cluster in the **cluster-network-03-config.yaml** file, such as in the following example:

Enable IPsec for the OVN-Kubernetes network provider

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig:
        mode: Full
```

4. Optional: Back up the **manifests/cluster-network-03-config.yaml** file. The installation program consumes the **manifests**/ directory when you create the Ignition config files.
5. Remove the Kubernetes manifest files that define the control plane machines and compute **MachineSets**:

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

Because you create and manage these resources yourself, you do not have to initialize them.

- You can preserve the **MachineSet** files to create compute machines by using the machine API, but you must update references to them to match your environment.

5.10. CLUSTER NETWORK OPERATOR CONFIGURATION

The configuration for the cluster network is specified as part of the Cluster Network Operator (CNO) configuration and stored in a custom resource (CR) object that is named **cluster**. The CR specifies the fields for the **Network** API in the **operator.openshift.io** API group.

The CNO configuration inherits the following fields during cluster installation from the **Network** API in the **Network.config.openshift.io** API group:

clusterNetwork

IP address pools from which pod IP addresses are allocated.

serviceNetwork

IP address pool for services.

defaultNetwork.type

Cluster network plugin. **OVNKubernetes** is the only supported plugin during installation.

You can specify the cluster network plugin configuration for your cluster by setting the fields for the **defaultNetwork** object in the CNO object named **cluster**.

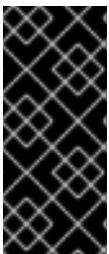
5.10.1. Cluster Network Operator configuration object

The fields for the Cluster Network Operator (CNO) are described in the following table:

Table 5.2. Cluster Network Operator configuration object

Field	Type	Description
metadata.name	string	The name of the CNO object. This name is always cluster .
spec.clusterNetwork	array	A list specifying the blocks of IP addresses from which pod IP addresses are allocated and the subnet prefix length assigned to each individual node in the cluster. For example: <pre>spec: clusterNetwork: - cidr: 10.128.0.0/19 hostPrefix: 23 - cidr: 10.128.32.0/19 hostPrefix: 23</pre>
spec.serviceNetwork	array	A block of IP addresses for services. The OVN-Kubernetes network plugin supports only a single IP address block for the service network. For example: <pre>spec: serviceNetwork: - 172.30.0.0/14</pre> You can customize this field only in the install-config.yaml file before you create the manifests. The value is read-only in the manifest file.

Field	Type	Description
spec.defaultNetwork	object	Configures the network plugin for the cluster network.
spec.kubeProxyConfig	object	The fields for this object specify the kube-proxy configuration. If you are using the OVN-Kubernetes cluster network plugin, the kube-proxy configuration has no effect.



IMPORTANT

For a cluster that needs to deploy objects across multiple networks, ensure that you specify the same value for the **clusterNetwork.hostPrefix** parameter for each network type that is defined in the **install-config.yaml** file. Setting a different value for each **clusterNetwork.hostPrefix** parameter can impact the OVN-Kubernetes network plugin, where the plugin cannot effectively route object traffic among different nodes.

defaultNetwork object configuration

The values for the **defaultNetwork** object are defined in the following table:

Table 5.3. **defaultNetwork** object

Field	Type	Description
type	string	<p>OVNKubernetes. The Red Hat OpenShift Networking network plugin is selected during installation. This value cannot be changed after cluster installation.</p> <p> NOTE</p> <p>OpenShift Container Platform uses the OVN-Kubernetes network plugin by default.</p>
ovnKubernetesConfig	object	This object is only valid for the OVN-Kubernetes network plugin.

Configuration for the OVN-Kubernetes network plugin

The following table describes the configuration fields for the OVN-Kubernetes network plugin:

Table 5.4. **ovnKubernetesConfig** object

Field	Type	Description

Field	Type	Description
mtu	integer	<p>The maximum transmission unit (MTU) for the Geneve (Generic Network Virtualization Encapsulation) overlay network. This is detected automatically based on the MTU of the primary network interface. You do not normally need to override the detected MTU.</p> <p>If the auto-detected value is not what you expect it to be, confirm that the MTU on the primary network interface on your nodes is correct. You cannot use this option to change the MTU value of the primary network interface on the nodes.</p> <p>If your cluster requires different MTU values for different nodes, you must set this value to 100 less than the lowest MTU value in your cluster. For example, if some nodes in your cluster have an MTU of 9001, and some have an MTU of 1500, you must set this value to 1400.</p>
genevePort	integer	The port to use for all Geneve packets. The default value is 6081 . This value cannot be changed after cluster installation.
ipsecConfig	object	Specify a configuration object for customizing the IPsec configuration.
ipv4	object	Specifies a configuration object for IPv4 settings.
ipv6	object	Specifies a configuration object for IPv6 settings.
policyAuditConfig	object	Specify a configuration object for customizing network policy audit logging. If unset, the defaults audit log settings are used.
gatewayConfig	object	<p>Optional: Specify a configuration object for customizing how egress traffic is sent to the node gateway. Valid values are Shared and Local. The default value is Shared. In the default setting, the Open vSwitch (OVS) outputs traffic directly to the node IP interface. In the Local setting, it traverses the host network; consequently, it gets applied to the routing table of the host.</p>  <p>NOTE</p> <p>While migrating egress traffic, you can expect some disruption to workloads and service traffic until the Cluster Network Operator (CNO) successfully rolls out the changes.</p>

Table 5.5. **ovnKubernetesConfig.ipv4** object

Field	Type	Description
internalTransitSwitchSubnet	string	<p>If your existing network infrastructure overlaps with the 100.88.0.0/16 IPv4 subnet, you can specify a different IP address range for internal use by OVN-Kubernetes. The subnet for the distributed transit switch that enables east-west traffic. This subnet cannot overlap with any other subnets used by OVN-Kubernetes or on the host itself. It must be large enough to accommodate one IP address per node in your cluster.</p> <p>The default value is 100.88.0.0/16.</p>
internalJoinSubnet	string	<p>If your existing network infrastructure overlaps with the 100.64.0.0/16 IPv4 subnet, you can specify a different IP address range for internal use by OVN-Kubernetes. You must ensure that the IP address range does not overlap with any other subnet used by your OpenShift Container Platform installation. The IP address range must be larger than the maximum number of nodes that can be added to the cluster. For example, if the clusterNetwork.cidr value is 10.128.0.0/14 and the clusterNetwork.hostPrefix value is /23, then the maximum number of nodes is 2^(23-14)=512.</p> <p>The default value is 100.64.0.0/16.</p>

Table 5.6. **ovnKubernetesConfig.ipv6** object

Field	Type	Description
internalTransitSwitchSubnet	string	<p>If your existing network infrastructure overlaps with the fd97::/64 IPv6 subnet, you can specify a different IP address range for internal use by OVN-Kubernetes. The subnet for the distributed transit switch that enables east-west traffic. This subnet cannot overlap with any other subnets used by OVN-Kubernetes or on the host itself. It must be large enough to accommodate one IP address per node in your cluster.</p> <p>The default value is fd97::/64.</p>
internalJoinSubnet	string	<p>If your existing network infrastructure overlaps with the fd98::/64 IPv6 subnet, you can specify a different IP address range for internal use by OVN-Kubernetes. You must ensure that the IP address range does not overlap with any other subnet used by your OpenShift Container Platform installation. The IP address range must be larger than the maximum number of nodes that can be added to the cluster.</p> <p>The default value is fd98::/64.</p>

Table 5.7. **policyAuditConfig** object

Field	Type	Description
rateLimit	integer	The maximum number of messages to generate every second per node. The default value is 20 messages per second.
maxFileSize	integer	The maximum size for the audit log in bytes. The default value is 50000000 or 50 MB.
maxLogFiles	integer	The maximum number of log files that are retained.
destination	string	<p>One of the following additional audit log targets:</p> <p>libc The libc syslog() function of the journald process on the host.</p> <p>udp:<host>:<port> A syslog server. Replace <host>:<port> with the host and port of the syslog server.</p> <p>unix:<file> A Unix Domain Socket file specified by <file>.</p> <p>null Do not send the audit logs to any additional target.</p>
syslogFacility	string	The syslog facility, such as kern , as defined by RFC5424. The default value is local0 .

Table 5.8. `gatewayConfig` object

Field	Type	Description
routingViaHost	boolean	<p>Set this field to true to send egress traffic from pods to the host networking stack. For highly-specialized installations and applications that rely on manually configured routes in the kernel routing table, you might want to route egress traffic to the host networking stack. By default, egress traffic is processed in OVN to exit the cluster and is not affected by specialized routes in the kernel routing table. The default value is false.</p> <p>This field has an interaction with the Open vSwitch hardware offloading feature. If you set this field to true, you do not receive the performance benefits of the offloading because egress traffic is processed by the host networking stack.</p>

Field	Type	Description
ipForwarding	object	<p>You can control IP forwarding for all traffic on OVN-Kubernetes managed interfaces by using the ipForwarding specification in the Network resource. Specify Restricted to only allow IP forwarding for Kubernetes related traffic. Specify Global to allow forwarding of all IP traffic. For new installations, the default is Restricted. For updates to OpenShift Container Platform 4.14 or later, the default is Global.</p>  <p>NOTE</p> <p>The default value of Restricted sets the IP forwarding to drop.</p>
ipv4	object	<p>Optional: Specify an object to configure the internal OVN-Kubernetes masquerade address for host to service traffic for IPv4 addresses.</p>
ipv6	object	<p>Optional: Specify an object to configure the internal OVN-Kubernetes masquerade address for host to service traffic for IPv6 addresses.</p>

Table 5.9. `gatewayConfig.ipv4` object

Field	Type	Description
internalMasqueradeSubnet	string	<p>The masquerade IPv4 addresses that are used internally to enable host to service traffic. The host is configured with these IP addresses as well as the shared gateway bridge interface. The default value is 169.254.169.0/29.</p>  <p>IMPORTANT</p> <p>For OpenShift Container Platform 4.17 and later versions, clusters use 169.254.0.0/17 as the default masquerade subnet. For upgraded clusters, there is no change to the default masquerade subnet.</p>

Table 5.10. `gatewayConfig.ipv6` object

Field	Type	Description
-------	------	-------------

Field	Type	Description
internalMasqueradeSubnet	string	<p>The masquerade IPv6 addresses that are used internally to enable host to service traffic. The host is configured with these IP addresses as well as the shared gateway bridge interface. The default value is fd69::/125.</p>  <p>IMPORTANT</p> <p>For OpenShift Container Platform 4.17 and later versions, clusters use fd69::/112 as the default masquerade subnet. For upgraded clusters, there is no change to the default masquerade subnet.</p>

Table 5.11. ipsecConfig object

Field	Type	Description
mode	string	<p>Specifies the behavior of the IPsec implementation. Must be one of the following values:</p> <ul style="list-style-type: none"> • Disabled: IPsec is not enabled on cluster nodes. • External: IPsec is enabled for network traffic with external hosts. • Full: IPsec is enabled for pod traffic and network traffic with external hosts.

Example OVN-Kubernetes configuration with IPSec enabled

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
    ipsecConfig:
      mode: Full
```

5.11. DEPLOYING THE CLUSTER

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- You have verified that the cloud provider account on your host has the correct permissions to deploy the cluster. An account with incorrect permissions causes the installation process to fail with an error message that displays the missing permissions.

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

2. Change to the directory that contains the installation program and initialize the cluster deployment:

```
$ ./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**.

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.

- If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
- If you included the **Service Account Key Admin** role, you can remove it.

Verification

When the cluster deployment completes successfully:

- The terminal displays directions for accessing your cluster, including a link to the web console and credentials for the **kubeadmin** user.
- Credential information also outputs to **<installation_directory>/openshift_install.log**.



IMPORTANT

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

Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

5.12. LOGGING IN TO THE CLUSTER BY USING THE CLI

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

- You deployed an OpenShift Container Platform cluster.
- You installed 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
```

Example output

```
system:admin
```

Additional resources

- See [Accessing the web console](#) for more details about accessing and understanding the OpenShift Container Platform web console.

5.13. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.18, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to [OpenShift Cluster Manager](#).

After you confirm that your [OpenShift Cluster Manager](#) inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See [About remote health monitoring](#) for more information about the Telemetry service

5.14. NEXT STEPS

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

CHAPTER 6. INSTALLING A CLUSTER ON GCP IN A DISCONNECTED ENVIRONMENT

In OpenShift Container Platform 4.18, you can install a cluster on Google Cloud Platform (GCP) in a restricted network by creating an internal mirror of the installation release content on an existing Google Virtual Private Cloud (VPC).



IMPORTANT

You can install an OpenShift Container Platform cluster by using mirrored installation release content, but your cluster will require internet access to use the GCP APIs.

6.1. PREREQUISITES

- You reviewed details about the [OpenShift Container Platform installation and update processes](#).
- You read the documentation on [selecting a cluster installation method and preparing it for users](#).
- You [configured a GCP project](#) to host the cluster.
- You [mirrored the images for a disconnected installation](#) to your registry and obtained the **imageContentSources** data for your version of OpenShift Container Platform.



IMPORTANT

Because the installation media is on the mirror host, you can use that computer to complete all installation steps.

- You have an existing VPC in GCP. While installing a cluster in a restricted network that uses installer-provisioned infrastructure, you cannot use the installer-provisioned VPC. You must use a user-provisioned VPC that satisfies one of the following requirements:
 - Contains the mirror registry
 - Has firewall rules or a peering connection to access the mirror registry hosted elsewhere
- If you use a firewall, you [configured it to allow the sites](#) that your cluster requires access to. While you might need to grant access to more sites, you must grant access to ***.googleapis.com** and **accounts.google.com**.

6.2. ABOUT INSTALLATIONS IN RESTRICTED NETWORKS

In OpenShift Container Platform 4.18, you can perform an installation that does not require an active connection to the internet to obtain software components. Restricted network installations can be completed using installer-provisioned infrastructure or user-provisioned infrastructure, depending on the cloud platform to which you are installing the cluster.

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 Route 53 DNS and IAM services, require internet access. Depending on your network, you might require less internet access for an installation on bare metal hardware, Nutanix, or on VMware vSphere.

To complete a restricted network installation, you must create a registry that mirrors the contents of the OpenShift image registry and contains the installation media. You can create this registry on a mirror host, which can access both the internet and your closed network, or by using other methods that meet your restrictions.

6.2.1. 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 image stream tags.

6.3. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.18, you require access to the internet to obtain the images that are necessary to install your cluster.

You must have internet access to:

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

6.4. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS

During an OpenShift Container Platform installation, you can provide an SSH public key to the installation program. The key is passed to the Red Hat Enterprise Linux CoreOS (RHCOS) nodes through their Ignition config files and is used to authenticate SSH access to the nodes. The key is added to the `~/.ssh/authorized_keys` list for the **core** user on each node, which enables password-less authentication.

After the key is passed to the nodes, you can use the key pair to SSH in to the RHCOS nodes as the user **core**. To access the nodes through SSH, the private key identity must be managed by SSH for your local user.

If you want to SSH in to your cluster nodes to perform installation debugging or disaster recovery, you must provide the SSH public key during the installation process. The `./openshift-install gather` command also requires the SSH public key to be in place on the cluster nodes.



IMPORTANT

Do not skip this procedure in production environments, where disaster recovery and debugging is required.



NOTE

You must use a local key, not one that you configured with platform-specific approaches.

Procedure

1. If you do not have an existing SSH key pair on your local machine to use for authentication onto your cluster nodes, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" -f <path>/<file_name> ①
```

- 1 Specify the path and file name, such as `~/.ssh/id_ed25519`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the **x86_64**, **ppc64le**, and **s390x** architectures, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. View the public SSH key:

```
$ cat <path>/<file_name>.pub
```

For example, run the following to view the `~/.ssh/id_ed25519.pub` public key:

```
$ cat ~/.ssh/id_ed25519.pub
```

3. Add the SSH private key identity to the SSH agent for your local user, if it has not already been added. SSH agent management of the key is required for password-less SSH authentication onto your cluster nodes, or if you want to use the `./openshift-install gather` command.



NOTE

On some distributions, default SSH private key identities such as `~/.ssh/id_rsa` and `~/.ssh/id_dsa` are managed automatically.

- a. If the **ssh-agent** process is not already running for your local user, start it as a background task:

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

Example output

```
Agent pid 31874
```



NOTE

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

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

\$ ssh-add <path>/<file_name> ①

- Specify the path and file name for your SSH private key, such as `~/.ssh/id_ed25519`

Example output

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

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

6.5. CREATING THE INSTALLATION CONFIGURATION FILE

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP).

Prerequisites

- You have the OpenShift Container Platform installation program and the pull secret for your cluster. For a restricted network installation, these files are on your mirror host.
- You have the **imageContentSources** values that were generated during mirror registry creation.
- You have obtained the contents of the certificate for your mirror registry.
- Configure a GCP account.

Procedure

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

- Change to the directory that contains the installation program and 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.

When specifying the directory:

- Verify that the directory has the **execute** permission. This permission is required to run Terraform binaries under the installation directory.
- Use an empty directory. Some installation assets, such as bootstrap X.509 certificates, have short expiration intervals, therefore 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, 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.

2. Edit the **install-config.yaml** file to give 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": {"<mirror_host_name>:5000": {"auth": "<credentials>","email": "you@example.com"}}}
```

For **<mirror_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.

```
additionalTrustBundle: |
-----BEGIN CERTIFICATE-----
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
-----END CERTIFICATE-----
```

The value must be the contents of the certificate file that you used for your mirror registry. The certificate file can be an existing, trusted certificate authority, or the self-signed certificate that you generated for the mirror registry.

- c. Define the network and subnets for the VPC to install the cluster in under the parent **platform.gcp** field:

```
network: <existing_vpc>
controlPlaneSubnet: <control_plane_subnet>
computeSubnet: <compute_subnet>
```

For **platform.gcp.network**, specify the name for the existing Google VPC. For **platform.gcp.controlPlaneSubnet** and **platform.gcp.computeSubnet**, specify the existing subnets to deploy the control plane machines and compute machines, respectively.

- d. Add the image content resources, which resemble the following YAML excerpt:

```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
    source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
    source: registry.redhat.io/ocp/release
```

For these values, use the **imageContentSources** that you recorded during mirror registry creation.

- e. Optional: Set the publishing strategy to **Internal**:

```
publish: Internal
```

By setting this option, you create an internal Ingress Controller and a private load balancer.

3. Make any other modifications to the **install-config.yaml** file that you require. For more information about the parameters, see "Installation configuration parameters".
4. 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.

Additional resources

- [Installation configuration parameters for GCP](#)

6.5.1. Minimum resource requirements for cluster installation

Each cluster machine must meet the following minimum requirements:

Table 6.1. Minimum resource requirements

Machine	Operating System	vCPU [1]	Virtual RAM	Storage	Input/Output Per Second (IOPS)[2]
Bootstrap	RHCOS	4	16 GB	100 GB	300

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

1. One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.
2. OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.
3. As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.



NOTE

For OpenShift Container Platform version 4.18, RHCOS is based on RHEL version 9.4, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see [Architectures](#) (RHEL documentation).

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

Additional resources

- [Optimizing storage](#)

6.5.2. Tested instance types for GCP

The following Google Cloud Platform instance types have been tested with OpenShift Container Platform.



NOTE

Not all instance types are available in all regions and zones. For a detailed breakdown of which instance types are available in which zones, see [regions and zones](#) (Google documentation).

Some instance types require the use of Hyperdisk storage. If you use an instance type that requires Hyperdisk storage, all of the nodes in your cluster must support Hyperdisk storage, and you must change the default storage class to use Hyperdisk storage. For more information, see [machine series support for Hyperdisk](#) (Google documentation). For instructions on modifying storage classes, see the "GCE PersistentDisk (gcePD) object definition" section in the Dynamic Provisioning page in *Storage*.

Example 6.1. Machine series

- **A2**
- **A3**
- **C2**
- **C2D**
- **C3**
- **C3D**
- **C4**
- **E2**
- **M1**
- **N1**
- **N2**
- **N2D**
- **N4**
- **Tau T2D**

6.5.3. Tested instance types for GCP on 64-bit ARM infrastructures

The following Google Cloud Platform (GCP) 64-bit ARM instance types have been tested with OpenShift Container Platform.

Example 6.2. Machine series for 64-bit ARM machines

- **C4A**

- Tau T2A

6.5.4. Using custom machine types

Using a custom machine type to install a OpenShift Container Platform cluster is supported.

Consider the following when using a custom machine type:

- Similar to predefined instance types, custom machine types must meet the minimum resource requirements for control plane and compute machines. For more information, see "Minimum resource requirements for cluster installation".
- The name of the custom machine type must adhere to the following syntax:
custom-<number_of_cpus>-<amount_of_memory_in_mb>

For example, **custom-6-20480**.

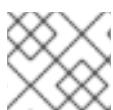
As part of the installation process, you specify the custom machine type in the **install-config.yaml** file.

Sample **install-config.yaml** file with a custom machine type

```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    gcp:
      type: custom-6-20480
  replicas: 3
```

6.5.5. Enabling Shielded VMs

You can use Shielded VMs when installing your cluster. Shielded VMs have extra security features including secure boot, firmware and integrity monitoring, and rootkit detection. For more information, see Google's documentation on [Shielded VMs](#).



NOTE

Shielded VMs are currently not supported on clusters with 64-bit ARM infrastructures.

Procedure

- Use a text editor to edit the **install-config.yaml** file prior to deploying your cluster and add one of the following stanzas:

- a. To use shielded VMs for only control plane machines:

```
controlPlane:
platform:
gcp:
  secureBoot: Enabled
```

- b. To use shielded VMs for only compute machines:

```
compute:
- platform:
  gcp:
    secureBoot: Enabled
```

- c. To use shielded VMs for all machines:

```
platform:
gcp:
  defaultMachinePlatform:
    secureBoot: Enabled
```

6.5.6. Enabling Confidential VMs

You can use Confidential VMs when installing your cluster. Confidential VMs encrypt data while it is being processed. For more information, see Google's documentation on [Confidential Computing](#). You can enable Confidential VMs and Shielded VMs at the same time, although they are not dependent on each other.



NOTE

Confidential VMs are currently not supported on 64-bit ARM architectures.

Procedure

- Use a text editor to edit the **install-config.yaml** file prior to deploying your cluster and add one of the following stanzas:

- a. To use confidential VMs for only control plane machines:

```
controlPlane:
platform:
gcp:
  confidentialCompute: Enabled ①
  type: n2d-standard-8 ②
  onHostMaintenance: Terminate ③
```

① Enable confidential VMs.

② Specify a machine type that supports Confidential VMs. Confidential VMs require the N2D or C2D series of machine types. For more information on supported machine types, see [Supported operating systems and machine types](#).

③

Specify the behavior of the VM during a host maintenance event, such as a hardware or software update. For a machine that uses Confidential VM, this value must be set to

- b. To use confidential VMs for only compute machines:

```
compute:
- platform:
  gcp:
    confidentialCompute: Enabled
    type: n2d-standard-8
    onHostMaintenance: Terminate
```

- c. To use confidential VMs for all machines:

```
platform:
gcp:
  defaultMachinePlatform:
    confidentialCompute: Enabled
    type: n2d-standard-8
    onHostMaintenance: Terminate
```

6.5.7. 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
credentialsMode: Mint 2
controlPlane: 3 4
hyperthreading: Enabled 5
name: master
platform:
  gcp:
    type: n2-standard-4
    zones:
      - us-central1-a
      - us-central1-c
    osDisk:
      diskType: pd-ssd
      diskSizeGB: 1024
      encryptionKey: 6
      kmsKey:
        name: worker-key
        keyRing: test-machine-keys
        location: global
        projectId: project-id
```

```

tags: 7
- control-plane-tag1
- control-plane-tag2
osImage: 8
  project: example-project-name
  name: example-image-name
replicas: 3
compute: 9 10
- hyperthreading: Enabled 11
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
    osDisk:
      diskType: pd-standard
      diskSizeGB: 128
      encryptionKey: 12
        kmsKey:
          name: worker-key
          keyRing: test-machine-keys
          location: global
          projectId: project-id
    tags: 13
      - compute-tag1
      - compute-tag2
  osImage: 14
    project: example-project-name
    name: example-image-name
replicas: 3
metadata:
  name: test-cluster 15
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  machineNetwork:
    - cidr: 10.0.0.0/16
  networkType: OVNKubernetes 16
  serviceNetwork:
    - 172.30.0.0/16
platform:
  gcp:
    projectId: openshift-production 17
    region: us-central1 18
    defaultMachinePlatform:
      tags: 19
        - global-tag1
        - global-tag2
      osImage: 20
        project: example-project-name
        name: example-image-name

```

```

network: existing_vpc 21
controlPlaneSubnet: control_plane_subnet 22
computeSubnet: compute_subnet 23
pullSecret: '{"auths": {"<local_registry>": {"auth": "<credentials>", "email": "you@example.com"}},}' 24
fips: false 25
sshKey: ssh-ed25519 AAAA... 26
additionalTrustBundle: | 27
-----BEGIN CERTIFICATE-----
<MY_TRUSTED_CA_CERT>
-----END CERTIFICATE-----
imageContentSources: 28
- mirrors:
  - <local_registry>/<local_repository_name>/release
    source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
    source: quay.io/openshift-release-dev/ocp-v4.0-art-dev

```

1 **15** **17** **18** Required. The installation program prompts you for this value.

2 Optional: Add this parameter to force the Cloud Credential Operator (CCO) to use the specified mode. By default, the CCO uses the root credentials in the **kube-system** namespace to dynamically try to determine the capabilities of the credentials. For details about CCO modes, see the "About the Cloud Credential Operator" section in the *Authentication and authorization* guide.

3 **9** If you do not provide these parameters and values, the installation program provides the default value.

4 **10** 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. Only one control plane pool is used.

5 **11** 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.

6 **12** Optional: The custom encryption key section to encrypt both virtual machines and persistent volumes. Your default compute service account must have the permissions granted to use your KMS key and have the correct IAM role assigned. The default service account name follows the **service-<project_number>@compute-system.iam.gserviceaccount.com** pattern. For more information about granting the correct permissions for your service account, see "Machine management" → "Creating compute machine sets" → "Creating a compute machine set on GCP".

7 **13** **19**

Optional: A set of network tags to apply to the control plane or compute machine sets. The **platform.gcp.defaultMachinePlatform.tags** parameter will apply to both control plane and

8 14 20 Optional: A custom Red Hat Enterprise Linux CoreOS (RHCOS) that should be used to boot control plane and compute machines. The **project** and **name** parameters under **platform.gcp.defaultMachinePlatform.osImage** apply to both control plane and compute machines. If the **project** and **name** parameters under **controlPlane.platform.gcp.osImage** or **compute.platform.gcp.osImage** are set, they override the **platform.gcp.defaultMachinePlatform.osImage** parameters.

- 16** The cluster network plugin to install. The default value **OVNKubernetes** is the only supported value.
- 21** Specify the name of an existing VPC.
- 22** Specify the name of the existing subnet to deploy the control plane machines to. The subnet must belong to the VPC that you specified.
- 23** Specify the name of the existing subnet to deploy the compute machines to. The subnet must belong to the VPC that you specified.
- 24** For **<local_registry>**, specify the registry domain name, and optionally the port, that your mirror registry uses to serve content. For example, **registry.example.com** or **registry.example.com:5000**. For **<credentials>**, specify the base64-encoded user name and password for your mirror registry.
- 25** Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.



IMPORTANT

When running Red Hat Enterprise Linux (RHEL) or Red Hat Enterprise Linux CoreOS (RHCOS) booted in FIPS mode, OpenShift Container Platform core components use the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the x86_64, ppc64le, and s390x architectures.

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

- 27** Provide the contents of the certificate file that you used for your mirror registry.
- 28** Provide the **imageContentSources** section from the output of the command to mirror the repository.

6.5.8. Create an Ingress Controller with global access on GCP

You can create an Ingress Controller that has global access to a Google Cloud Platform (GCP) cluster. Global access is only available to Ingress Controllers using internal load balancers.

Prerequisites

- You created the **install-config.yaml** and complete any modifications to it.

Procedure

Create an Ingress Controller with global access on a new GCP cluster.

1. Change to the directory that contains the installation program and create a manifest file:

```
$ ./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-ingress-default-ingresscontroller.yaml** in the **<installation_directory>/manifests** directory:

```
$ touch <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml 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-ingress-default-ingresscontroller.yaml
```

Example output

```
cluster-ingress-default-ingresscontroller.yaml
```

3. Open the **cluster-ingress-default-ingresscontroller.yaml** file in an editor and enter a custom resource (CR) that describes the Operator configuration you want:

Sample clientAccess configuration to Global

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      providerParameters:
        gcp:
          clientAccess: Global ①
```

```

type: GCP
scope: Internal 2
type: LoadBalancerService

```

- 1** Set **gcp.clientAccess** to **Global**.
- 2** Global access is only available to Ingress Controllers using internal load balancers.

6.5.9. 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

- You have an existing **install-config.yaml** file.
- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

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: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
  additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5

```

- 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.
- 3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with `.` to match subdomains only. For example, `.y.com` matches `x.y.com`, but not `y.com`. Use `*` to bypass the proxy for all destinations.
- 4 If provided, the installation program generates a config map 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** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **trustedCA** field of the **Proxy** object. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.
- 5 Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.



NOTE

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



NOTE

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

- 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.

6.6. INSTALLING THE OPENSHIFT CLI

You can install the OpenShift CLI (**oc**) to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.18. Download and install the new version of **oc**.

Installing the OpenShift CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** drop-down list.
3. Select the appropriate version from the **Version** drop-down list.
4. Click **Download Now** next to the **OpenShift v4.18 Linux Clients** entry and save the file.
5. Unpack the archive:

```
$ tar xvf <file>
```

6. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

Verification

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

```
$ oc <command>
```

Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.18 Windows Client** entry and save the file.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

Verification

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

```
C:\> oc <command>
```

Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.18 macOS Clients** entry and save the file.



NOTE

For macOS arm64, choose the **OpenShift v4.18 macOS arm64 Client** entry.

4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.

To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

Verification

- Verify your installation by using an **oc** command:

```
$ oc <command>
```

6.7. ALTERNATIVES TO STORING ADMINISTRATOR-LEVEL SECRETS IN THE KUBE-SYSTEM PROJECT

By default, administrator secrets are stored in the **kube-system** project. If you configured the **credentialsMode** parameter in the **install-config.yaml** file to **Manual**, you must use one of the following alternatives:

- To manage long-term cloud credentials manually, follow the procedure in [Manually creating long-term credentials](#).
- To implement short-term credentials that are managed outside the cluster for individual components, follow the procedures in [Configuring a GCP cluster to use short-term credentials](#).

6.7.1. Manually creating long-term credentials

The Cloud Credential Operator (CCO) can be put into manual mode prior to installation in environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace.

Procedure

1. Add the following granular permissions to the GCP account that the installation program uses:

Example 6.3. Required GCP permissions

- compute.machineTypes.list
- compute.regions.list
- compute.zones.list
- dns.changes.create
- dns.changes.get
- dns.managedZones.create
- dns.managedZones.delete
- dns.managedZones.get
- dns.managedZones.list
- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

2. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

Sample configuration file snippet

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

3. If you have not previously created installation manifest files, do so by running the following command:

```
$ openshift-install create manifests --dir <installation_directory>
```

where **<installation_directory>** is the directory in which the installation program creates files.

4. Set a **\$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

5. Extract the list of **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
```

```
--from=$RELEASE_IMAGE \
--credentials-requests \
--included \①
--install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \②
--to=<path_to_directory_for_credentials_requests> ③
```

- ① The **--included** parameter includes only the manifests that your specific cluster configuration requires.
- ② Specify the location of the **install-config.yaml** file.
- ③ Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

This command creates a YAML file for each **CredentialsRequest** object.

Sample CredentialsRequest object

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: GCPProviderSpec
    predefinedRoles:
      - roles/storage.admin
      - roles/iam.serviceAccountUser
    skipServiceCheck: true
...
```

6. Create YAML files for secrets in the **openshift-install** manifests directory that you generated previously. The secrets must be stored using the namespace and secret name defined in the **spec.secretRef** for each **CredentialsRequest** object.

Sample CredentialsRequest object with secrets

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
...
```

Sample Secret object

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  service_account.json: <base64_encoded_gcp_service_account_file>
```



IMPORTANT

Before upgrading a cluster that uses manually maintained credentials, you must ensure that the CCO is in an upgradeable state.

6.7.2. Configuring a GCP cluster to use short-term credentials

To install a cluster that is configured to use GCP Workload Identity, you must configure the CCO utility and create the required GCP resources for your cluster.

6.7.2.1. Configuring the Cloud Credential Operator utility

To create and manage cloud credentials from outside of the cluster when the Cloud Credential Operator (CCO) is operating in manual mode, extract and prepare the CCO utility (**ccocctl**) binary.



NOTE

The **ccocctl** utility is a Linux binary that must run in a Linux environment.

Prerequisites

- You have access to an OpenShift Container Platform account with cluster administrator access.
- You have installed the OpenShift CLI (**oc**).
- You have added one of the following authentication options to the GCP account that the **ccocctl** utility uses:
 - The **IAM Workload Identity Pool Adminrole**
 - The following granular permissions:
 - **compute.projects.get**
 - **iam.googleapis.com/workloadIdentityPoolProviders.create**
 - **iam.googleapis.com/workloadIdentityPoolProviders.get**
 - **iam.googleapis.com/workloadIdentityPools.create**
 - **iam.googleapis.com/workloadIdentityPools.delete**
 - **iam.googleapis.com/workloadIdentityPools.get**
 - **iam.googleapis.com/workloadIdentityPools.undelete**

- **iam.roles.create**
- **iam.roles.delete**
- **iam.roles.list**
- **iam.roles.undelete**
- **iam.roles.update**
- **iam.serviceAccounts.create**
- **iam.serviceAccounts.delete**
- **iam.serviceAccounts.getIamPolicy**
- **iam.serviceAccounts.list**
- **iam.serviceAccounts.setIamPolicy**
- **iam.workloadIdentityPoolProviders.get**
- **iam.workloadIdentityPools.delete**
- **resourcemanager.projects.get**
- **resourcemanager.projects.getIamPolicy**
- **resourcemanager.projects.setIamPolicy**
- **storage.buckets.create**
- **storage.buckets.delete**
- **storage.buckets.get**
- **storage.buckets.getIamPolicy**
- **storage.buckets.setIamPolicy**
- **storage.objects.create**
- **storage.objects.delete**
- **storage.objects.list**

Procedure

1. Set a variable for the OpenShift Container Platform release image by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Obtain the CCO container image from the OpenShift Container Platform release image by running the following command:

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/pull-secret)
```



NOTE

Ensure that the architecture of the **\$RELEASE_IMAGE** matches the architecture of the environment in which you will use the **ccctl** tool.

- Extract the **ccctl** binary from the CCO container image within the OpenShift Container Platform release image by running the following command:

```
$ oc image extract $CCO_IMAGE \
--file="/usr/bin/ccctl.<rhel_version>" \①
-a ~/pull-secret
```

- For **<rhel_version>**, specify the value that corresponds to the version of Red Hat Enterprise Linux (RHEL) that the host uses. If no value is specified, **ccctl.rhel8** is used by default. The following values are valid:
 - rhel8**: Specify this value for hosts that use RHEL 8.
 - rhel9**: Specify this value for hosts that use RHEL 9.

- Change the permissions to make **ccctl** executable by running the following command:

```
$ chmod 775 ccctl.<rhel_version>
```

Verification

- To verify that **ccctl** is ready to use, display the help file. Use a relative file name when you run the command, for example:

```
$ ./ccctl.rhel9
```

Example output

```
OpenShift credentials provisioning tool
```

Usage:

```
ccctl [command]
```

Available Commands:

aws	Manage credentials objects for AWS cloud
azure	Manage credentials objects for Azure
gcp	Manage credentials objects for Google cloud
help	Help about any command
ibmcloud	Manage credentials objects for {ibm-cloud-title}
nutanix	Manage credentials objects for Nutanix

Flags:

`-h, --help` help for ccoctl

Use "ccoctl [command] --help" for more information about a command.

6.7.2.2. Creating GCP resources with the Cloud Credential Operator utility

You can use the **ccoctl gcp create-all** command to automate the creation of GCP resources.



NOTE

By default, **ccoctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccoctl_output_dir>** to refer to this directory.

Prerequisites

You must have:

- Extracted and prepared the **ccoctl** binary.

Procedure

- Set a **\$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

- Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
--from=$RELEASE_IMAGE \
--credentials-requests \
--included ① \
--install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
--to=<path_to_directory_for_credentials_requests> ② ③
```

- The **--included** parameter includes only the manifests that your specific cluster configuration requires.
- Specify the location of the **install-config.yaml** file.
- Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.



NOTE

This command might take a few moments to run.

- Use the **ccoctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccoctl gcp create-all \
--name=<name> \①
--region=<gcp_region> \②
--project=<gcp_project_id> \③
--credentials-requests-dir=<path_to_credentials_requests_directory> \④
```

- ① Specify the user-defined name for all created GCP resources used for tracking. If you plan to install the GCP Filestore Container Storage Interface (CSI) Driver Operator, retain this value.
- ② Specify the GCP region in which cloud resources will be created.
- ③ Specify the GCP project ID in which cloud resources will be created.
- ④ Specify the directory containing the files of **CredentialsRequest** manifests to create GCP service accounts.



NOTE

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

Example output

```
cluster-authentication-02-config.yaml
openshift-cloud-controller-manager-gcp-ccm-cloud-credentials-credentials.yaml
openshift-cloud-credential-operator-cloud-credential-operator-gcp-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capg-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-gcp-pd-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-gcp-cloud-credentials-credentials.yaml
```

You can verify that the IAM service accounts are created by querying GCP. For more information, refer to GCP documentation on listing IAM service accounts.

6.7.2.3. Incorporating the Cloud Credential Operator utility manifests

To implement short-term security credentials managed outside the cluster for individual components, you must move the manifest files that the Cloud Credential Operator utility (**ccoctl**) created to the correct directories for the installation program.

Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have configured the Cloud Credential Operator utility (**ccocctl**).
- You have created the cloud provider resources that are required for your cluster with the **ccocctl** utility.

Procedure

1. Add the following granular permissions to the GCP account that the installation program uses:

Example 6.4. Required GCP permissions

- compute.machineTypes.list
- compute.regions.list
- compute.zones.list
- dns.changes.create
- dns.changes.get
- dns.managedZones.create
- dns.managedZones.delete
- dns.managedZones.get
- dns.managedZones.list
- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

2. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

Sample configuration file snippet

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

3. If you have not previously created installation manifest files, do so by running the following command:

```
$ openshift-install create manifests --dir <installation_directory>
```

where **<installation_directory>** is the directory in which the installation program creates files.

4. Copy the manifests that the **ccctl** utility generated to the **manifests** directory that the installation program created by running the following command:

```
$ cp /<path_to_ccctl_output_dir>/manifests/* ./manifests/
```

5. Copy the **tls** directory that contains the private key to the installation directory:

```
$ cp -a /<path_to_ccctl_output_dir>/tls .
```

6.8. DEPLOYING THE CLUSTER

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- You have verified that the cloud provider account on your host has the correct permissions to deploy the cluster. An account with incorrect permissions causes the installation process to fail with an error message that displays the missing permissions.

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

2. Change to the directory that contains the installation program and initialize the cluster deployment:

```
$ ./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**.

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.
 - If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
 - If you included the **Service Account Key Admin** role, you can remove it.

Verification

When the cluster deployment completes successfully:

- The terminal displays directions for accessing your cluster, including a link to the web console and credentials for the **kubeadmin** user.
- Credential information also outputs to `<installation_directory>/openshift_install.log`.



IMPORTANT

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

Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

6.9. LOGGING IN TO THE CLUSTER BY USING THE CLI

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

- You deployed an OpenShift Container Platform cluster.
- You installed 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
```

Example output

```
system:admin
```

6.10. DISABLING THE DEFAULT OPERATORHUB CATALOG SOURCES

Operator catalogs that source content provided by Red Hat and community projects are configured for OperatorHub by default during an OpenShift Container Platform installation. In a restricted network environment, you must disable the default catalogs as a cluster administrator.

Procedure

- Disable the sources for the default catalogs by adding **disableAllDefaultSources: true** to the **OperatorHub** object:

```
$ oc patch OperatorHub cluster --type json \
-p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

TIP

Alternatively, you can use the web console to manage catalog sources. From the **Administration → Cluster Settings → Configuration → OperatorHub** page, click the **Sources** tab, where you can create, update, delete, disable, and enable individual sources.

6.11. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.18, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to [OpenShift Cluster Manager](#).

After you confirm that your [OpenShift Cluster Manager](#) inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See [About remote health monitoring](#) for more information about the Telemetry service

6.12. NEXT STEPS

- [Validate an installation.](#)
- [Customize your cluster.](#)
- [Configure image streams for the Cluster Samples Operator and the **must-gather** tool.](#)
- Learn how to [use Operator Lifecycle Manager in disconnected environments](#).
- If the mirror registry that you used to install your cluster has a trusted CA, add it to the cluster by [configuring additional trust stores](#).
- If necessary, you can [opt out of remote health reporting](#).
- If necessary, see [Registering your disconnected cluster](#)

CHAPTER 7. INSTALLING A CLUSTER ON GCP INTO AN EXISTING VPC

In OpenShift Container Platform version 4.18, you can install a cluster into an existing Virtual Private Cloud (VPC) on Google Cloud Platform (GCP). The installation program provisions the rest of the required infrastructure, which you can further customize. To customize the installation, you modify parameters in the `install-config.yaml` file before you install the cluster.

7.1. PREREQUISITES

- You reviewed details about the [OpenShift Container Platform installation and update processes](#).
- You read the documentation on [selecting a cluster installation method and preparing it for users](#).
- You [configured a GCP project](#) to host the cluster.
- If you use a firewall, you [configured it to allow the sites](#) that your cluster requires access to.

7.2. ABOUT USING A CUSTOM VPC

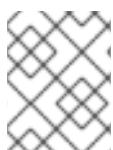
In OpenShift Container Platform 4.18, you can deploy a cluster into existing subnets in an existing Virtual Private Cloud (VPC) in Google Cloud Platform (GCP). By deploying OpenShift Container Platform into an existing GCP VPC, you might be able to avoid limit constraints in new accounts or more easily abide by the operational constraints that your company's guidelines set. If you cannot obtain the infrastructure creation permissions that are required to create the VPC yourself, use this installation option. You must configure networking for the subnets.

7.2.1. Requirements for using your VPC

The union of the VPC CIDR block and the machine network CIDR must be non-empty. The subnets must be within the machine network.

The installation program does not create the following components:

- NAT gateways
- Subnets
- Route tables
- VPC network



NOTE

The installation program requires that you use the cloud-provided DNS server. Using a custom DNS server is not supported and causes the installation to fail.

7.2.2. VPC validation

To ensure that the subnets that you provide are suitable, the installation program confirms the following data:

- All the subnets that you specify exist.
- You provide one subnet for control-plane machines and one subnet for compute machines.
- The subnet's CIDRs belong to the machine CIDR that you specified.

7.2.3. Division of permissions

Some individuals can create different resources in your clouds than others. For example, you might be able to create application-specific items, like instances, buckets, and load balancers, but not networking-related components such as VPCs, subnets, or ingress rules.

7.2.4. Isolation between clusters

If you deploy OpenShift Container Platform to an existing network, the isolation of cluster services is reduced in the following ways:

- You can install multiple OpenShift Container Platform clusters in the same VPC.
- ICMP ingress is allowed to the entire network.
- TCP 22 ingress (SSH) is allowed to the entire network.
- Control plane TCP 6443 ingress (Kubernetes API) is allowed to the entire network.
- Control plane TCP 22623 ingress (MCS) is allowed to the entire network.

7.3. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM

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

You must have internet access to:

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



IMPORTANT

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

7.4. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS

During an OpenShift Container Platform installation, you can provide an SSH public key to the installation program. The key is passed to the Red Hat Enterprise Linux CoreOS (RHCOS) nodes through their Ignition config files and is used to authenticate SSH access to the nodes. The key is added

to the `~/.ssh/authorized_keys` list for the **core** user on each node, which enables password-less authentication.

After the key is passed to the nodes, you can use the key pair to SSH in to the RHCOS nodes as the user **core**. To access the nodes through SSH, the private key identity must be managed by SSH for your local user.

If you want to SSH in to your cluster nodes to perform installation debugging or disaster recovery, you must provide the SSH public key during the installation process. The `./openshift-install gather` command also requires the SSH public key to be in place on the cluster nodes.



IMPORTANT

Do not skip this procedure in production environments, where disaster recovery and debugging is required.



NOTE

You must use a local key, not one that you configured with platform-specific approaches.

Procedure

1. If you do not have an existing SSH key pair on your local machine to use for authentication onto your cluster nodes, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" -f <path>/<file_name> ①
```

- 1 Specify the path and file name, such as `~/.ssh/id_ed25519`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the **x86_64**, **ppc64le**, and **s390x** architectures, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. View the public SSH key:

```
$ cat <path>/<file_name>.pub
```

For example, run the following to view the `~/.ssh/id_ed25519.pub` public key:

```
$ cat ~/.ssh/id_ed25519.pub
```

3. Add the SSH private key identity to the SSH agent for your local user, if it has not already been added. SSH agent management of the key is required for password-less SSH authentication onto your cluster nodes, or if you want to use the `./openshift-install gather` command.



NOTE

On some distributions, default SSH private key identities such as `~/.ssh/id_rsa` and `~/.ssh/id_dsa` are managed automatically.

- a. If the **ssh-agent** process is not already running for your local user, start it as a background task:

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

Example output

```
Agent pid 31874
```



NOTE

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

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

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

- ① Specify the path and file name for your SSH private key, such as `~/.ssh/id_ed25519`

Example output

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

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

7.5. OBTAINING THE INSTALLATION PROGRAM

Before you install OpenShift Container Platform, download the installation file on the host you are using for installation.

Prerequisites

- You have a computer that runs Linux or macOS, with 500 MB of local disk space.

Procedure

1. Go to the [Cluster Type](#) page on the Red Hat Hybrid Cloud Console. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

TIP

You can also [download the binaries for a specific OpenShift Container Platform release](#).

2. Select your infrastructure provider from the **Run it yourself** section of the page.
3. Select your host operating system and architecture from the dropdown menus under **OpenShift Installer** and click **Download Installer**.
4. Place the downloaded file in the directory where you want to 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 the installation program and the files that the installation program creates after you finish installing the cluster. Both of the files are required to delete the cluster.
- Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. To remove your cluster, complete the OpenShift Container Platform uninstallation procedures for your specific cloud provider.

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

```
$ tar -xvf openshift-install-linux.tar.gz
```

6. Download your installation [pull secret from Red Hat OpenShift Cluster Manager](#). 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.

TIP

Alternatively, you can retrieve the installation program from the [Red Hat Customer Portal](#), where you can specify a version of the installation program to download. However, you must have an active subscription to access this page.

7.6. CREATING THE INSTALLATION CONFIGURATION FILE

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP).

Prerequisites

- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- Configure a GCP account.

Procedure

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

- Change to the directory that contains the installation program and 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.

When specifying the directory:

- Verify that the directory has the **execute** permission. This permission is required to run Terraform binaries under the installation directory.
- Use an empty directory. Some installation assets, such as bootstrap X.509 certificates, have short expiration intervals, therefore 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, specify an SSH key that your **ssh-agent** process uses.

- Select **gcp** as the platform to target.
- 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.
- Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
- Select the region to deploy the cluster to.
- Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
- Enter a descriptive name for your cluster.

- 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.

Additional resources

- [Installation configuration parameters for GCP](#)

7.6.1. Minimum resource requirements for cluster installation

Each cluster machine must meet the following minimum requirements:

Table 7.1. Minimum resource requirements

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

1. One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.
2. OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.
3. As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.



NOTE

For OpenShift Container Platform version 4.18, RHCOS is based on RHEL version 9.4, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see [Architectures](#) (RHEL documentation).

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

Additional resources

- [Optimizing storage](#)

7.6.2. Tested instance types for GCP

The following Google Cloud Platform instance types have been tested with OpenShift Container Platform.



NOTE

Not all instance types are available in all regions and zones. For a detailed breakdown of which instance types are available in which zones, see [regions and zones](#) (Google documentation).

Some instance types require the use of Hyperdisk storage. If you use an instance type that requires Hyperdisk storage, all of the nodes in your cluster must support Hyperdisk storage, and you must change the default storage class to use Hyperdisk storage. For more information, see [machine series support for Hyperdisk](#) (Google documentation). For instructions on modifying storage classes, see the "GCE PersistentDisk (gcePD) object definition" section in the Dynamic Provisioning page in [Storage](#).

Example 7.1. Machine series

- A2
- A3
- C2
- C2D
- C3
- C3D

- C4
- E2
- M1
- N1
- N2
- N2D
- N4
- Tau T2D

7.6.3. Tested instance types for GCP on 64-bit ARM infrastructures

The following Google Cloud Platform (GCP) 64-bit ARM instance types have been tested with OpenShift Container Platform.

Example 7.2. Machine series for 64-bit ARM machines

- C4A
- Tau T2A

7.6.4. Using custom machine types

Using a custom machine type to install a OpenShift Container Platform cluster is supported.

Consider the following when using a custom machine type:

- Similar to predefined instance types, custom machine types must meet the minimum resource requirements for control plane and compute machines. For more information, see "Minimum resource requirements for cluster installation".
- The name of the custom machine type must adhere to the following syntax:
custom-<number_of_cpus>-<amount_of_memory_in_mb>

For example, **custom-6-20480**.

As part of the installation process, you specify the custom machine type in the **install-config.yaml** file.

Sample **install-config.yaml** file with a custom machine type

```
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
```

```
replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    gcp:
      type: custom-6-20480
  replicas: 3
```

7.6.5. Enabling Shielded VMs

You can use Shielded VMs when installing your cluster. Shielded VMs have extra security features including secure boot, firmware and integrity monitoring, and rootkit detection. For more information, see Google's documentation on [Shielded VMs](#).



NOTE

Shielded VMs are currently not supported on clusters with 64-bit ARM infrastructures.

Procedure

- Use a text editor to edit the **install-config.yaml** file prior to deploying your cluster and add one of the following stanzas:
 - To use shielded VMs for only control plane machines:

```
controlPlane:
  platform:
    gcp:
      secureBoot: Enabled
```

- To use shielded VMs for only compute machines:

```
compute:
- platform:
  gcp:
    secureBoot: Enabled
```

- To use shielded VMs for all machines:

```
platform:
  gcp:
    defaultMachinePlatform:
      secureBoot: Enabled
```

7.6.6. Enabling Confidential VMs

You can use Confidential VMs when installing your cluster. Confidential VMs encrypt data while it is being processed. For more information, see Google's documentation on [Confidential Computing](#). You can enable Confidential VMs and Shielded VMs at the same time, although they are not dependent on each other.

**NOTE**

Confidential VMs are currently not supported on 64-bit ARM architectures.

Procedure

- Use a text editor to edit the **install-config.yaml** file prior to deploying your cluster and add one of the following stanzas:

- To use confidential VMs for only control plane machines:

```
controlPlane:
  platform:
    gcp:
      confidentialCompute: Enabled ①
      type: n2d-standard-8 ②
      onHostMaintenance: Terminate ③
```

- ① Enable confidential VMs.
- ② Specify a machine type that supports Confidential VMs. Confidential VMs require the N2D or C2D series of machine types. For more information on supported machine types, see [Supported operating systems and machine types](#).
- ③ Specify the behavior of the VM during a host maintenance event, such as a hardware or software update. For a machine that uses Confidential VM, this value must be set to **Terminate**, which stops the VM. Confidential VMs do not support live VM migration.

- To use confidential VMs for only compute machines:

```
compute:
  - platform:
    gcp:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

- To use confidential VMs for all machines:

```
platform:
  gcp:
    defaultMachinePlatform:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

7.6.7. 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
credentialsMode: Mint 2
controlPlane: 3 4
hyperthreading: Enabled 5
name: master
platform:
  gcp:
    type: n2-standard-4
    zones:
      - us-central1-a
      - us-central1-c
    osDisk:
      diskType: pd-ssd
      diskSizeGB: 1024
      encryptionKey: 6
      kmsKey:
        name: worker-key
        keyRing: test-machine-keys
        location: global
        projectId: project-id
    tags: 7
      - control-plane-tag1
      - control-plane-tag2
    osImage: 8
      project: example-project-name
      name: example-image-name
replicas: 3
compute: 9 10
- hyperthreading: Enabled 11
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: 12
        kmsKey:
          name: worker-key
          keyRing: test-machine-keys
          location: global
          projectId: project-id
    tags: 13
      - compute-tag1
      - compute-tag2
```

```

osImage: ⑯
  project: example-project-name
  name: example-image-name
replicas: 3
metadata:
  name: test-cluster ⑰
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  machineNetwork:
    - cidr: 10.0.0.0/16
  networkType: OVNKubernetes ⑱
  serviceNetwork:
    - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production ⑲
    region: us-central1 ⑳
    defaultMachinePlatform:
      tags: ㉑
        - global-tag1
        - global-tag2
      osImage: ㉒
        project: example-project-name
        name: example-image-name
    network: existing_vpc ㉓
      controlPlaneSubnet: control_plane_subnet ㉔
      computeSubnet: compute_subnet ㉕
    pullSecret: '{"auths": ...}' ㉖
    fips: false ㉗
    sshKey: ssh-ed25519 AAAA... ㉘

```

⑯ ⑰ ⑲ ㉑ ㉖ Required. The installation program prompts you for this value.

⑰ Optional: Add this parameter to force the Cloud Credential Operator (CCO) to use the specified mode. By default, the CCO uses the root credentials in the **kube-system** namespace to dynamically try to determine the capabilities of the credentials. For details about CCO modes, see the "About the Cloud Credential Operator" section in the *Authentication and authorization* guide.

⑳ ㉗ If you do not provide these parameters and values, the installation program provides the default value.

㉓ ㉔ 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. Only one control plane pool is used.

㉕ ㉘ 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.

- 6 12** Optional: The custom encryption key section to encrypt both virtual machines and persistent volumes. Your default compute service account must have the permissions granted to use your KMS key and have the correct IAM role assigned. The default service account name follows the **service-<project_number>@compute-system.iam.gserviceaccount.com** pattern. For more information about granting the correct permissions for your service account, see "Machine management" → "Creating compute machine sets" → "Creating a compute machine set on GCP".

- 7 13 19** Optional: A set of network tags to apply to the control plane or compute machine sets. The **platform.gcp.defaultMachinePlatform.tags** parameter will apply to both control plane and compute machines. If the **compute.platform.gcp.tags** or **controlPlane.platform.gcp.tags** parameters are set, they override the **platform.gcp.defaultMachinePlatform.tags** parameter.

- 8 14 20** Optional: A custom Red Hat Enterprise Linux CoreOS (RHCOS) that should be used to boot control plane and compute machines. The **project** and **name** parameters under **platform.gcp.defaultMachinePlatform.osImage** apply to both control plane and compute machines. If the **project** and **name** parameters under **controlPlane.platform.gcp.osImage** or **compute.platform.gcp.osImage** are set, they override the **platform.gcp.defaultMachinePlatform.osImage** parameters.

- 16** The cluster network plugin to install. The default value **OVNKubernetes** is the only supported value.
- 21** Specify the name of an existing VPC.
- 22** Specify the name of the existing subnet to deploy the control plane machines to. The subnet must belong to the VPC that you specified.
- 23** Specify the name of the existing subnet to deploy the compute machines to. The subnet must belong to the VPC that you specified.
- 25** Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.



IMPORTANT

To enable FIPS mode for your cluster, you must run the installation program from a Red Hat Enterprise Linux (RHEL) computer configured to operate in FIPS mode. For more information about configuring FIPS mode on RHEL, see [Installing the system in FIPS mode](#).

When running Red Hat Enterprise Linux (RHEL) or Red Hat Enterprise Linux CoreOS (RHCOS) booted in FIPS mode, OpenShift Container Platform core components use the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the x86_64, ppc64le, and s390x architectures.

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

Additional resources

- [Enabling customer-managed encryption keys for a compute machine set](#)

7.6.8. Create an Ingress Controller with global access on GCP

You can create an Ingress Controller that has global access to a Google Cloud Platform (GCP) cluster. Global access is only available to Ingress Controllers using internal load balancers.

Prerequisites

- You created the **install-config.yaml** and complete any modifications to it.

Procedure

Create an Ingress Controller with global access on a new GCP cluster.

1. Change to the directory that contains the installation program and create a manifest file:

```
$ ./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-ingress-default-ingresscontroller.yaml** in the **<installation_directory>/manifests** directory:

```
$ touch <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml 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-ingress-default-ingresscontroller.yaml
```

Example output

```
cluster-ingress-default-ingresscontroller.yaml
```

3. Open the **cluster-ingress-default-ingresscontroller.yaml** file in an editor and enter a custom resource (CR) that describes the Operator configuration you want:

Sample clientAccess configuration to Global

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      providerParameters:
        gcp:
          clientAccess: Global 1
          type: GCP
          scope: Internal 2
      type: LoadBalancerService
```

- 1** Set **gcp.clientAccess** to **Global**.
- 2** Global access is only available to Ingress Controllers using internal load balancers.

7.6.9. 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

- You have an existing **install-config.yaml** file.
- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

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: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5

```

- 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.
- 3** A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with `.` to match subdomains only. For example, `.y.com` matches `x.y.com`, but not `y.com`. Use `*` to bypass the proxy for all destinations.
- 4** If provided, the installation program generates a config map 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** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **trustedCA** field of the **Proxy** object. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.
- 5** Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.



NOTE

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



NOTE

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

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.

7.7. INSTALLING THE OPENSHIFT CLI

You can install the OpenShift CLI (**oc**) to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.18. Download and install the new version of **oc**.

Installing the OpenShift CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** drop-down list.
3. Select the appropriate version from the **Version** drop-down list.
4. Click **Download Now** next to the **OpenShift v4.18 Linux Clients** entry and save the file.
5. Unpack the archive:

```
$ tar xvf <file>
```

6. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

Verification

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

```
$ oc <command>
```

Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.

3. Click **Download Now** next to the **OpenShift v4.18 Windows Client** entry and save the file.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:> path
```

Verification

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

```
C:> oc <command>
```

Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.18 macOS Clients** entry and save the file.



NOTE

For macOS arm64, choose the **OpenShift v4.18 macOS arm64 Client** entry.

4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

Verification

- Verify your installation by using an **oc** command:

```
$ oc <command>
```

7.8. ALTERNATIVES TO STORING ADMINISTRATOR-LEVEL SECRETS IN THE KUBE-SYSTEM PROJECT

By default, administrator secrets are stored in the **kube-system** project. If you configured the **credentialsMode** parameter in the **install-config.yaml** file to **Manual**, you must use one of the following alternatives:

- To manage long-term cloud credentials manually, follow the procedure in [Manually creating long-term credentials](#).
- To implement short-term credentials that are managed outside the cluster for individual components, follow the procedures in [Configuring a GCP cluster to use short-term credentials](#).

7.8.1. Manually creating long-term credentials

The Cloud Credential Operator (CCO) can be put into manual mode prior to installation in environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace.

Procedure

1. Add the following granular permissions to the GCP account that the installation program uses:

Example 7.3. Required GCP permissions

- compute.machineTypes.list
- compute.regions.list
- compute.zones.list
- dns.changes.create
- dns.changes.get
- dns.managedZones.create
- dns.managedZones.delete
- dns.managedZones.get
- dns.managedZones.list
- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

2. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

Sample configuration file snippet

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

3. If you have not previously created installation manifest files, do so by running the following command:

```
$ openshift-install create manifests --dir <installation_directory>
```

where **<installation_directory>** is the directory in which the installation program creates files.

4. Set a **\$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

5. Extract the list of **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
--from=$RELEASE_IMAGE \
--credentials-requests \
--included ① \
--install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \② \
--to=<path_to_directory_for_credentials_requests> ③
```

① The **--included** parameter includes only the manifests that your specific cluster configuration requires.

② Specify the location of the **install-config.yaml** file.

③ Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

This command creates a YAML file for each **CredentialsRequest** object.

Sample CredentialsRequest object

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: GCPPublisherSpec
    predefinedRoles:
      - roles/storage.admin
      - roles/iam.serviceAccountUser
    skipServiceCheck: true
  ...
```

6. Create YAML files for secrets in the **openshift-install** manifests directory that you generated previously. The secrets must be stored using the namespace and secret name defined in the **spec.secretRef** for each **CredentialsRequest** object.

Sample CredentialsRequest object with secrets

```

apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
...

```

Sample Secret object

```

apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  service_account.json: <base64_encoded_gcp_service_account_file>

```



IMPORTANT

Before upgrading a cluster that uses manually maintained credentials, you must ensure that the CCO is in an upgradeable state.

7.8.2. Configuring a GCP cluster to use short-term credentials

To install a cluster that is configured to use GCP Workload Identity, you must configure the CCO utility and create the required GCP resources for your cluster.

7.8.2.1. Configuring the Cloud Credential Operator utility

To create and manage cloud credentials from outside of the cluster when the Cloud Credential Operator (CCO) is operating in manual mode, extract and prepare the CCO utility (**ccctl**) binary.



NOTE

The **ccctl** utility is a Linux binary that must run in a Linux environment.

Prerequisites

- You have access to an OpenShift Container Platform account with cluster administrator access.
- You have installed the OpenShift CLI (**oc**).

- You have added one of the following authentication options to the GCP account that the **ccctl** utility uses:
 - The **IAM Workload Identity Pool Adminrole**
 - The following granular permissions:
 - **compute.projects.get**
 - **iam.googleapis.com/workloadIdentityPoolProviders.create**
 - **iam.googleapis.com/workloadIdentityPoolProviders.get**
 - **iam.googleapis.com/workloadIdentityPools.create**
 - **iam.googleapis.com/workloadIdentityPools.delete**
 - **iam.googleapis.com/workloadIdentityPools.get**
 - **iam.googleapis.com/workloadIdentityPools.undelete**
 - **iam.roles.create**
 - **iam.roles.delete**
 - **iam.roles.list**
 - **iam.roles.undelete**
 - **iam.roles.update**
 - **iam.serviceAccounts.create**
 - **iam.serviceAccounts.delete**
 - **iam.serviceAccounts.getiamPolicy**
 - **iam.serviceAccounts.list**
 - **iam.serviceAccounts.setiamPolicy**
 - **iam.workloadIdentityPoolProviders.get**
 - **iam.workloadIdentityPools.delete**
 - **resourcemanager.projects.get**
 - **resourcemanager.projects.getiamPolicy**
 - **resourcemanager.projects.setiamPolicy**
 - **storage.buckets.create**
 - **storage.buckets.delete**
 - **storage.buckets.get**
 - **storage.buckets.getiamPolicy**

- **storage.buckets.setIamPolicy**
- **storage.objects.create**
- **storage.objects.delete**
- **storage.objects.list**

Procedure

1. Set a variable for the OpenShift Container Platform release image by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Obtain the CCO container image from the OpenShift Container Platform release image by running the following command:

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'  
$RELEASE_IMAGE -a ~/.pull-secret)
```



NOTE

Ensure that the architecture of the **\$RELEASE_IMAGE** matches the architecture of the environment in which you will use the **ccctl** tool.

3. Extract the **ccctl** binary from the CCO container image within the OpenShift Container Platform release image by running the following command:

```
$ oc image extract $CCO_IMAGE \  
--file="/usr/bin/ccctl.<rhel_version>" \①  
-a ~/.pull-secret
```

- ① For **<rhel_version>**, specify the value that corresponds to the version of Red Hat Enterprise Linux (RHEL) that the host uses. If no value is specified, **ccctl.rhel8** is used by default. The following values are valid:

- **rhel8**: Specify this value for hosts that use RHEL 8.
- **rhel9**: Specify this value for hosts that use RHEL 9.

4. Change the permissions to make **ccctl** executable by running the following command:

```
$ chmod 775 ccctl.<rhel_version>
```

Verification

- To verify that **ccctl** is ready to use, display the help file. Use a relative file name when you run the command, for example:

```
$ ./ccctl.rhel9
```

Example output

OpenShift credentials provisioning tool

Usage:

```
ccctl [command]
```

Available Commands:

aws	Manage credentials objects for AWS cloud
azure	Manage credentials objects for Azure
gcp	Manage credentials objects for Google cloud
help	Help about any command
ibmcloud	Manage credentials objects for {ibm-cloud-title}
nutanix	Manage credentials objects for Nutanix

Flags:

```
-h, --help help for ccctl
```

Use "ccctl [command] --help" for more information about a command.

7.8.2.2. Creating GCP resources with the Cloud Credential Operator utility

You can use the **ccctl gcp create-all** command to automate the creation of GCP resources.



NOTE

By default, **ccctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccctl_output_dir>** to refer to this directory.

Prerequisites

You must have:

- Extracted and prepared the **ccctl** binary.

Procedure

- Set a **\$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

- Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
--from=$RELEASE_IMAGE \
--credentials-requests \
--included ① \
--install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
--to=<path_to_directory_for_credentials_requests> ② ③
```

- 1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.
- 2 Specify the location of the **install-config.yaml** file.
- 3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

**NOTE**

This command might take a few moments to run.

3. Use the **ccctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccctl gcp create-all \
--name=<name> \ ①
--region=<gcp_region> \ ②
--project=<gcp_project_id> \ ③
--credentials-requests-dir=<path_to_credentials_requests_directory> ④
```

- 1 Specify the user-defined name for all created GCP resources used for tracking. If you plan to install the GCP Filestore Container Storage Interface (CSI) Driver Operator, retain this value.
- 2 Specify the GCP region in which cloud resources will be created.
- 3 Specify the GCP project ID in which cloud resources will be created.
- 4 Specify the directory containing the files of **CredentialsRequest** manifests to create GCP service accounts.

**NOTE**

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccctl_output_dir>/manifests** directory:

```
$ ls <path_to_ccctl_output_dir>/manifests
```

Example output

```
cluster-authentication-02-config.yaml
openshift-cloud-controller-manager-gcp-ccm-cloud-credentials-credentials.yaml
openshift-cloud-credential-operator-cloud-credential-operator-gcp-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capg-manager-bootstrap-credentials-credentials.yaml
```

```
openshift-cluster-csi-drivers-gcp-pd-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-gcp-cloud-credentials-credentials.yaml
```

You can verify that the IAM service accounts are created by querying GCP. For more information, refer to GCP documentation on listing IAM service accounts.

7.8.2.3. Incorporating the Cloud Credential Operator utility manifests

To implement short-term security credentials managed outside the cluster for individual components, you must move the manifest files that the Cloud Credential Operator utility (**ccctl**) created to the correct directories for the installation program.

Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have configured the Cloud Credential Operator utility (**ccctl**).
- You have created the cloud provider resources that are required for your cluster with the **ccctl** utility.

Procedure

1. Add the following granular permissions to the GCP account that the installation program uses:

Example 7.4. Required GCP permissions

- compute.machineTypes.list
- compute.regions.list
- compute.zones.list
- dns.changes.create
- dns.changes.get
- dns.managedZones.create
- dns.managedZones.delete
- dns.managedZones.get
- dns.managedZones.list
- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

- If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

Sample configuration file snippet

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

- If you have not previously created installation manifest files, do so by running the following command:

```
$ openshift-install create manifests --dir <installation_directory>
```

where **<installation_directory>** is the directory in which the installation program creates files.

- Copy the manifests that the **ccctl** utility generated to the **manifests** directory that the installation program created by running the following command:

```
$ cp /<path_to_ccctl_output_dir>/manifests/* ./manifests/
```

- Copy the **tls** directory that contains the private key to the installation directory:

```
$ cp -a /<path_to_ccctl_output_dir>/tls .
```

7.9. DEPLOYING THE CLUSTER

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- You have verified that the cloud provider account on your host has the correct permissions to deploy the cluster. An account with incorrect permissions causes the installation process to fail with an error message that displays the missing permissions.

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. Change to the directory that contains the installation program and initialize the cluster deployment:

```
$ ./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**.

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.
 - If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
 - If you included the **Service Account Key Admin** role, you can remove it.

Verification

When the cluster deployment completes successfully:

- The terminal displays directions for accessing your cluster, including a link to the web console and credentials for the **kubeadmin** user.
- Credential information also outputs to `<installation_directory>/openshift_install.log`.



IMPORTANT

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

Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

7.10. LOGGING IN TO THE CLUSTER BY USING THE CLI

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

- You deployed an OpenShift Container Platform cluster.
- You installed 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
```

Example output

```
system:admin
```

Additional resources

- See [Accessing the web console](#) for more details about accessing and understanding the OpenShift Container Platform web console.

7.11. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.18, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to [OpenShift Cluster Manager](#).

After you confirm that your [OpenShift Cluster Manager](#) inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See [About remote health monitoring](#) for more information about the Telemetry service

7.12. NEXT STEPS

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

CHAPTER 8. INSTALLING A CLUSTER ON GCP INTO A SHARED VPC

In OpenShift Container Platform version 4.18, you can install a cluster into a shared Virtual Private Cloud (VPC) on Google Cloud Platform (GCP). In this installation method, the cluster is configured to use a VPC from a different GCP project. A shared VPC enables an organization to connect resources from multiple projects to a common VPC network. You can communicate within the organization securely and efficiently by using internal IP addresses from that network. For more information about shared VPC, see [Shared VPC overview in the GCP documentation](#).

The installation program provisions the rest of the required infrastructure, which you can further customize. To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

8.1. PREREQUISITES

- You reviewed details about the [OpenShift Container Platform installation and update processes](#).
- You read the documentation on [selecting a cluster installation method and preparing it for users](#).
- If you use a firewall, you [configured it to allow the sites](#) that your cluster requires access to.
- You have a GCP host project which contains a shared VPC network.
- You [configured a GCP project](#) to host the cluster. This project, known as the service project, must be attached to the host project. For more information, see [Attaching service projects in the GCP documentation](#).
- You have a GCP service account that has the [required GCP permissions](#) in both the host and service projects.
- If you want to provide your own private hosted zone, you must have created one in the service project with the DNS pattern **cluster-name.baseDomain**, for example **testCluster.example.com**. The private hosted zone must be bound to the VPC in the host project. For more information about cross-project binding, see [Create a zone with cross-project binding](#) (Google documentation). If you do not provide a private hosted zone, the installation program will provision one automatically.

8.2. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM

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

You must have internet access to:

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



IMPORTANT

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

8.3. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS

During an OpenShift Container Platform installation, you can provide an SSH public key to the installation program. The key is passed to the Red Hat Enterprise Linux CoreOS (RHCOS) nodes through their Ignition config files and is used to authenticate SSH access to the nodes. The key is added to the `~/.ssh/authorized_keys` list for the **core** user on each node, which enables password-less authentication.

After the key is passed to the nodes, you can use the key pair to SSH in to the RHCOS nodes as the user **core**. To access the nodes through SSH, the private key identity must be managed by SSH for your local user.

If you want to SSH in to your cluster nodes to perform installation debugging or disaster recovery, you must provide the SSH public key during the installation process. The `./openshift-install gather` command also requires the SSH public key to be in place on the cluster nodes.



IMPORTANT

Do not skip this procedure in production environments, where disaster recovery and debugging is required.



NOTE

You must use a local key, not one that you configured with platform-specific approaches.

Procedure

- 1 If you do not have an existing SSH key pair on your local machine to use for authentication onto your cluster nodes, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" -f <path>/<file_name> ①
```

- 1 Specify the path and file name, such as `~/.ssh/id_ed25519`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the **x86_64**, **ppc64le**, and **s390x** architectures, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. View the public SSH key:

```
$ cat <path>/<file_name>.pub
```

For example, run the following to view the `~/.ssh/id_ed25519.pub` public key:

```
$ cat ~/.ssh/id_ed25519.pub
```

3. Add the SSH private key identity to the SSH agent for your local user, if it has not already been added. SSH agent management of the key is required for password-less SSH authentication onto your cluster nodes, or if you want to use the `./openshift-install gather` command.



NOTE

On some distributions, default SSH private key identities such as `~/.ssh/id_rsa` and `~/.ssh/id_dsa` are managed automatically.

- If the `ssh-agent` process is not already running for your local user, start it as a background task:

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

Example output

```
Agent pid 31874
```



NOTE

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

4. Add your SSH private key to the `ssh-agent`:

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

1

- Specify the path and file name for your SSH private key, such as `~/.ssh/id_ed25519`

Example output

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

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

8.4. OBTAINING THE INSTALLATION PROGRAM

Before you install OpenShift Container Platform, download the installation file on the host you are using for installation.

Prerequisites

- You have a computer that runs Linux or macOS, with 500 MB of local disk space.

Procedure

1. Go to the [Cluster Type](#) page on the Red Hat Hybrid Cloud Console. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

TIP

You can also [download the binaries for a specific OpenShift Container Platform release](#).

2. Select your infrastructure provider from the **Run it yourself** section of the page.
3. Select your host operating system and architecture from the dropdown menus under **OpenShift Installer** and click **Download Installer**.
4. Place the downloaded file in the directory where you want to 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 the installation program and the files that the installation program creates after you finish installing the cluster. Both of the files are required to delete the cluster.
- Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. To remove your cluster, complete the OpenShift Container Platform uninstallation procedures for your specific cloud provider.

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

```
$ tar -xvf openshift-install-linux.tar.gz
```

6. Download your installation [pull secret from Red Hat OpenShift Cluster Manager](#). 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.

TIP

Alternatively, you can retrieve the installation program from the [Red Hat Customer Portal](#), where you can specify a version of the installation program to download. However, you must have an active subscription to access this page.

8.5. CREATING THE INSTALLATION FILES FOR GCP

To install OpenShift Container Platform on Google Cloud Platform (GCP) into a shared VPC, you must generate the **install-config.yaml** file and modify it so that the cluster uses the correct VPC networks, DNS zones, and project names.

8.5.1. Manually creating the installation configuration file

Installing the cluster requires that you manually create the installation configuration file.

Prerequisites

- You have an SSH public key on your local machine to provide to the installation program. The key will be used for SSH authentication onto your cluster nodes for debugging and disaster recovery.
- You have obtained the OpenShift Container Platform installation program and the pull secret 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 sample **install-config.yaml** file template that is provided 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.

Additional resources

- [Installation configuration parameters for GCP](#)

8.5.2. Enabling Shielded VMs

You can use Shielded VMs when installing your cluster. Shielded VMs have extra security features including secure boot, firmware and integrity monitoring, and rootkit detection. For more information, see Google's documentation on [Shielded VMs](#).



NOTE

Shielded VMs are currently not supported on clusters with 64-bit ARM infrastructures.

Procedure

- Use a text editor to edit the **install-config.yaml** file prior to deploying your cluster and add one of the following stanzas:

- To use shielded VMs for only control plane machines:

```
controlPlane:
  platform:
    gcp:
      secureBoot: Enabled
```

- To use shielded VMs for only compute machines:

```
compute:
- platform:
  gcp:
    secureBoot: Enabled
```

- To use shielded VMs for all machines:

```
platform:
  gcp:
    defaultMachinePlatform:
      secureBoot: Enabled
```

8.5.3. Enabling Confidential VMs

You can use Confidential VMs when installing your cluster. Confidential VMs encrypt data while it is being processed. For more information, see Google's documentation on [Confidential Computing](#). You can enable Confidential VMs and Shielded VMs at the same time, although they are not dependent on each other.



NOTE

Confidential VMs are currently not supported on 64-bit ARM architectures.

Procedure

- Use a text editor to edit the **install-config.yaml** file prior to deploying your cluster and add one of the following stanzas:

- To use confidential VMs for only control plane machines:

```
controlPlane:
  platform:
```

```
gcp:
  confidentialCompute: Enabled ①
  type: n2d-standard-8 ②
  onHostMaintenance: Terminate ③
```

- ① Enable confidential VMs.
- ② Specify a machine type that supports Confidential VMs. Confidential VMs require the N2D or C2D series of machine types. For more information on supported machine types, see [Supported operating systems and machine types](#).
- ③ Specify the behavior of the VM during a host maintenance event, such as a hardware or software update. For a machine that uses Confidential VM, this value must be set to **Terminate**, which stops the VM. Confidential VMs do not support live VM migration.

- b. To use confidential VMs for only compute machines:

```
compute:
- platform:
  gcp:
    confidentialCompute: Enabled
    type: n2d-standard-8
    onHostMaintenance: Terminate
```

- c. To use confidential VMs for all machines:

```
platform:
gcp:
  defaultMachinePlatform:
    confidentialCompute: Enabled
    type: n2d-standard-8
    onHostMaintenance: Terminate
```

8.5.4. Sample customized `install-config.yaml` file for shared VPC installation

There are several configuration parameters which are required to install OpenShift Container Platform on GCP using a shared VPC. The following is a sample `install-config.yaml` file which demonstrates these fields.



IMPORTANT

This sample YAML file is provided for reference only. You must modify this file with the correct values for your environment and cluster.

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Passthrough ①
metadata:
  name: cluster_name
platform:
  gcp:
    computeSubnet: shared-vpc-subnet-1 ②
```

```

controlPlaneSubnet: shared-vpc-subnet-2 3
network: shared-vpc 4
networkProjectID: host-project-name 5
projectId: service-project-name 6
region: us-east1
defaultMachinePlatform:
tags: 7
- global-tag1
controlPlane:
name: master
platform:
gcp:
tags: 8
- control-plane-tag1
type: n2-standard-4
zones:
- us-central1-a
- us-central1-c
replicas: 3
compute:
- name: worker
platform:
gcp:
tags: 9
- compute-tag1
type: n2-standard-4
zones:
- us-central1-a
- us-central1-c
replicas: 3
networking:
clusterNetwork:
- cidr: 10.128.0.0/14
hostPrefix: 23
machineNetwork:
- cidr: 10.0.0.0/16
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA... 10

```

- 1** **credentialsMode** must be set to **Passthrough** or **Manual**. See the "Prerequisites" section for the required GCP permissions that your service account must have.
- 2** The name of the subnet in the shared VPC for compute machines to use.
- 3** The name of the subnet in the shared VPC for control plane machines to use.
- 4** The name of the shared VPC.
- 5** The name of the host project where the shared VPC exists.
- 6** The name of the GCP project where you want to install the cluster.
- 7** **8** **9** Optional. One or more network tags to apply to compute machines, control plane machines, or all machines.

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

8.5.5. 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

- You have an existing **install-config.yaml** file.
- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

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: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
  additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

- 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.
- 3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass the proxy for all destinations.

- 4** If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates
- 5** Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.

**NOTE**

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

**NOTE**

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

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.

8.6. INSTALLING THE OPENSHIFT CLI

You can install the OpenShift CLI (**oc**) to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.18. Download and install the new version of **oc**.

Installing the OpenShift CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** drop-down list.
3. Select the appropriate version from the **Version** drop-down list.

4. Click **Download Now** next to the **OpenShift v4.18 Linux Clients** entry and save the file.

5. Unpack the archive:

```
$ tar xvf <file>
```

6. Place the **oc** binary in a directory that is on your **PATH**.

To check your **PATH**, execute the following command:

```
$ echo $PATH
```

Verification

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

```
$ oc <command>
```

Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.

2. Select the appropriate version from the **Version** drop-down list.

3. Click **Download Now** next to the **OpenShift v4.18 Windows Client** entry and save the file.

4. Unzip the archive with a ZIP program.

5. Move the **oc** binary to a directory that is on your **PATH**.

To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

Verification

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

```
C:\> oc <command>
```

Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.

2. Select the appropriate version from the **Version** drop-down list.

3. Click **Download Now** next to the **OpenShift v4.18 macOS Clients** entry and save the file.

**NOTE**

For macOS arm64, choose the **OpenShift v4.18 macOS arm64 Cliententry**.

4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.

To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

Verification

- Verify your installation by using an **oc** command:

```
$ oc <command>
```

8.7. ALTERNATIVES TO STORING ADMINISTRATOR-LEVEL SECRETS IN THE KUBE-SYSTEM PROJECT

By default, administrator secrets are stored in the **kube-system** project. If you configured the **credentialsMode** parameter in the **install-config.yaml** file to **Manual**, you must use one of the following alternatives:

- To manage long-term cloud credentials manually, follow the procedure in [Manually creating long-term credentials](#).
- To implement short-term credentials that are managed outside the cluster for individual components, follow the procedures in [Configuring a GCP cluster to use short-term credentials](#).

8.7.1. Manually creating long-term credentials

The Cloud Credential Operator (CCO) can be put into manual mode prior to installation in environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace.

Procedure

1. Add the following granular permissions to the GCP account that the installation program uses:

Example 8.1. Required GCP permissions

- compute.machineTypes.list
- compute.regions.list
- compute.zones.list
- dns.changes.create
- dns.changes.get
- dns.managedZones.create

- dns.managedZones.delete
- dns.managedZones.get
- dns.managedZones.list
- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

2. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

Sample configuration file snippet

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

3. If you have not previously created installation manifest files, do so by running the following command:

```
$ openshift-install create manifests --dir <installation_directory>
```

where **<installation_directory>** is the directory in which the installation program creates files.

4. Set a **\$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3})
```

5. Extract the list of **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
--from=$RELEASE_IMAGE \
--credentials-requests \
--included ① \
--install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \② \
--to=<path_to_directory_for_credentials_requests> ③
```

1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

2 Specify the location of the **install-config.yaml** file.

3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

This command creates a YAML file for each **CredentialsRequest** object.

Sample CredentialsRequest object

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: GCPProviderSpec
    predefinedRoles:
      - roles/storage.admin
      - roles/iam.serviceAccountUser
    skipServiceCheck: true
...
```

6. Create YAML files for secrets in the **openshift-install** manifests directory that you generated previously. The secrets must be stored using the namespace and secret name defined in the **spec.secretRef** for each **CredentialsRequest** object.

Sample CredentialsRequest object with secrets

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
...

```

Sample Secret object

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  service_account.json: <base64_encoded_gcp_service_account_file>
```



IMPORTANT

Before upgrading a cluster that uses manually maintained credentials, you must ensure that the CCO is in an upgradeable state.

8.7.2. Configuring a GCP cluster to use short-term credentials

To install a cluster that is configured to use GCP Workload Identity, you must configure the CCO utility and create the required GCP resources for your cluster.

8.7.2.1. Configuring the Cloud Credential Operator utility

To create and manage cloud credentials from outside of the cluster when the Cloud Credential Operator (CCO) is operating in manual mode, extract and prepare the CCO utility (**ccocctl**) binary.



NOTE

The **ccocctl** utility is a Linux binary that must run in a Linux environment.

Prerequisites

- You have access to an OpenShift Container Platform account with cluster administrator access.
- You have installed the OpenShift CLI (**oc**).
- You have added one of the following authentication options to the GCP account that the **ccocctl** utility uses:
 - The **IAM Workload Identity Pool Adminrole**
 - The following granular permissions:
 - **compute.projects.get**
 - **iam.googleapis.com/workloadIdentityPoolProviders.create**
 - **iam.googleapis.com/workloadIdentityPoolProviders.get**
 - **iam.googleapis.com/workloadIdentityPools.create**
 - **iam.googleapis.com/workloadIdentityPools.delete**
 - **iam.googleapis.com/workloadIdentityPools.get**
 - **iam.googleapis.com/workloadIdentityPools.undelete**
 - **iam.roles.create**
 - **iam.roles.delete**
 - **iam.roles.list**
 - **iam.roles.undelete**
 - **iam.roles.update**
 - **iam.serviceAccounts.create**

- **iam.serviceAccounts.delete**
- **iam.serviceAccounts.getIamPolicy**
- **iam.serviceAccounts.list**
- **iam.serviceAccounts.setIamPolicy**
- **iam.workloadIdentityPoolProviders.get**
- **iam.workloadIdentityPools.delete**
- **resourcemanager.projects.get**
- **resourcemanager.projects.getIamPolicy**
- **resourcemanager.projects.setIamPolicy**
- **storage.buckets.create**
- **storage.buckets.delete**
- **storage.buckets.get**
- **storage.buckets.getIamPolicy**
- **storage.buckets.setIamPolicy**
- **storage.objects.create**
- **storage.objects.delete**
- **storage.objects.list**

Procedure

1. Set a variable for the OpenShift Container Platform release image by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Obtain the CCO container image from the OpenShift Container Platform release image by running the following command:

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'  
$RELEASE_IMAGE -a ~/.pull-secret)
```



NOTE

Ensure that the architecture of the **\$RELEASE_IMAGE** matches the architecture of the environment in which you will use the **ccctl** tool.

3. Extract the **ccctl** binary from the CCO container image within the OpenShift Container Platform release image by running the following command:

```
$ oc image extract $CCO_IMAGE \
--file="/usr/bin/ccctl.<rhel_version>" \①
-a ~/.pull-secret
```

- 1 For **<rhel_version>**, specify the value that corresponds to the version of Red Hat Enterprise Linux (RHEL) that the host uses. If no value is specified, **ccctl.rhel8** is used by default. The following values are valid:

- **rhel8**: Specify this value for hosts that use RHEL 8.
- **rhel9**: Specify this value for hosts that use RHEL 9.

4. Change the permissions to make **ccctl** executable by running the following command:

```
$ chmod 775 ccctl.<rhel_version>
```

Verification

- To verify that **ccctl** is ready to use, display the help file. Use a relative file name when you run the command, for example:

```
$ ./ccctl.rhel9
```

Example output

```
OpenShift credentials provisioning tool

Usage:
  ccctl [command]

Available Commands:
  aws      Manage credentials objects for AWS cloud
  azure    Manage credentials objects for Azure
  gcp      Manage credentials objects for Google cloud
  help     Help about any command
  ibmcloud Manage credentials objects for {ibm-cloud-title}
  nutanix   Manage credentials objects for Nutanix

Flags:
  -h, --help  help for ccctl

Use "ccctl [command] --help" for more information about a command.
```

8.7.2.2. Creating GCP resources with the Cloud Credential Operator utility

You can use the **ccctl gcp create-all** command to automate the creation of GCP resources.



NOTE

By default, **ccctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccctl_output_dir>** to refer to this directory.

Prerequisites

You must have:

- Extracted and prepared the **ccocctl** binary.

Procedure

- Set a **\$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

- Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
--from=$RELEASE_IMAGE \
--credentials-requests \
--included \① \
--install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \② \
--to=<path_to_directory_for_credentials_requests> \③
```

① The **--included** parameter includes only the manifests that your specific cluster configuration requires.

② Specify the location of the **install-config.yaml** file.

③ Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.



NOTE

This command might take a few moments to run.

- Use the **ccocctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccocctl gcp create-all \
--name=<name> \① \
--region=<gcp_region> \② \
--project=<gcp_project_id> \③ \
--credentials-requests-dir=<path_to_credentials_requests_directory> \④
```

① Specify the user-defined name for all created GCP resources used for tracking. If you plan to install the GCP Filestore Container Storage Interface (CSI) Driver Operator, retain this value.

② Specify the GCP region in which cloud resources will be created.

③ Specify the GCP project ID in which cloud resources will be created.

④ Specify the directory containing the files of **CredentialsRequest** manifests to create GCP

**NOTE**

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

Example output

```
cluster-authentication-02-config.yaml
openshift-cloud-controller-manager-gcp-ccm-cloud-credentials-credentials.yaml
openshift-cloud-credential-operator-cloud-credential-operator-gcp-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capg-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-gcp-pd-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-gcp-cloud-credentials-credentials.yaml
```

You can verify that the IAM service accounts are created by querying GCP. For more information, refer to GCP documentation on listing IAM service accounts.

8.7.2.3. Incorporating the Cloud Credential Operator utility manifests

To implement short-term security credentials managed outside the cluster for individual components, you must move the manifest files that the Cloud Credential Operator utility (**ccoctl**) created to the correct directories for the installation program.

Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have configured the Cloud Credential Operator utility (**ccoctl**).
- You have created the cloud provider resources that are required for your cluster with the **ccoctl** utility.

Procedure

1. Add the following granular permissions to the GCP account that the installation program uses:

Example 8.2. Required GCP permissions

- compute.machineTypes.list
- compute.regions.list
- compute.zones.list

- dns.changes.create
- dns.changes.get
- dns.managedZones.create
- dns.managedZones.delete
- dns.managedZones.get
- dns.managedZones.list
- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

2. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

Sample configuration file snippet

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

3. If you have not previously created installation manifest files, do so by running the following command:

```
$ openshift-install create manifests --dir <installation_directory>
```

where **<installation_directory>** is the directory in which the installation program creates files.

4. Copy the manifests that the **ccctl** utility generated to the **manifests** directory that the installation program created by running the following command:

```
$ cp /<path_to_ccctl_output_dir>/manifests/* ./manifests/
```

5. Copy the **tls** directory that contains the private key to the installation directory:

```
$ cp -a /<path_to_ccctl_output_dir>/tls .
```

8.8. DEPLOYING THE CLUSTER

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- You have verified that the cloud provider account on your host has the correct permissions to deploy the cluster. An account with incorrect permissions causes the installation process to fail with an error message that displays the missing permissions.

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

2. Change to the directory that contains the installation program and initialize the cluster deployment:

```
$ ./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**.

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.

- If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
- If you included the **Service Account Key Admin** role, you can remove it.

Verification

When the cluster deployment completes successfully:

- The terminal displays directions for accessing your cluster, including a link to the web console and credentials for the **kubeadmin** user.
- Credential information also outputs to **<installation_directory>/openshift_install.log**.



IMPORTANT

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

Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```



IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

8.9. LOGGING IN TO THE CLUSTER BY USING THE CLI

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

- You deployed an OpenShift Container Platform cluster.
- You installed 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
```

Example output

```
system:admin
```

Additional resources

- See [Accessing the web console](#) for more details about accessing and understanding the OpenShift Container Platform web console.

8.10. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.18, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to [OpenShift Cluster Manager](#).

After you confirm that your [OpenShift Cluster Manager](#) inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See [About remote health monitoring](#) for more information about the Telemetry service

8.11. NEXT STEPS

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

CHAPTER 9. INSTALLING A PRIVATE CLUSTER ON GCP

In OpenShift Container Platform version 4.18, you can install a private cluster into an existing VPC on Google Cloud Platform (GCP). The installation program provisions the rest of the required infrastructure, which you can further customize. To customize the installation, you modify parameters in the `install-config.yaml` file before you install the cluster.

9.1. PREREQUISITES

- You reviewed details about the [OpenShift Container Platform installation and update processes](#).
- You read the documentation on [selecting a cluster installation method and preparing it for users](#).
- You [configured a GCP project](#) to host the cluster.
- If you use a firewall, you [configured it to allow the sites](#) that your cluster requires access to.

9.2. PRIVATE CLUSTERS

You can deploy a private OpenShift Container Platform cluster that does not expose external endpoints. Private clusters are accessible from only an internal network and are not visible to the internet.

By default, OpenShift Container Platform is provisioned to use publicly-accessible DNS and endpoints. A private cluster sets the DNS, Ingress Controller, and API server to private when you deploy your cluster. This means that the cluster resources are only accessible from your internal network and are not visible to the internet.



IMPORTANT

If the cluster has any public subnets, load balancer services created by administrators might be publicly accessible. To ensure cluster security, verify that these services are explicitly annotated as private.

To deploy a private cluster, you must:

- Use existing networking that meets your requirements. Your cluster resources might be shared between other clusters on the network.
- Deploy from a machine that has access to:
 - The API services for the cloud to which you provision.
 - The hosts on the network that you provision.
 - The internet to obtain installation media.

You can use any machine that meets these access requirements and follows your company's guidelines. For example, this machine can be a bastion host on your cloud network or a machine that has access to the network through a VPN.

9.2.1. Private clusters in GCP

To create a private cluster on Google Cloud Platform (GCP), you must provide an existing private VPC and subnets to host the cluster. The installation program must also be able to resolve the DNS records that the cluster requires. The installation program configures the Ingress Operator and API server for only internal traffic.

The cluster still requires access to internet to access the GCP APIs.

The following items are not required or created when you install a private cluster:

- Public subnets
- Public network load balancers, which support public ingress
- A public DNS zone that matches the **baseDomain** for the cluster

The installation program does use the **baseDomain** that you specify to create a private DNS zone and the required records for the cluster. The cluster is configured so that the Operators do not create public records for the cluster and all cluster machines are placed in the private subnets that you specify.

Because it is not possible to limit access to external load balancers based on source tags, the private cluster uses only internal load balancers to allow access to internal instances.

The internal load balancer relies on instance groups rather than the target pools that the network load balancers use. The installation program creates instance groups for each zone, even if there is no instance in that group.

- The cluster IP address is internal only.
- One forwarding rule manages both the Kubernetes API and machine config server ports.
- The backend service is comprised of each zone's instance group and, while it exists, the bootstrap instance group.
- The firewall uses a single rule that is based on only internal source ranges.

9.2.1.1. Limitations

No health check for the Machine config server, `/healthz`, runs because of a difference in load balancer functionality. Two internal load balancers cannot share a single IP address, but two network load balancers can share a single external IP address. Instead, the health of an instance is determined entirely by the `/readyz` check on port 6443.

9.3. ABOUT USING A CUSTOM VPC

In OpenShift Container Platform 4.18, you can deploy a cluster into an existing VPC in Google Cloud Platform (GCP). If you do, you must also use existing subnets within the VPC and routing rules.

By deploying OpenShift Container Platform into an existing GCP VPC, you might be able to avoid limit constraints in new accounts or more easily abide by the operational constraints that your company's guidelines set. This is a good option to use if you cannot obtain the infrastructure creation permissions that are required to create the VPC yourself.

9.3.1. Requirements for using your VPC

The installation program will no longer create the following components:

- VPC
- Subnets
- Cloud router
- Cloud NAT
- NAT IP addresses

If you use a custom VPC, you must correctly configure it and its subnets for the installation program and the cluster to use. The installation program cannot subdivide network ranges for the cluster to use, set route tables for the subnets, or set VPC options like DHCP, so you must do so before you install the cluster.

Your VPC and subnets must meet the following characteristics:

- The VPC must be in the same GCP project that you deploy the OpenShift Container Platform cluster to.
- To allow access to the internet from the control plane and compute machines, you must configure cloud NAT on the subnets to allow egress to it. These machines do not have a public address. Even if you do not require access to the internet, you must allow egress to the VPC network to obtain the installation program and images. Because multiple cloud NATs cannot be configured on the shared subnets, the installation program cannot configure it.

To ensure that the subnets that you provide are suitable, the installation program confirms the following data:

- All the subnets that you specify exist and belong to the VPC that you specified.
- The subnet CIDRs belong to the machine CIDR.
- You must provide a subnet to deploy the cluster control plane and compute machines to. You can use the same subnet for both machine types.

If you destroy a cluster that uses an existing VPC, the VPC is not deleted.

9.3.2. Division of permissions

Starting with OpenShift Container Platform 4.3, you do not need all of the permissions that are required for an installation program-provisioned infrastructure cluster to deploy a cluster. This change mimics the division of permissions that you might have at your company: some individuals can create different resources in your clouds than others. For example, you might be able to create application-specific items, like instances, buckets, and load balancers, but not networking-related components such as VPCs, subnets, or Ingress rules.

The GCP credentials that you use when you create your cluster do not need the networking permissions that are required to make VPCs and core networking components within the VPC, such as subnets, routing tables, internet gateways, NAT, and VPN. You still need permission to make the application resources that the machines within the cluster require, such as load balancers, security groups, storage, and nodes.

9.3.3. Isolation between clusters

If you deploy OpenShift Container Platform to an existing network, the isolation of cluster services is preserved by firewall rules that reference the machines in your cluster by the cluster's infrastructure ID. Only traffic within the cluster is allowed.

If you deploy multiple clusters to the same VPC, the following components might share access between clusters:

- The API, which is globally available with an external publishing strategy or available throughout the network in an internal publishing strategy
- Debugging tools, such as ports on VM instances that are open to the machine CIDR for SSH and ICMP access

9.4. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM

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

You must have internet access to:

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



IMPORTANT

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

9.5. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS

During an OpenShift Container Platform installation, you can provide an SSH public key to the installation program. The key is passed to the Red Hat Enterprise Linux CoreOS (RHCOS) nodes through their Ignition config files and is used to authenticate SSH access to the nodes. The key is added to the `~/.ssh/authorized_keys` list for the **core** user on each node, which enables password-less authentication.

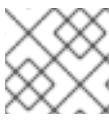
After the key is passed to the nodes, you can use the key pair to SSH in to the RHCOS nodes as the user **core**. To access the nodes through SSH, the private key identity must be managed by SSH for your local user.

If you want to SSH in to your cluster nodes to perform installation debugging or disaster recovery, you must provide the SSH public key during the installation process. The `./openshift-install gather` command also requires the SSH public key to be in place on the cluster nodes.



IMPORTANT

Do not skip this procedure in production environments, where disaster recovery and debugging is required.



NOTE

You must use a local key, not one that you configured with platform-specific approaches.

Procedure

1. If you do not have an existing SSH key pair on your local machine to use for authentication onto your cluster nodes, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" -f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_ed25519`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the **x86_64**, **ppc64le**, and **s390x** architectures, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. View the public SSH key:

```
$ cat <path>/<file_name>.pub
```

For example, run the following to view the `~/.ssh/id_ed25519.pub` public key:

```
$ cat ~/.ssh/id_ed25519.pub
```

3. Add the SSH private key identity to the SSH agent for your local user, if it has not already been added. SSH agent management of the key is required for password-less SSH authentication onto your cluster nodes, or if you want to use the `./openshift-install gather` command.



NOTE

On some distributions, default SSH private key identities such as `~/.ssh/id_rsa` and `~/.ssh/id_dsa` are managed automatically.

- a. If the **ssh-agent** process is not already running for your local user, start it as a background task:

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

Example output

Agent pid 31874



NOTE

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

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

\$ ssh-add <path>/<file_name> ①

- ① Specify the path and file name for your SSH private key, such as `~/.ssh/id_ed25519`

Example output

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

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

9.6. OBTAINING THE INSTALLATION PROGRAM

Before you install OpenShift Container Platform, download the installation file on the host you are using for installation.

Prerequisites

- You have a computer that runs Linux or macOS, with 500 MB of local disk space.

Procedure

1. Go to the [Cluster Type](#) page on the Red Hat Hybrid Cloud Console. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

TIP

You can also [download the binaries for a specific OpenShift Container Platform release](#).

2. Select your infrastructure provider from the **Run it yourself** section of the page.
3. Select your host operating system and architecture from the dropdown menus under **OpenShift Installer** and click **Download Installer**.
4. Place the downloaded file in the directory where you want to 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 the installation program and the files that the installation program creates after you finish installing the cluster. Both of the files are required to delete the cluster.
- Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. To remove your cluster, complete the OpenShift Container Platform uninstallation procedures for your specific cloud provider.

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

```
$ tar -xvf openshift-install-linux.tar.gz
```

6. Download your installation [pull secret from Red Hat OpenShift Cluster Manager](#). 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.

TIP

Alternatively, you can retrieve the installation program from the [Red Hat Customer Portal](#), where you can specify a version of the installation program to download. However, you must have an active subscription to access this page.

9.7. MANUALLY CREATING THE INSTALLATION CONFIGURATION FILE

Installing the cluster requires that you manually create the installation configuration file.

Prerequisites

- You have an SSH public key on your local machine to provide to the installation program. The key will be used for SSH authentication onto your cluster nodes for debugging and disaster recovery.
- You have obtained the OpenShift Container Platform installation program and the pull secret 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 sample **install-config.yaml** file template that is provided 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.

Additional resources

- [Installation configuration parameters for GCP](#)

9.7.1. Minimum resource requirements for cluster installation

Each cluster machine must meet the following minimum requirements:

Table 9.1. Minimum resource requirements

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

- One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.
- OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so

you might need to over-allocate storage volume to obtain sufficient performance.

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



NOTE

For OpenShift Container Platform version 4.18, RHCOS is based on RHEL version 9.4, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see [Architectures](#) (RHEL documentation).

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

Additional resources

- [Optimizing storage](#)

9.7.2. Tested instance types for GCP

The following Google Cloud Platform instance types have been tested with OpenShift Container Platform.



NOTE

Not all instance types are available in all regions and zones. For a detailed breakdown of which instance types are available in which zones, see [regions and zones](#) (Google documentation).

Some instance types require the use of Hyperdisk storage. If you use an instance type that requires Hyperdisk storage, all of the nodes in your cluster must support Hyperdisk storage, and you must change the default storage class to use Hyperdisk storage. For more information, see [machine series support for Hyperdisk](#) (Google documentation). For instructions on modifying storage classes, see the "GCE PersistentDisk (gcePD) object definition" section in the Dynamic Provisioning page in *Storage*.

Example 9.1. Machine series

- A2
- A3

- C2
- C2D
- C3
- C3D
- C4
- E2
- M1
- N1
- N2
- N2D
- N4
- Tau T2D

9.7.3. Tested instance types for GCP on 64-bit ARM infrastructures

The following Google Cloud Platform (GCP) 64-bit ARM instance types have been tested with OpenShift Container Platform.

Example 9.2. Machine series for 64-bit ARM machines

- C4A
- Tau T2A

9.7.4. Using custom machine types

Using a custom machine type to install a OpenShift Container Platform cluster is supported.

Consider the following when using a custom machine type:

- Similar to predefined instance types, custom machine types must meet the minimum resource requirements for control plane and compute machines. For more information, see "Minimum resource requirements for cluster installation".
- The name of the custom machine type must adhere to the following syntax:
custom-<number_of_cpus>-<amount_of_memory_in_mb>

For example, **custom-6-20480**.

As part of the installation process, you specify the custom machine type in the **install-config.yaml** file.

Sample **install-config.yaml** file with a custom machine type

```

compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    gcp:
      type: custom-6-20480
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    gcp:
      type: custom-6-20480
  replicas: 3

```

9.7.5. Enabling Shielded VMs

You can use Shielded VMs when installing your cluster. Shielded VMs have extra security features including secure boot, firmware and integrity monitoring, and rootkit detection. For more information, see Google's documentation on [Shielded VMs](#).



NOTE

Shielded VMs are currently not supported on clusters with 64-bit ARM infrastructures.

Procedure

- Use a text editor to edit the **install-config.yaml** file prior to deploying your cluster and add one of the following stanzas:

- To use shielded VMs for only control plane machines:

```

controlPlane:
  platform:
    gcp:
      secureBoot: Enabled

```

- To use shielded VMs for only compute machines:

```

compute:
- platform:
  gcp:
    secureBoot: Enabled

```

- To use shielded VMs for all machines:

```

platform:
gcp:
  defaultMachinePlatform:
    secureBoot: Enabled

```

9.7.6. Enabling Confidential VMs

You can use Confidential VMs when installing your cluster. Confidential VMs encrypt data while it is being processed. For more information, see Google's documentation on [Confidential Computing](#). You can enable Confidential VMs and Shielded VMs at the same time, although they are not dependent on each other.



NOTE

Confidential VMs are currently not supported on 64-bit ARM architectures.

Procedure

- Use a text editor to edit the **install-config.yaml** file prior to deploying your cluster and add one of the following stanzas:

- To use confidential VMs for only control plane machines:

```
controlPlane:
  platform:
    gcp:
      confidentialCompute: Enabled ①
      type: n2d-standard-8 ②
      onHostMaintenance: Terminate ③
```

- ① Enable confidential VMs.
- ② Specify a machine type that supports Confidential VMs. Confidential VMs require the N2D or C2D series of machine types. For more information on supported machine types, see [Supported operating systems and machine types](#).
- ③ Specify the behavior of the VM during a host maintenance event, such as a hardware or software update. For a machine that uses Confidential VM, this value must be set to **Terminate**, which stops the VM. Confidential VMs do not support live VM migration.

- To use confidential VMs for only compute machines:

```
compute:
  - platform:
    gcp:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

- To use confidential VMs for all machines:

```
platform:
  gcp:
    defaultMachinePlatform:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate
```

9.7.7. 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
credentialsMode: Mint 2
controlPlane: 3 4
hyperthreading: Enabled 5
name: master
platform:
gcp:
  type: n2-standard-4
  zones:
    - us-central1-a
    - us-central1-c
osDisk:
  diskType: pd-ssd
  diskSizeGB: 1024
  encryptionKey: 6
  kmsKey:
    name: worker-key
    keyRing: test-machine-keys
    location: global
    projectId: project-id
tags: 7
  - control-plane-tag1
  - control-plane-tag2
osImage: 8
  project: example-project-name
  name: example-image-name
replicas: 3
compute: 9 10
  - hyperthreading: Enabled 11
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
        encryptionKey: 12
        kmsKey:
          name: worker-key
          keyRing: test-machine-keys

```

```

location: global
projectID: project-id
tags: ⑯
- compute-tag1
- compute-tag2
osImage: ⑯
  project: example-project-name
  name: example-image-name
replicas: 3
metadata:
  name: test-cluster ⑮
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  machineNetwork:
    - cidr: 10.0.0.0/16
  networkType: OVNKubernetes ⑯
  serviceNetwork:
    - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production ⑰
    region: us-central1 ⑱
    defaultMachinePlatform:
      tags: ⑲
      - global-tag1
      - global-tag2
      osImage: ⑳
        project: example-project-name
        name: example-image-name
      network: existing_vpc ㉑
      controlPlaneSubnet: control_plane_subnet ㉒
      computeSubnet: compute_subnet ㉓
    pullSecret: '{"auths": ...}' ㉔
    fips: false ㉕
    sshKey: ssh-ed25519 AAAA... ㉖
    publish: Internal ㉗

```

① ⑮ ⑯ ⑰ ⑱ ⑲ ⑳ ㉑ ㉒ ㉓ ㉔ ㉕ ㉖ ㉗ Required. The installation program prompts you for this value.

② Optional: Add this parameter to force the Cloud Credential Operator (CCO) to use the specified mode. By default, the CCO uses the root credentials in the **kube-system** namespace to dynamically try to determine the capabilities of the credentials. For details about CCO modes, see the "About the Cloud Credential Operator" section in the *Authentication and authorization* guide.

③ ⑨ If you do not provide these parameters and values, the installation program provides the default value.

④ ⑩ 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. Only one control plane pool is used.

- 5 11** Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You



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.

- 6 12** Optional: The custom encryption key section to encrypt both virtual machines and persistent volumes. Your default compute service account must have the permissions granted to use your KMS key and have the correct IAM role assigned. The default service account name follows the **service-<project_number>@compute-system.iam.gserviceaccount.com** pattern. For more information about granting the correct permissions for your service account, see "Machine management" → "Creating compute machine sets" → "Creating a compute machine set on GCP".

- 7 13 19** Optional: A set of network tags to apply to the control plane or compute machine sets. The **platform.gcp.defaultMachinePlatform.tags** parameter will apply to both control plane and compute machines. If the **compute.platform.gcp.tags** or **controlPlane.platform.gcp.tags** parameters are set, they override the **platform.gcp.defaultMachinePlatform.tags** parameter.

- 8 14 20** Optional: A custom Red Hat Enterprise Linux CoreOS (RHCOS) that should be used to boot control plane and compute machines. The **project** and **name** parameters under **platform.gcp.defaultMachinePlatform.osImage** apply to both control plane and compute machines. If the **project** and **name** parameters under **controlPlane.platform.gcp.osImage** or **compute.platform.gcp.osImage** are set, they override the **platform.gcp.defaultMachinePlatform.osImage** parameters.

- 16** The cluster network plugin to install. The default value **OVNKubernetes** is the only supported value.

- 21** Specify the name of an existing VPC.

- 22** Specify the name of the existing subnet to deploy the control plane machines to. The subnet must belong to the VPC that you specified.

- 23** Specify the name of the existing subnet to deploy the compute machines to. The subnet must belong to the VPC that you specified.

- 25** Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.



IMPORTANT

To enable FIPS mode for your cluster, you must run the installation program from a Red Hat Enterprise Linux (RHEL) computer configured to operate in FIPS mode. For more information about configuring FIPS mode on RHEL, see [Installing the system in FIPS mode](#).

When running Red Hat Enterprise Linux (RHEL) or Red Hat Enterprise Linux CoreOS (RHCOS) booted in FIPS mode, OpenShift Container Platform core components use the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the x86_64, ppc64le, and s390x architectures.

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

- 27** How to publish the user-facing endpoints of your cluster. Set **publish** to **Internal** to deploy a private cluster, which cannot be accessed from the internet. The default value is **External**.

Additional resources

- [Enabling customer-managed encryption keys for a compute machine set](#)

9.7.8. Create an Ingress Controller with global access on GCP

You can create an Ingress Controller that has global access to a Google Cloud Platform (GCP) cluster. Global access is only available to Ingress Controllers using internal load balancers.

Prerequisites

- You created the **install-config.yaml** and complete any modifications to it.

Procedure

Create an Ingress Controller with global access on a new GCP cluster.

1. Change to the directory that contains the installation program and create a manifest file:

```
$ ./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-ingress-default-ingresscontroller.yaml** in the **<installation_directory>/manifests** directory:

```
$ touch <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml 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-ingress-default-ingresscontroller.yaml
```

Example output

```
cluster-ingress-default-ingresscontroller.yaml
```

3. Open the **cluster-ingress-default-ingresscontroller.yaml** file in an editor and enter a custom resource (CR) that describes the Operator configuration you want:

Sample clientAccess configuration to Global

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      providerParameters:
        gcp:
          clientAccess: Global 1
          type: GCP
          scope: Internal 2
          type: LoadBalancerService
```

- 1 Set **gcp.clientAccess** to **Global**.

- 2 Global access is only available to Ingress Controllers using internal load balancers.

9.7.9. 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

- You have an existing **install-config.yaml** file.
- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (169.254.169.254).

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: https://<username>:<pswd>@<ip>:<port> ②
  noProxy: example.com ③
  additionalTrustBundle: | ④
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> ⑤
```

- 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.
- 3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with . to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use * to bypass the proxy for all destinations.
- 4 If provided, the installation program generates a config map 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** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **trustedCA** field of the **Proxy** object. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.
- 5 Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.



NOTE

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

**NOTE**

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

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.

9.8. INSTALLING THE OPENSHIFT CLI

You can install the OpenShift CLI (**oc**) to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.18. Download and install the new version of **oc**.

Installing the OpenShift CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** drop-down list.
3. Select the appropriate version from the **Version** drop-down list.
4. Click **Download Now** next to the **OpenShift v4.18 Linux Clients** entry and save the file.
5. Unpack the archive:

```
$ tar xvf <file>
```

6. Place the **oc** binary in a directory that is on your **PATH**. To check your **PATH**, execute the following command:

```
$ echo $PATH
```

Verification

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

```
$ oc <command>
```

Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

- Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
- Select the appropriate version from the **Version** drop-down list.
- Click **Download Now** next to the **OpenShift v4.18 Windows Client** entry and save the file.
- Unzip the archive with a ZIP program.
- Move the **oc** binary to a directory that is on your **PATH**.

To check your **PATH**, open the command prompt and execute the following command:

```
C:> path
```

Verification

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

```
C:> oc <command>
```

Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

- Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
- Select the appropriate version from the **Version** drop-down list.
- Click **Download Now** next to the **OpenShift v4.18 macOS Clients** entry and save the file.



NOTE

For macOS arm64, choose the **OpenShift v4.18 macOS arm64 Client** entry.

- Unpack and unzip the archive.

- Move the **oc** binary to a directory on your **PATH**.

To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

Verification

- Verify your installation by using an **oc** command:

```
$ oc <command>
```

9.9. ALTERNATIVES TO STORING ADMINISTRATOR-LEVEL SECRETS IN THE KUBE-SYSTEM PROJECT

By default, administrator secrets are stored in the **kube-system** project. If you configured the **credentialsMode** parameter in the **install-config.yaml** file to **Manual**, you must use one of the following alternatives:

- To manage long-term cloud credentials manually, follow the procedure in [Manually creating long-term credentials](#).
- To implement short-term credentials that are managed outside the cluster for individual components, follow the procedures in [Configuring a GCP cluster to use short-term credentials](#).

9.9.1. Manually creating long-term credentials

The Cloud Credential Operator (CCO) can be put into manual mode prior to installation in environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace.

Procedure

- Add the following granular permissions to the GCP account that the installation program uses:

Example 9.3. Required GCP permissions

- compute.machineTypes.list
- compute.regions.list
- compute.zones.list
- dns.changes.create
- dns.changes.get
- dns.managedZones.create
- dns.managedZones.delete
- dns.managedZones.get
- dns.managedZones.list
- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

- If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

Sample configuration file snippet

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

- If you have not previously created installation manifest files, do so by running the following command:

```
$ openshift-install create manifests --dir <installation_directory>
```

where **<installation_directory>** is the directory in which the installation program creates files.

- Set a **\$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

- Extract the list of **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
--from=$RELEASE_IMAGE \
--credentials-requests \
--included ① \
--install-config=<path_to_directory_with_installation_configuration>/install-config.yaml ② \
--to=<path_to_directory_for_credentials_requests> ③
```

- ① The **--included** parameter includes only the manifests that your specific cluster configuration requires.
- ② Specify the location of the **install-config.yaml** file.
- ③ Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

This command creates a YAML file for each **CredentialsRequest** object.

Sample CredentialsRequest object

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
```

```

apiVersion: cloudcredential.openshift.io/v1
kind: GCPProviderSpec
predefinedRoles:
- roles/storage.admin
- roles/iam.serviceAccountUser
skipServiceCheck: true
...

```

6. Create YAML files for secrets in the **openshift-install** manifests directory that you generated previously. The secrets must be stored using the namespace and secret name defined in the **spec.secretRef** for each **CredentialsRequest** object.

Sample **CredentialsRequest** object with secrets

```

apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...

```

Sample **Secret** object

```

apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  service_account.json: <base64_encoded_gcp_service_account_file>

```



IMPORTANT

Before upgrading a cluster that uses manually maintained credentials, you must ensure that the CCO is in an upgradeable state.

9.9.2. Configuring a GCP cluster to use short-term credentials

To install a cluster that is configured to use GCP Workload Identity, you must configure the CCO utility and create the required GCP resources for your cluster.

9.9.2.1. Configuring the Cloud Credential Operator utility

To create and manage cloud credentials from outside of the cluster when the Cloud Credential Operator (CCO) is operating in manual mode, extract and prepare the CCO utility (**ccctl**) binary.



NOTE

The **ccocctl** utility is a Linux binary that must run in a Linux environment.

Prerequisites

- You have access to an OpenShift Container Platform account with cluster administrator access.
- You have installed the OpenShift CLI (**oc**).
- You have added one of the following authentication options to the GCP account that the **ccocctl** utility uses:
 - The **IAM Workload Identity Pool Adminrole**
 - The following granular permissions:
 - **compute.projects.get**
 - **iam.googleapis.com/workloadIdentityPoolProviders.create**
 - **iam.googleapis.com/workloadIdentityPoolProviders.get**
 - **iam.googleapis.com/workloadIdentityPools.create**
 - **iam.googleapis.com/workloadIdentityPools.delete**
 - **iam.googleapis.com/workloadIdentityPools.get**
 - **iam.googleapis.com/workloadIdentityPools.undelete**
 - **iam.roles.create**
 - **iam.roles.delete**
 - **iam.roles.list**
 - **iam.roles.undelete**
 - **iam.roles.update**
 - **iam.serviceAccounts.create**
 - **iam.serviceAccounts.delete**
 - **iam.serviceAccounts.getiamPolicy**
 - **iam.serviceAccounts.list**
 - **iam.serviceAccounts.setiamPolicy**
 - **iam.workloadIdentityPoolProviders.get**
 - **iam.workloadIdentityPools.delete**
 - **resourcemanager.projects.get**
 - **resourcemanager.projects.getiamPolicy**

- **resourcemanager.projects.setIamPolicy**
- **storage.buckets.create**
- **storage.buckets.delete**
- **storage.buckets.get**
- **storage.buckets.getIamPolicy**
- **storage.buckets.setIamPolicy**
- **storage.objects.create**
- **storage.objects.delete**
- **storage.objects.list**

Procedure

1. Set a variable for the OpenShift Container Platform release image by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Obtain the CCO container image from the OpenShift Container Platform release image by running the following command:

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator' $RELEASE_IMAGE -a ~/.pull-secret)
```



NOTE

Ensure that the architecture of the **\$RELEASE_IMAGE** matches the architecture of the environment in which you will use the **ccctl** tool.

3. Extract the **ccctl** binary from the CCO container image within the OpenShift Container Platform release image by running the following command:

```
$ oc image extract $CCO_IMAGE \
--file="/usr/bin/ccctl.<rhel_version>" \①
-a ~/.pull-secret
```

- 1 For **<rhel_version>**, specify the value that corresponds to the version of Red Hat Enterprise Linux (RHEL) that the host uses. If no value is specified, **ccctl.rhel8** is used by default. The following values are valid:

- **rhel8**: Specify this value for hosts that use RHEL 8.
- **rhel9**: Specify this value for hosts that use RHEL 9.

4. Change the permissions to make **ccctl** executable by running the following command:

```
$ chmod 775 ccoctl.<rhel_version>
```

Verification

- To verify that **ccocctl** is ready to use, display the help file. Use a relative file name when you run the command, for example:

```
$ ./ccocctl.rhel9
```

Example output

OpenShift credentials provisioning tool

Usage:

```
ccocctl [command]
```

Available Commands:

aws	Manage credentials objects for AWS cloud
azure	Manage credentials objects for Azure
gcp	Manage credentials objects for Google cloud
help	Help about any command
ibmcloud	Manage credentials objects for {ibm-cloud-title}
nutanix	Manage credentials objects for Nutanix

Flags:

```
-h, --help help for ccocctl
```

Use "ccocctl [command] --help" for more information about a command.

9.9.2.2. Creating GCP resources with the Cloud Credential Operator utility

You can use the **ccocctl gcp create-all** command to automate the creation of GCP resources.



NOTE

By default, **ccocctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccocctl_output_dir>** to refer to this directory.

Prerequisites

You must have:

- Extracted and prepared the **ccocctl** binary.

Procedure

- Set a **\$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
--from=$RELEASE_IMAGE \
--credentials-requests \
--included ① \
--install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \② \
--to=<path_to_directory_for_credentials_requests> ③
```

- ① The **--included** parameter includes only the manifests that your specific cluster configuration requires.
- ② Specify the location of the **install-config.yaml** file.
- ③ Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.



NOTE

This command might take a few moments to run.

3. Use the **ccctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccctl gcp create-all \
--name=<name> \① \
--region=<gcp_region> \② \
--project=<gcp_project_id> \③ \
--credentials-requests-dir=<path_to_credentials_requests_directory> \④
```

- ① Specify the user-defined name for all created GCP resources used for tracking. If you plan to install the GCP Filestore Container Storage Interface (CSI) Driver Operator, retain this value.
- ② Specify the GCP region in which cloud resources will be created.
- ③ Specify the GCP project ID in which cloud resources will be created.
- ④ Specify the directory containing the files of **CredentialsRequest** manifests to create GCP service accounts.



NOTE

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccctl_output_dir>/manifests** directory:

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

Example output

```
cluster-authentication-02-config.yaml
openshift-cloud-controller-manager-gcp-ccm-cloud-credentials-credentials.yaml
openshift-cloud-credential-operator-cloud-credential-operator-gcp-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capg-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-gcp-pd-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-gcp-cloud-credentials-credentials.yaml
```

You can verify that the IAM service accounts are created by querying GCP. For more information, refer to GCP documentation on listing IAM service accounts.

9.9.2.3. Incorporating the Cloud Credential Operator utility manifests

To implement short-term security credentials managed outside the cluster for individual components, you must move the manifest files that the Cloud Credential Operator utility (**ccoctl**) created to the correct directories for the installation program.

Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have configured the Cloud Credential Operator utility (**ccoctl**).
- You have created the cloud provider resources that are required for your cluster with the **ccoctl** utility.

Procedure

1. Add the following granular permissions to the GCP account that the installation program uses:

Example 9.4. Required GCP permissions

- compute.machineTypes.list
- compute.regions.list
- compute.zones.list
- dns.changes.create
- dns.changes.get
- dns.managedZones.create
- dns.managedZones.delete
- dns.managedZones.get
- dns.managedZones.list

- dns.networks.bindPrivateDNSZone
- dns.resourceRecordSets.create
- dns.resourceRecordSets.delete
- dns.resourceRecordSets.list

2. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

Sample configuration file snippet

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

3. If you have not previously created installation manifest files, do so by running the following command:

```
$ openshift-install create manifests --dir <installation_directory>
```

where **<installation_directory>** is the directory in which the installation program creates files.

4. Copy the manifests that the **ccctl** utility generated to the **manifests** directory that the installation program created by running the following command:

```
$ cp /<path_to_ccctl_output_dir>/manifests/* ./manifests/
```

5. Copy the **tls** directory that contains the private key to the installation directory:

```
$ cp -a /<path_to_ccctl_output_dir>/tls .
```

9.10. DEPLOYING THE CLUSTER

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

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

- You have verified that the cloud provider account on your host has the correct permissions to deploy the cluster. An account with incorrect permissions causes the installation process to fail with an error message that displays the missing permissions.

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. Change to the directory that contains the installation program and initialize the cluster deployment:


```
$ ./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**.
3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.
 - If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
 - If you included the **Service Account Key Admin** role, you can remove it.

Verification

When the cluster deployment completes successfully:

- The terminal displays directions for accessing your cluster, including a link to the web console and credentials for the **kubeadmin** user.
- Credential information also outputs to `<installation_directory>/openshift_install.log`.



IMPORTANT

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

Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
```

INFO Access the OpenShift web-console here: <https://console-openshift-console.apps.mycluster.example.com>
 INFO Login to the console with user: "kubeadmin", and password: "password"
 INFO Time elapsed: 36m22s



IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

9.11. LOGGING IN TO THE CLUSTER BY USING THE CLI

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

- You deployed an OpenShift Container Platform cluster.
- You installed 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
```

Example output

```
system:admin
```

Additional resources

- See [Accessing the web console](#) for more details about accessing and understanding the OpenShift Container Platform web console.

9.12. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.18, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to [OpenShift Cluster Manager](#).

After you confirm that your [OpenShift Cluster Manager](#) inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See [About remote health monitoring](#) for more information about the Telemetry service

9.13. NEXT STEPS

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

CHAPTER 10. INSTALLING A CLUSTER ON USER-PROVISIONED INFRASTRUCTURE IN GCP BY USING DEPLOYMENT MANAGER TEMPLATES

In OpenShift Container Platform version 4.18, you can install a cluster on Google Cloud Platform (GCP) that uses 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.

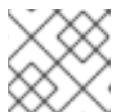


IMPORTANT

The steps for performing a user-provisioned infrastructure installation are provided as an example only. Installing a cluster with infrastructure you provide requires knowledge of the cloud provider and the installation process of OpenShift Container Platform. 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.

10.1. PREREQUISITES

- You reviewed details about the [OpenShift Container Platform installation and update processes](#).
- You read the documentation on [selecting a cluster installation method and preparing it for users](#).
- If you use a firewall and plan to use the Telemetry service, you [configured the firewall to allow the sites](#) that your cluster requires access to.
- If the cloud identity and access management (IAM) APIs are not accessible in your environment, or if you do not want to store an administrator-level credential secret in the **kube-system** namespace, you can [manually create and maintain long-term credentials](#).



NOTE

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

10.2. 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.

10.3. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM

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

You must have internet access to:

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



IMPORTANT

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

10.4. CONFIGURING YOUR GCP PROJECT

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

10.4.1. 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.



IMPORTANT

Your GCP project must use the Premium Network Service Tier if you are using installer-provisioned infrastructure. The Standard Network Service Tier is not supported for clusters installed using the installation program. The installation program configures internal load balancing for the **api-int.<cluster_name>. <base_domain>** URL; the Premium Tier is required for internal load balancing.

10.4.2. 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. You may also enable optional API services which are not required for installation. See [Enabling services](#) in the GCP documentation.

Table 10.1. Required API services

API service	Console service name
Compute Engine API	compute.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 Usage API	serviceusage.googleapis.com

Table 10.2. Optional API services

API service	Console service name
Cloud Deployment Manager V2 API	deploymentmanager.googleapis.com
Google Cloud APIs	clouddapis.googleapis.com
Service Management API	servicemanagement.googleapis.com
Google Cloud Storage JSON API	storage-api.googleapis.com
Cloud Storage	storage-component.googleapis.com

10.4.3. 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

- 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 migrated your root domain to Google Cloud DNS, migrate your DNS records. See [Migrating to Cloud DNS](#) in the GCP documentation.
6. If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain. This process might include a request to your company's IT department or the division that controls the root domain and DNS services for your company.

10.4.4. 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 10.3. GCP resources used in a default cluster

Service	Component	Location	Total resources required	Resources removed after bootstrap
Service account	IAM	Global	6	1
Firewall rules	Networking	Global	11	1
Forwarding rules	Compute	Global	2	0
Health checks	Compute	Global	2	0
Images	Compute	Global	1	0
Networks	Networking	Global	1	0

Service	Component	Location	Total resources required	Resources removed after bootstrap
Routers	Networking	Global	1	0
Routes	Networking	Global	2	0
Subnetworks	Compute	Global	2	0
Target pools	Networking	Global	2	0



NOTE

If any of the quotas are insufficient during installation, the installation program displays an error that states both which quota was exceeded and the region.

Be sure to consider your actual cluster size, planned cluster growth, and any usage from other clusters that are associated with your account. The CPU, static IP addresses, and persistent disk SSD (storage) quotas are the ones that are most likely to be insufficient.

If you plan to deploy your cluster in one of the following regions, you will exceed the maximum storage quota and are likely to exceed the CPU quota limit:

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europe-north1**
- **europe-west2**
- **europe-west3**
- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

You can increase resource quotas from the [GCP console](#), but you might need to file a support ticket. Be sure to plan your cluster size early so that you can allow time to resolve the support ticket before you install your OpenShift Container Platform cluster.

10.4.5. Creating a service account in GCP

OpenShift Container Platform requires a Google Cloud Platform (GCP) service account that provides authentication and authorization to access data in the Google APIs. If you do not have an existing IAM service account that contains the required roles in your project, you must create one.

Prerequisites

- You created a project to host your cluster.

Procedure

1. Create a 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](#).



NOTE

While making the service account an owner of the project is the easiest way to gain the required permissions, it means that service account has complete control over the project. You must determine if the risk that comes from offering that power is acceptable.

3. You can create the service account key in JSON format, or attach the service account to a GCP virtual machine. See [Creating service account keys](#) and [Creating and enabling service accounts for instances](#) in the GCP documentation.



NOTE

If you use a virtual machine with an attached service account to create your cluster, you must set **credentialsMode: Manual** in the **install-config.yaml** file before installation.

10.4.6. Required GCP roles

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. If your organization's security policies require a more restrictive set of permissions, you can create a service account with the following permissions. If you deploy your cluster into an existing virtual private cloud (VPC), the service account does not require certain networking permissions, which are noted in the following lists:

Required roles for the installation program

- Compute Admin
- Role Administrator
- Security Admin
- Service Account Admin
- Service Account Key Admin

- Service Account User
- Storage Admin

Required roles for creating network resources during installation

- DNS Administrator

Required roles for using the Cloud Credential Operator in passthrough mode

- Compute Load Balancer Admin
- Tag User

Required roles for user-provisioned GCP infrastructure

- Deployment Manager Editor

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

Table 10.4. GCP service account roles

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
	roles/artifactregistry.reader

10.4.7. Required GCP permissions for user-provisioned infrastructure

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.

If your organization's security policies require a more restrictive set of permissions, you can create [custom roles](#) with the necessary permissions. The following permissions are required for the user-provisioned infrastructure for creating and deleting the OpenShift Container Platform cluster.

Example 10.1. Required permissions for creating network resources

- `compute.addresses.create`
- `compute.addresses.createInternal`
- `compute.addresses.delete`
- `compute.addresses.get`
- `compute.addresses.list`
- `compute.addresses.use`
- `compute.addresses.useInternal`
- `compute.firewalls.create`
- `compute.firewalls.delete`
- `compute.firewalls.get`
- `compute.firewalls.list`
- `compute.forwardingRules.create`
- `compute.forwardingRules.get`
- `compute.forwardingRules.list`
- `compute.forwardingRules.setLabels`
- `compute.globalAddresses.create`
- `compute.globalAddresses.get`
- `compute.globalAddresses.use`
- `compute.globalForwardingRules.create`
- `compute.globalForwardingRules.get`
- `compute.globalForwardingRules.setLabels`
- `compute.networks.create`
- `compute.networks.get`
- `compute.networks.list`
- `compute.networks.updatePolicy`
- `compute.networks.use`
- `compute.routers.create`
- `compute.routers.get`
- `compute.routers.list`

- `compute.routers.update`
- `compute.routes.list`
- `compute.subnetworks.create`
- `compute.subnetworks.get`
- `compute.subnetworks.list`
- `compute.subnetworks.use`
- `compute.subnetworks.useExternalIIP`

Example 10.2. Required permissions for creating load balancer resources

- `compute.backendServices.create`
- `compute.backendServices.get`
- `compute.backendServices.list`
- `compute.backendServices.update`
- `compute.backendServices.use`
- `compute.regionBackendServices.create`
- `compute.regionBackendServices.get`
- `compute.regionBackendServices.list`
- `compute.regionBackendServices.update`
- `compute.regionBackendServices.use`
- `compute.targetPools.addInstance`
- `compute.targetPools.create`
- `compute.targetPools.get`
- `compute.targetPools.list`
- `compute.targetPools.removeInstance`
- `compute.targetPools.use`
- `compute.targetTcpProxies.create`
- `compute.targetTcpProxies.get`
- `compute.targetTcpProxies.use`

Example 10.3. Required permissions for creating DNS resources

- **dns.changes.create**
- **dns.changes.get**
- **dns.managedZones.create**
- **dns.managedZones.get**
- **dns.managedZones.list**
- **dns.networks.bindPrivateDNSZone**
- **dns.resourceRecordSets.create**
- **dns.resourceRecordSets.list**
- **dns.resourceRecordSets.update**

Example 10.4. Required permissions for creating Service Account resources

- **iam.serviceAccountKeys.create**
- **iam.serviceAccountKeys.delete**
- **iam.serviceAccountKeys.get**
- **iam.serviceAccountKeys.list**
- **iam.serviceAccounts.actAs**
- **iam.serviceAccounts.create**
- **iam.serviceAccounts.delete**
- **iam.serviceAccounts.get**
- **iam.serviceAccounts.list**
- **resourcemanager.projects.get**
- **resourcemanager.projects.getIamPolicy**
- **resourcemanager.projects.setIamPolicy**

Example 10.5. Required permissions for creating compute resources

- **compute.disks.create**
- **compute.disks.get**
- **compute.disks.list**
- **compute.instanceGroups.create**
- **compute.instanceGroups.delete**

- `compute.instanceGroups.get`
- `compute.instanceGroups.list`
- `compute.instanceGroups.update`
- `compute.instanceGroups.use`
- `compute.instances.create`
- `compute.instances.delete`
- `compute.instances.get`
- `compute.instances.list`
- `compute.instances.setLabels`
- `compute.instances.setMetadata`
- `compute.instances.setServiceAccount`
- `compute.instances.setTags`
- `compute.instances.use`
- `compute.machineTypes.get`
- `compute.machineTypes.list`

Example 10.6. Required for creating storage resources

- `storage.buckets.create`
- `storage.buckets.delete`
- `storage.buckets.get`
- `storage.buckets.list`
- `storage.objects.create`
- `storage.objects.delete`
- `storage.objects.get`
- `storage.objects.list`

Example 10.7. Required permissions for creating health check resources

- `compute.healthChecks.create`
- `compute.healthChecks.get`
- `compute.healthChecks.list`

- `compute.healthChecks.useReadOnly`
- `compute.httpHealthChecks.create`
- `compute.httpHealthChecks.get`
- `compute.httpHealthChecks.list`
- `compute.httpHealthChecks.useReadOnly`
- `compute.regionHealthChecks.create`
- `compute.regionHealthChecks.get`
- `compute.regionHealthChecks.useReadOnly`

Example 10.8. Required permissions to get GCP zone and region related information

- `compute.globalOperations.get`
- `compute.regionOperations.get`
- `compute.regions.get`
- `compute.regions.list`
- `compute.zoneOperations.get`
- `compute.zones.get`
- `compute.zones.list`

Example 10.9. Required permissions for checking services and quotas

- `monitoring.timeSeries.list`
- `serviceusage.quotas.get`
- `serviceusage.services.list`

Example 10.10. Required IAM permissions for installation

- `iam.roles.get`

Example 10.11. Required permissions when authenticating without a service account key

- `iam.serviceAccounts.signBlob`

Example 10.12. Required Images permissions for installation

- `compute.images.create`
- `compute.images.delete`
- `compute.images.get`
- `compute.images.list`

Example 10.13. Optional permission for running `gather bootstrap`

- `compute.instances.getSerialPortOutput`

Example 10.14. Required permissions for deleting network resources

- `compute.addresses.delete`
- `compute.addresses.deleteInternal`
- `compute.addresses.list`
- `compute.addresses.setLabels`
- `compute.firewalls.delete`
- `compute.firewalls.list`
- `compute.forwardingRules.delete`
- `compute.forwardingRules.list`
- `compute.globalAddresses.delete`
- `compute.globalAddresses.list`
- `compute.globalForwardingRules.delete`
- `compute.globalForwardingRules.list`
- `compute.networks.delete`
- `compute.networks.list`
- `compute.networks.updatePolicy`
- `compute.routers.delete`
- `compute.routers.list`
- `compute.routes.list`
- `compute.subnetworks.delete`
- `compute.subnetworks.list`

Example 10.15. Required permissions for deleting load balancer resources

- **compute.backendServices.delete**
- **compute.backendServices.list**
- **compute.regionBackendServices.delete**
- **compute.regionBackendServices.list**
- **compute.targetPools.delete**
- **compute.targetPools.list**
- **compute.targetTcpProxies.delete**
- **compute.targetTcpProxies.list**

Example 10.16. Required permissions for deleting DNS resources

- **dns.changes.create**
- **dns.managedZones.delete**
- **dns.managedZones.get**
- **dns.managedZones.list**
- **dns.resourceRecordSets.delete**
- **dns.resourceRecordSets.list**

Example 10.17. Required permissions for deleting Service Account resources

- **iam.serviceAccounts.delete**
- **iam.serviceAccounts.get**
- **iam.serviceAccounts.list**
- **resourcemanager.projects.getIamPolicy**
- **resourcemanager.projects.setIamPolicy**

Example 10.18. Required permissions for deleting compute resources

- **compute.disks.delete**
- **compute.disks.list**
- **compute.instanceGroups.delete**
- **compute.instanceGroups.list**

- `compute.instances.delete`
- `compute.instances.list`
- `compute.instances.stop`
- `compute.machineTypes.list`

Example 10.19. Required for deleting storage resources

- `storage.buckets.delete`
- `storage.buckets.getIamPolicy`
- `storage.buckets.list`
- `storage.objects.delete`
- `storage.objects.list`

Example 10.20. Required permissions for deleting health check resources

- `compute.healthChecks.delete`
- `compute.healthChecks.list`
- `compute.httpHealthChecks.delete`
- `compute.httpHealthChecks.list`
- `compute.regionHealthChecks.delete`
- `compute.regionHealthChecks.list`

Example 10.21. Required Images permissions for deletion

- `compute.images.delete`
- `compute.images.list`

Example 10.22. Required permissions to get Region related information

- `compute.regions.get`

Example 10.23. Required Deployment Manager permissions

- `deploymentmanager.deployments.create`
- `deploymentmanager.deployments.delete`

- `deploymentmanager.deployments.get`
- `deploymentmanager.deployments.list`
- `deploymentmanager.manifests.get`
- `deploymentmanager.operations.get`
- `deploymentmanager.resources.list`

Additional resources

- [Optimizing storage](#)

10.4.8. Supported GCP regions

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

- **africa-south1** (Johannesburg, South Africa)
- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-northeast3** (Seoul, South Korea)
- **asia-south1** (Mumbai, India)
- **asia-south2** (Delhi, India)
- **asia-southeast1** (Jurong West, Singapore)
- **asia-southeast2** (Jakarta, Indonesia)
- **australia-southeast1** (Sydney, Australia)
- **australia-southeast2** (Melbourne, Australia)
- **europe-central2** (Warsaw, Poland)
- **europe-north1** (Hamina, Finland)
- **europe-southwest1** (Madrid, Spain)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)

- **europe-west6** (Zürich, Switzerland)
- **europe-west8** (Milan, Italy)
- **europe-west9** (Paris, France)
- **europe-west12** (Turin, Italy)
- **me-central1** (Doha, Qatar, Middle East)
- **me-central2** (Dammam, Saudi Arabia, Middle East)
- **me-west1** (Tel Aviv, Israel)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **northamerica-northeast2** (Toronto, Ontario, Canada)
- **southamerica-east1** (São Paulo, Brazil)
- **southamerica-west1** (Santiago, Chile)
- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-east5** (Columbus, Ohio)
- **us-south1** (Dallas, Texas)
- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)
- **us-west3** (Salt Lake City, Utah, USA)
- **us-west4** (Las Vegas, Nevada, USA)



NOTE

To determine which machine type instances are available by region and zone, see the Google [documentation](#).

10.4.9. 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.
See [Authorizing with a service account](#) in the GCP documentation.

10.5. REQUIREMENTS FOR A CLUSTER WITH USER-PROVISIONED INFRASTRUCTURE

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

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

10.5.1. Required machines for cluster installation

The smallest OpenShift Container Platform clusters require the following hosts:

Table 10.5. Minimum required hosts

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



IMPORTANT

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

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

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

10.5.2. Minimum resource requirements for cluster installation

Each cluster machine must meet the following minimum requirements:

Table 10.6. Minimum resource requirements

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

- One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.
- OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.
- As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.



NOTE

For OpenShift Container Platform version 4.18, RHCOS is based on RHEL version 9.4, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see [Architectures](#) (RHEL documentation).

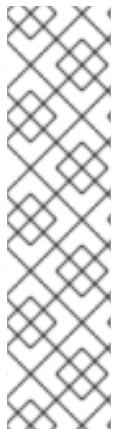
If an instance type for your platform meets the minimum requirements for cluster machines, it is supported to use in OpenShift Container Platform.

Additional resources

- Optimizing storage

10.5.3. Tested instance types for GCP

The following Google Cloud Platform instance types have been tested with OpenShift Container Platform.



NOTE

Not all instance types are available in all regions and zones. For a detailed breakdown of which instance types are available in which zones, see [regions and zones](#) (Google documentation).

Some instance types require the use of Hyperdisk storage. If you use an instance type that requires Hyperdisk storage, all of the nodes in your cluster must support Hyperdisk storage, and you must change the default storage class to use Hyperdisk storage. For more information, see [machine series support for Hyperdisk](#) (Google documentation). For instructions on modifying storage classes, see the "GCE PersistentDisk (gcePD) object definition" section in the Dynamic Provisioning page in *Storage*.

Example 10.24. Machine series

- A2
- A3
- C2
- C2D
- C3
- C3D
- C4
- E2
- M1
- N1
- N2
- N2D
- N4
- Tau T2D

10.5.4. Tested instance types for GCP on 64-bit ARM infrastructures

The following Google Cloud Platform (GCP) 64-bit ARM instance types have been tested with OpenShift Container Platform.

Example 10.25. Machine series for 64-bit ARM machines

- **C4A**
- **Tau T2A**

10.5.5. Using custom machine types

Using a custom machine type to install a OpenShift Container Platform cluster is supported.

Consider the following when using a custom machine type:

- Similar to predefined instance types, custom machine types must meet the minimum resource requirements for control plane and compute machines. For more information, see "Minimum resource requirements for cluster installation".
- The name of the custom machine type must adhere to the following syntax:
custom-<number_of_cpus>-<amount_of_memory_in_mb>

For example, **custom-6-20480**.

10.6. 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. You also have the option to first set up a separate **var** partition during the preparation phases of installation.

10.6.1. Optional: Creating a separate /var partition

It is recommended that disk partitioning for OpenShift Container Platform be left to the installer. However, there are cases where you might want to create separate partitions in a part of the filesystem that you expect to grow.

OpenShift Container Platform supports the addition of a single partition to attach storage to either the **/var** partition or a subdirectory of **/var**. For example:

- **/var/lib/containers**: Holds container-related content that can grow as more images and containers are added to a system.
- **/var/lib/etcd**: Holds data that you might want to keep separate for purposes such as performance optimization of etcd storage.
- **/var**: Holds data that you might want to keep separate for purposes such as auditing.

Storing the contents of a **/var** directory separately makes it easier to grow storage for those areas as needed and reinstall OpenShift Container Platform at a later date and keep that data intact. With this method, you will not have to pull all your containers again, nor will you have to copy massive log files when you update systems.

Because **/var** must be in place before a fresh installation of Red Hat Enterprise Linux CoreOS (RHCOS), the following procedure sets up the separate **/var** partition by creating a machine config manifest that is inserted during the **openshift-install** preparation phases of an OpenShift Container Platform

installation.



IMPORTANT

If you follow the steps to create a separate **/var** partition in this procedure, it is not necessary to create the Kubernetes manifest and Ignition config files again as described later in this section.

Procedure

1. Create a directory to hold the OpenShift Container Platform installation files:

```
$ mkdir $HOME/clusterconfig
```

2. Run **openshift-install** to create a set of files in the **manifest** and **openshift** subdirectories. Answer the system questions as you are prompted:

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

Example output

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. Optional: Confirm that the installation program created manifests in the **clusterconfig/openshift** directory:

```
$ ls $HOME/clusterconfig/openshift/
```

Example output

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
...
```

4. Create a Butane config that configures the additional partition. For example, name the file **\$HOME/clusterconfig/98-var-partition.bu**, change the disk device name to the name of the storage device on the **worker** systems, and set the storage size as appropriate. This example places the **/var** directory on a separate partition:

```
variant: openshift
version: 4.18.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
```

```

- device: /dev/disk/by-id/<device_name> ①
  partitions:
    - label: var
      start_mib: <partition_start_offset> ②
      size_mib: <partition_size> ③
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] ④
      with_mount_unit: true

```

- ① The storage device name of the disk that you want to partition.
- ② When adding a data partition to the boot disk, a minimum value of 25000 MiB (Mebibytes) is recommended. The root file system is automatically resized to fill all available space up to the specified offset. If no value is specified, or if the specified value is smaller than the recommended minimum, the resulting root file system will be too small, and future reinstalls of RHCOS might overwrite the beginning of the data partition.
- ③ The size of the data partition in mebibytes.
- ④ The **prjquota** mount option must be enabled for filesystems used for container storage.



NOTE

When creating a separate **/var** partition, you cannot use different instance types for worker nodes, if the different instance types do not have the same device name.

5. Create a manifest from the Butane config and save it to the **clusterconfig/openshift** directory. For example, run the following command:

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. Run **openshift-install** again to create Ignition configs from a set of files in the **manifest** and **openshift** subdirectories:

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth bootstrap.ign master.ign metadata.json worker.ign
```

Now you can use the Ignition config files as input to the installation procedures to install Red Hat Enterprise Linux CoreOS (RHCOS) systems.

10.6.2. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP).

Prerequisites

- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- Configure a GCP account.

Procedure

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

- a. Change to the directory that contains the installation program and 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.

When specifying the directory:

- a. Verify that the directory has the **execute** permission. This permission is required to run Terraform binaries under the installation directory.
- b. Use an empty directory. Some installation assets, such as bootstrap X.509 certificates, have short expiration intervals, therefore 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.

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



NOTE

If you are installing a three-node cluster, be sure to set the **compute.replicas** parameter to **0**. This ensures that the cluster's control planes are schedulable. For more information, see "Installing a three-node cluster on GCP".

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.

Additional resources

- [Installation configuration parameters for GCP](#)

10.6.3. Enabling Shielded VMs

You can use Shielded VMs when installing your cluster. Shielded VMs have extra security features including secure boot, firmware and integrity monitoring, and rootkit detection. For more information, see Google's documentation on [Shielded VMs](#).



NOTE

Shielded VMs are currently not supported on clusters with 64-bit ARM infrastructures.

Procedure

- Use a text editor to edit the **install-config.yaml** file prior to deploying your cluster and add one of the following stanzas:
 - a. To use shielded VMs for only control plane machines:


```
controlPlane:
  platform:
    gcp:
      secureBoot: Enabled
```
 - b. To use shielded VMs for only compute machines:


```
compute:
  - platform:
    gcp:
      secureBoot: Enabled
```
 - c. To use shielded VMs for all machines:


```
all:
  platform:
    gcp:
      secureBoot: Enabled
```

```

platform:
  gcp:
    defaultMachinePlatform:
      secureBoot: Enabled

```

10.6.4. Enabling Confidential VMs

You can use Confidential VMs when installing your cluster. Confidential VMs encrypt data while it is being processed. For more information, see Google's documentation on [Confidential Computing](#). You can enable Confidential VMs and Shielded VMs at the same time, although they are not dependent on each other.



NOTE

Confidential VMs are currently not supported on 64-bit ARM architectures.

Procedure

- Use a text editor to edit the **install-config.yaml** file prior to deploying your cluster and add one of the following stanzas:
- To use confidential VMs for only control plane machines:

```

controlPlane:
  platform:
    gcp:
      confidentialCompute: Enabled ①
      type: n2d-standard-8 ②
      onHostMaintenance: Terminate ③

```

- ① Enable confidential VMs.
- ② Specify a machine type that supports Confidential VMs. Confidential VMs require the N2D or C2D series of machine types. For more information on supported machine types, see [Supported operating systems and machine types](#).
- ③ Specify the behavior of the VM during a host maintenance event, such as a hardware or software update. For a machine that uses Confidential VM, this value must be set to **Terminate**, which stops the VM. Confidential VMs do not support live VM migration.

- To use confidential VMs for only compute machines:

```

compute:
- platform:
  gcp:
    confidentialCompute: Enabled
    type: n2d-standard-8
    onHostMaintenance: Terminate

```

- To use confidential VMs for all machines:

```

platform:
  gcp:

```

```
defaultMachinePlatform:
  confidentialCompute: Enabled
  type: n2d-standard-8
  onHostMaintenance: Terminate
```

10.6.5. 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

- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

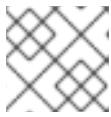
Procedure

- 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: https://<username>:<pswd>@<ip>:<port> ②
  noProxy: example.com ③
  additionalTrustBundle: | ④
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> ⑤
```

- 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.
- 3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass the proxy for all destinations.

- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates
- 5 Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.



NOTE

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



NOTE

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

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**.



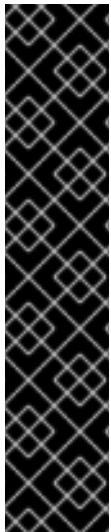
NOTE

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

10.6.6. 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 configure the machines.

The installation configuration file transforms into the Kubernetes manifests. The manifests wrap into the Ignition configuration files, which are later used to configure the cluster machines.



IMPORTANT

- The Ignition config files that the OpenShift Container Platform installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

Procedure

1. Change to the directory that contains the OpenShift Container Platform installation program and generate the Kubernetes manifests for the cluster:

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

- ① For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

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

```
$ rm -f <installation_directory>/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 control plane machine set:

```
$ rm -f <installation_directory>/openshift/99_openshift-machine-api_master-control-plane-machine-set.yaml
```

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

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-* yaml
```



IMPORTANT

If you disabled the **MachineAPI** capability when installing a cluster on user-provisioned infrastructure, you must remove the Kubernetes manifest files that define the worker machines. Otherwise, your cluster fails to install.

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

**WARNING**

If you are installing a three-node cluster, skip the following step to allow the control plane nodes to be schedulable.

**IMPORTANT**

When you configure control plane nodes from the default unschedulable to schedulable, additional subscriptions are required. This is because control plane nodes then become compute nodes.

5. Check that the **mastersSchedulable** parameter in the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes manifest file is set to **false**. This setting prevents pods from being scheduled on the control plane machines:
 - a. Open the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** file.
 - b. Locate the **mastersSchedulable** parameter and ensure that it is set to **false**.
 - c. Save and exit the file.
6. Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the **privateZone** and **publicZone** sections from the **<installation_directory>/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 this section completely.

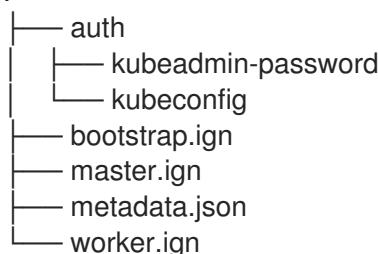
If you do so, you must add ingress DNS records manually in a later step.

7. To create the Ignition configuration files, run the following command from the directory that contains the installation program:

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

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

Ignition config files are created for the bootstrap, control plane, and compute nodes in the installation directory. The **kubeadmin-password** and **kubeconfig** files are created in the **./<installation_directory>/auth** directory:



Additional resources

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

10.7. EXPORTING COMMON VARIABLES

10.7.1. Extracting the infrastructure name

The Ignition config files contain a unique cluster identifier that you can use to uniquely identify your cluster in Google Cloud Platform (GCP). The infrastructure name is also used to locate the appropriate GCP resources during an OpenShift Container Platform installation. The provided Deployment Manager templates contain references to this infrastructure name, so you must extract it.

Prerequisites

- You installed 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
```

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

Example output

```
openshift-vw9j6 ①
```

- ① The output of this command is your cluster name and a random string.

10.7.2. 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.

Procedure

1. 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='10.0.0.0/17'  
$ export WORKER_SUBNET_CIDR='10.0.128.0/17'  
  
$ 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.

10.8. 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

- You have defined the variables in the *Exporting common variables* section.

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-central1`.
- 3** `master_subnet_cidr` is the CIDR for the master subnet, for example `10.0.0.0/17`.
- 4** `worker_subnet_cidr` is the CIDR for the worker subnet, for example `10.0.128.0/17`.

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

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

10.8.1. 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:

Example 10.26. 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': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
    'ipCidrRange': context.properties['worker_subnet_cidr']
}
}, {
    'name': context.properties['infra_id'] + '-router',
    'type': 'compute.v1.router',
    'properties': {
        'region': context.properties['region'],
        'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
        'nats': [{
            'name': context.properties['infra_id'] + '-nat-master',
            'natIpAllocateOption': 'AUTO_ONLY',
            'minPortsPerVm': 7168,
            'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
            'subnetworks': [{
                'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
                'sourceIpRangesToNat': ['ALL_IP_RANGES']
            }]
        }, {
            'name': context.properties['infra_id'] + '-nat-worker',
            'natIpAllocateOption': 'AUTO_ONLY',
            'minPortsPerVm': 512,
            'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
            'subnetworks': [{
                'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
                'sourceIpRangesToNat': ['ALL_IP_RANGES']
            }]
        }]
    }
}
]

return {'resources': resources}

```

10.9. NETWORKING REQUIREMENTS FOR USER-PROVISIONED INFRASTRUCTURE

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

10.9.1. Setting the cluster node hostnames through DHCP

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

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

10.9.2. Network connectivity requirements

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

This section provides details about the ports that are required.



IMPORTANT

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

Table 10.7. Ports used for all-machine to all-machine communications

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	1936	Metrics
	9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .
	10250-10259	The default ports that Kubernetes reserves
UDP	4789	VXLAN
	6081	Geneve
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
	500	IPsec IKE packets
	4500	IPsec NAT-T packets
	123	Network Time Protocol (NTP) on UDP port 123 If an external NTP time server is configured, you must open UDP port 123 .
	30000-32767	Kubernetes node port
ESP	N/A	IPsec Encapsulating Security Payload (ESP)

Table 10.8. Ports used for all-machine to control plane communications

Protocol	Port	Description
TCP	6443	Kubernetes API

Table 10.9. Ports used for control plane machine to control plane machine communications

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

10.10. CREATING LOAD BALANCERS IN GCP

You must configure load balancers 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

- You have defined the variables in the *Exporting common variables* section.

Procedure

1. Copy the template from the **Deployment Manager template for the internal load balancer** section of this topic and save it as **02_lb_int.py** on your computer. This template describes the internal load balancing objects that your cluster requires.
2. For an external cluster, also copy the template from the **Deployment Manager template for the external load balancer** section of this topic and save it as **02_lb_ext.py** on your computer. This template describes the external load balancing objects that your cluster requires.
3. Export the variables that the deployment template uses:
 - a. Export the cluster network location:

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

- b. Export the control plane subnet location:

```
$ export CONTROL_SUBNET=`gcloud compute networks subnets describe ${INFRA_ID}-master-subnet --region=${REGION} --format json | jq -r .selfLink`
```

- c. Export the three zones that the cluster uses:

```
$ 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`
```

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

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py ①
resources:
- name: cluster-lb-ext ②
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}' ③
    region: '${REGION}' ④
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' ⑤
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: ⑥
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF
```

① ② Required only when deploying an external cluster.

③ **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

④ **region** is the region to deploy the cluster into, for example **us-central1**.

⑤ **control_subnet** is the URI to the control subnet.

⑥ **zones** are the zones to deploy the control plane instances into, like **us-east1-b**, **us-east1-c**, and **us-east1-d**.

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

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

6. Export the cluster IP address:

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

7. For an external cluster, also export the cluster public IP address:

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

10.10.1. Deployment Manager template for the external load balancer

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

Example 10.27. 02_lb_ext.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']
            }
        },
        {
            # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
            # probe for kube-apiserver
            'name': context.properties['infra_id'] + '-api-http-health-check',
            'type': 'compute.v1.httpHealthCheck',
            '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'
            }
        }
    ]

    return {'resources': resources}
```

10.10.2. Deployment Manager template for the internal load balancer

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

Example 10.28. 02_lb_int.py Deployment Manager template

```
def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '${ref.' + context.properties['infra_id'] + '-master-' + zone + '-ig' + '.selfLink)'})
    }

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
        # probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal',
        'type': 'compute.v1.regionBackendService',
        'properties': {
            'backends': backends,
            'healthChecks': ['${ref.' + context.properties['infra_id'] + '-api-internal-health-
check.selfLink)']',
            'loadBalancingScheme': 'INTERNAL',
            'region': context.properties['region'],
            'protocol': 'TCP',
            'timeoutSec': 120
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'backendService': '${ref.' + context.properties['infra_id'] + '-api-internal.selfLink)',
            'IPAddress': '${ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)'},
            'loadBalancingScheme': 'INTERNAL',
            'ports': ['6443','22623'],
            'region': context.properties['region']
        }
    }]
}
```

```

        'subnetwork': context.properties['control_subnet']
    }
}]

for zone in context.properties['zones']:
    resources.append({
        'name': context.properties['infra_id'] + '-master-' + zone + '-ig',
        'type': 'compute.v1.instanceGroup',
        'properties': {
            'namedPorts': [
                {
                    'name': 'ignition',
                    'port': 22623
                },
                {
                    'name': 'https',
                    'port': 6443
                }
            ],
            'network': context.properties['cluster_network'],
            'zone': zone
        }
    })

return {'resources': resources}

```

You will need this template in addition to the **02_lb_ext.py** template when you create an external cluster.

10.11. CREATING A PRIVATE DNS ZONE IN GCP

You must configure a private DNS zone in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create this component 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

- Ensure you defined the variables in the *Exporting common variables* and *Creating load balancers in GCP* sections.

Procedure

1. Copy the template from the **Deployment Manager template for the private DNS** section of this topic and save it as **02_dns.py** on your computer. This template describes the private DNS objects that your cluster requires.
2. Create a **02_dns.yaml** resource definition file:

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

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' 1
    cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' 2
    cluster_network: '${CLUSTER_NETWORK}' 3
EOF
```

- 1** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 2** **cluster_domain** is the domain for the cluster, for example **openshift.example.com**.
- 3** **cluster_network** is the **selfLink** URL to the cluster network.

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

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml
```

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

a. Add the 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
```

b. For an external cluster, also add the 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 ${CLUSTER_PUBLIC_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
```

10.11.1. Deployment Manager template for the private DNS

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

Example 10.29. 02_dns.py Deployment Manager template

```

def GenerateConfig(context):

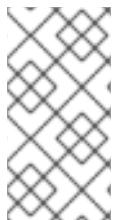
    resources = [
        {
            'name': context.properties['infra_id'] + '-private-zone',
            'type': 'dns.v1.managedZone',
            'properties': {
                'description': '',
                'dnsName': context.properties['cluster_domain'] + '',
                'visibility': 'private',
                'privateVisibilityConfig': {
                    'networks': [
                        {
                            'networkUrl': context.properties['cluster_network']
                        }
                    ]
                }
            }
        }
    ]

    return {'resources': resources}

```

10.12. CREATING FIREWALL RULES IN GCP

You must create firewall rules 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

- Ensure you defined the variables in the *Exporting common variables* and *Creating load balancers in GCP* sections.

Procedure

1. Copy the template from the **Deployment Manager template for firewall rules** section of this topic and save it as **03_firewall.py** on your computer. This template describes the security groups that your cluster requires.
2. Create a **03_firewall.yaml** resource definition file:

```

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

resources:
- name: cluster-firewall
  type: 03_firewall.py
  properties:

```

```

allowed_external_cidr: '0.0.0.0/0' 1
infra_id: '${INFRA_ID}' 2
cluster_network: '${CLUSTER_NETWORK}' 3
network_cidr: '${NETWORK_CIDR}' 4
EOF

```

- 1** **allowed_external_cidr** is the CIDR range that can access the cluster API and SSH to the bootstrap host. For an internal cluster, set this value to **\${NETWORK_CIDR}**.
- 2** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 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**.

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

```

$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml

```

10.12.1. Deployment Manager template for firewall rules

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

Example 10.30. 03_firewall.py Deployment Manager template

```

def GenerateConfig(context):
    resources = [
        {
            'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
            'type': 'compute.v1.firewall',
            'properties': {
                'network': context.properties['cluster_network'],
                'allowed': [
                    {
                        'IPProtocol': 'tcp',
                        'ports': ['22']
                    },
                    {
                        'sourceRanges': [context.properties['allowed_external_cidr']],
                        'targetTags': [context.properties['infra_id'] + '-bootstrap']
                    }
                ],
                'sourceRanges': [context.properties['allowed_external_cidr']],
                'targetTags': [context.properties['infra_id'] + '-master']
            }
        },
        {
            'name': context.properties['infra_id'] + '-api',
            'type': 'compute.v1.firewall',
            'properties': {
                'network': context.properties['cluster_network'],
                'allowed': [
                    {
                        'IPProtocol': 'tcp',
                        'ports': ['6443']
                    },
                    {
                        'sourceRanges': [context.properties['allowed_external_cidr']],
                        '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', '6443', '22624']
   }],
  'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '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']
     },
      {'IPProtocol': 'tcp',
       'ports': ['22623']
     }],
    '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'] + '-internal']
}
}
]

```

```

        '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': 'udp',
                'ports': ['500', '4500']
            },{
                'IPProtocol': 'esp',
            },{
                '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'
            ]
        }
    }
]

return {'resources': resources}

```

10.13. CREATING IAM ROLES IN GCP

You must create IAM 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

- You have defined the variables in the *Exporting common variables* section.

Procedure

1. Copy the template from the **Deployment Manager template for IAM roles** section of this topic and save it as **03_iam.py** on your computer. This template describes the IAM roles that your cluster requires.
2. Create a **03_iam.yaml** resource definition file:

```
$ cat <<EOF >03_iam.yaml
imports:
- path: 03_iam.py
resources:
- name: cluster-iam
  type: 03_iam.py
  properties:
    infra_id: '${INFRA_ID}' 1
EOF
```

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

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

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
```

4. Export the variable for the master service account:

```
$ export MASTER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

5. Export the variable for the worker service account:

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

6. Export the variable for the subnet that hosts the compute machines:

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

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

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. 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_SERVICE_ACCOUNT}
```

10.13.1. Deployment Manager template for IAM roles

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

Example 10.31. 03_iam.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [
        {
            '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}
```

10.14. 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 [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>-<arch>-gcp.<arch>.tar.gz**.

2. Create the Google storage bucket:

```
$ gsutil mb gs://<bucket_name>
```

3. Upload the RHCOS image to the Google storage bucket:

```
$ gsutil cp <downloaded_image_file_path>/rhcoss-<version>-x86_64-gcp.x86_64.tar.gz  
gs://<bucket_name>
```

4. Export the uploaded RHCOS image location as a variable:

```
$ export IMAGE_SOURCE=gs://<bucket_name>/rhcoss-<version>-x86_64-gcp.x86_64.tar.gz
```

5. Create the cluster image:

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

10.15. 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

- Ensure you defined the variables in the *Exporting common variables* and *Creating load balancers in GCP* sections.
- Ensure you installed pyOpenSSL.

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 location of the Red Hat Enterprise Linux CoreOS (RHCOS) image that the installation program requires:

```
$ export CLUSTER_IMAGE=`gcloud compute images describe ${INFRA_ID}-rhcos-image --format json | jq -r .selfLink`
```

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

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

```
$ gsutil cp <installation_directory>/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-central1**.
- 3 **zone** is the zone to deploy the bootstrap instance into, for example **us-central1-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 **root_volume_size** is the boot disk size for the bootstrap machine.
- 9 **bootstrap_ign** is the URL output when creating a signed URL.

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.

a. Add the bootstrap instance to the internal load balancer instance group:

```
$ gcloud compute instance-groups unmanaged add-instances \
${INFRA_ID}-bootstrap-ig --zone=${ZONE_0} --instances=${INFRA_ID}-bootstrap
```

b. Add the bootstrap instance group to the internal load balancer backend service:

```
$ gcloud compute backend-services add-backend \
${INFRA_ID}-api-internal --region=${REGION} --instance-group=${INFRA_ID}-
bootstrap-ig --instance-group-zone=${ZONE_0}
```

10.15.1. 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:

Example 10.32. 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',
            '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'] +
'"}}}, "version": "3.2.0"}'
        }
    ],
},
'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']
}
},
{
    'name': context.properties['infra_id'] + '-bootstrap-ig',
    'type': 'compute.v1.instanceGroup',
    'properties': {
        'namedPorts': [
            {
                'name': 'ignition',
                'port': 22623
            },
            {
                'name': 'https',
                'port': 6443
            }
        ],
        'network': context.properties['cluster_network'],
        'zone': context.properties['zone']
    }
}
]

return {'resources': resources}

```

10.16. 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

- Ensure you defined the variables in the *Exporting common variables*, *Creating load balancers in GCP*, *Creating IAM roles in GCP*, and *Creating the bootstrap machine in GCP* sections.
- 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 variable required by the resource definition:

```
$ export MASTER_IGNITION=`cat <installation_directory>/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: '${INFRA_ID}' 1
    zones: 2
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' 3
    image: '${CLUSTER_IMAGE}' 4
    machine_type: 'n1-standard-4' 5
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}' 6

    ignition: '${MASTER_IGNITION}' 7
EOF
```

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

- 2 **zones** are the zones to deploy the control plane instances into, for example **us-central1-a**, **us-central1-b**, and **us-central1-c**.
- 3 **control_subnet** is the **selfLink** URL to the control subnet.
- 4 **image** is the **selfLink** URL to the RHCOS image.
- 5 **machine_type** is the machine type of the instance, for example **n1-standard-4**.
- 6 **service_account_email** is the email address for the master service account that you created.
- 7 **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 load balancer membership due to limitations of Deployment Manager, so you must add the control plane machines manually.

- Run the following commands to add the control plane machines to the appropriate instance groups:

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_0}-ig --zone=${ZONE_0} --instances=${INFRA_ID}-master-0
```

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_1}-ig --zone=${ZONE_1} --instances=${INFRA_ID}-master-1
```

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_2}-ig --zone=${ZONE_2} --instances=${INFRA_ID}-master-2
```

- For an external cluster, you must also run the following commands to add the control plane machines to the target pools:

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_0}" --instances=${INFRA_ID}-master-0
```

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_1}" --instances=${INFRA_ID}-master-1
```

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_2}" --instances=${INFRA_ID}-master-2
```

10.16.1. 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:

Example 10.33. 05_control_plane.py Deployment Manager template

```

def GenerateConfig(context):

    resources = [
        {
            'name': context.properties['infra_id'] + '-master-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'] + '-master-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'] + '-master-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}

```

10.17. 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

- Ensure you defined the variables in the *Exporting common variables* and *Creating load balancers in GCP* sections.
- 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 backend-services remove-backend ${INFRA_ID}-api-internal --  
region=${REGION} --instance-group=${INFRA_ID}-bootstrap-ig --instance-group-  
zone=${ZONE_0}  
  
$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign  
  
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition  
  
$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

10.18. 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.



NOTE

If you are installing a three-node cluster, skip this step. A three-node cluster consists of three control plane machines, which also act as compute machines.

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

- Ensure you defined the variables in the *Exporting common variables*, *Creating load balancers in GCP*, and *Creating the bootstrap machine in GCP* sections.
- 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 variables that the resource definition uses.

- a. Export the subnet that hosts the compute machines:

```
$ export COMPUTE_SUBNET=`gcloud compute networks subnets describe
${INFRA_ID}-worker-subnet --region=${REGION} --format json | jq -r .selfLink`
```

- b. Export the email address for your service account:

```
$ export WORKER_SERVICE_ACCOUNT=`gcloud iam service-accounts list --filter
'email~^${INFRA_ID}-w@${PROJECT_NAME}.' --format json | jq -r '[0].email'`
```

- c. Export the location of the compute machine Ignition config file:

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
```

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

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

resources:
- name: 'worker-0'
```

1

```

type: 06_worker.py
properties:
  infra_id: '${INFRA_ID}' 2
  zone: '${ZONE_0}' 3
  compute_subnet: '${COMPUTE_SUBNET}' 4
  image: '${CLUSTER_IMAGE}' 5
  machine_type: 'n1-standard-4' 6
  root_volume_size: '128'
  service_account_email: '${WORKER_SERVICE_ACCOUNT}' 7
  ignition: '${WORKER_IGNITION}' 8
- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' 9
    zone: '${ZONE_1}' 10
    compute_subnet: '${COMPUTE_SUBNET}' 11
    image: '${CLUSTER_IMAGE}' 12
    machine_type: 'n1-standard-4' 13
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' 14
    ignition: '${WORKER_IGNITION}' 15
EOF

```

- 1** **name** is the name of the worker machine, for example **worker-0**.
- 2** **9** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 3** **10** **zone** is the zone to deploy the worker machine into, for example **us-central1-a**.
- 4** **11** **compute_subnet** is the **selfLink** URL to the compute subnet.
- 5** **12** **image** is the **selfLink** URL to the RHCOS image.¹
- 6** **13** **machine_type** is the machine type of the instance, for example **n1-standard-4**.
- 7** **14** **service_account_email** is the email address for the worker service account that you created.
- 8** **15** **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
```

1. To use a GCP Marketplace image, specify the offer to use:

- OpenShift Container Platform:
<https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-413-x86-64-202305021736>

- OpenShift Platform Plus: <https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-413-x86-64-202305021736>
- OpenShift Kubernetes Engine:
<https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-413-x86-64-202305021736>

10.18.1. 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:

Example 10.34. 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}
```

10.19. INSTALLING THE OPENSHIFT CLI

You can install the OpenShift CLI (**oc**) to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.18. Download and install the new version of **oc**.

Installing the OpenShift CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** drop-down list.
3. Select the appropriate version from the **Version** drop-down list.
4. Click **Download Now** next to the **OpenShift v4.18 Linux Clients** entry and save the file.
5. Unpack the archive:

```
$ tar xvf <file>
```

6. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

Verification

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

```
$ oc <command>
```

Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.18 Windows Client** entry and save the file.
4. Unzip the archive with a ZIP program.

- Move the **oc** binary to a directory that is on your **PATH**.

To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

Verification

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

```
C:\> oc <command>
```

Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

- Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
- Select the appropriate version from the **Version** drop-down list.
- Click **Download Now** next to the **OpenShift v4.18 macOS Clients** entry and save the file.



NOTE

For macOS arm64, choose the **OpenShift v4.18 macOS arm64 Client** entry.

- Unpack and unzip the archive.
- Move the **oc** binary to a directory on your PATH.

To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

Verification

- Verify your installation by using an **oc** command:

```
$ oc <command>
```

10.20. LOGGING IN TO THE CLUSTER BY USING THE CLI

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

- You installed the **oc** CLI.
- Ensure the bootstrap process completed successfully.

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
```

Example output

```
system:admin
```

10.21. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	63m	v1.31.3
master-1	Ready	master	63m	v1.31.3
master-2	Ready	master	64m	v1.31.3

The output lists all of the machines that you created.



NOTE

The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

NAME	AGE	REQUESTOR	CONDITION
csr-8b2br	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
csr-8vnps	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
...			

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 the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.



NOTE

For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- 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.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}\n{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

**NOTE**

Some Operators might not become available until some CSRs are approved.

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

NAME	AGE	REQUESTOR	CONDITION
csr-bfd72	5m26s	system:node:ip-10-0-50-126.us-east-2.compute.internal	Pending
csr-c571v	5m26s	system:node:ip-10-0-95-157.us-east-2.compute.internal	Pending
...			

- If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- 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.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}\n{{end}}{{end}}' | xargs oc adm certificate approve
```

- After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	73m	v1.31.3
master-1	Ready	master	73m	v1.31.3
master-2	Ready	master	74m	v1.31.3
worker-0	Ready	worker	11m	v1.31.3
worker-1	Ready	worker	11m	v1.31.3

**NOTE**

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- [Certificate Signing Requests](#)

10.22. 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

- Ensure you defined the variables in the *Exporting common variables* section.
- Remove the DNS Zone configuration when creating Kubernetes manifests and generating Ignition configs.
- Ensure the bootstrap process completed successfully.

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
```

Example output

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 zones:

- To use A records:

- i. Export the variable for the router IP address:

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

- ii. Add the A record to the private zones:

```
$ 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
```

- iii. For an external cluster, also add the A record to the public zones:

```
$ 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}
```

- To add explicit domains instead of using a wildcard, 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
```

Example output

```
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
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

10.23. 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

- Ensure the bootstrap process completed successfully.

Procedure

1. Complete the cluster installation:

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

Example output

```
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, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

2. Observe the running state of your cluster.

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

```
$ oc get clusterversion
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	SINCE	STATUS
version		False	True	24m	Working towards 4.5.4: 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
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED
SINCE				
authentication	4.5.4	True	False	False 7m56s
cloud-credential	4.5.4	True	False	False 31m
cluster-autoscaler	4.5.4	True	False	False 16m
console	4.5.4	True	False	False 10m
csi-snapshot-controller	4.5.4	True	False	False 16m
dns	4.5.4	True	False	False 22m
etcd	4.5.4	False	False	False 25s
image-registry	4.5.4	True	False	False 16m
ingress	4.5.4	True	False	False 16m
insights	4.5.4	True	False	False 17m
kube-apiserver	4.5.4	True	False	False 19m
kube-controller-manager	4.5.4	True	False	False 20m
kube-scheduler	4.5.4	True	False	False 20m
kube-storage-version-migrator	4.5.4	True	False	False 16m
machine-api	4.5.4	True	False	False 22m
machine-config	4.5.4	True	False	False 22m
marketplace	4.5.4	True	False	False 16m

monitoring	4.5.4	True	False	False	10m
network	4.5.4	True	False	False	23m
node-tuning	4.5.4	True	False	False	23m
openshift-apiserver	4.5.4	True	False	False	17m
openshift-controller-manager	4.5.4	True	False	False	15m
openshift-samples	4.5.4	True	False	False	16m
operator-lifecycle-manager	4.5.4	True	False	False	22m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	22m
operator-lifecycle-manager-packageserver	4.5.4	True	False	False	18m
service-ca	4.5.4	True	False	False	23m
service-catalog-apiserver	4.5.4	True	False	False	23m
service-catalog-controller-manager	4.5.4	True	False	False	23m
storage	4.5.4	True	False	False	17m

c. Run the following command to view your cluster pods:

```
$ oc get pods --all-namespaces
```

Example output

NAMESPACE	READY	STATUS	RESTARTS	AGE	NAME
kube-system					etcd-member-ip-10-0-3-111.us-east-
2.compute.internal	1/1	Running	0	35m	etcd-member-ip-10-0-3-239.us-east-
kube-system					etcd-member-ip-10-0-3-24.us-east-
2.compute.internal	1/1	Running	0	37m	openshift-apiserver-operator-6d6674f4f4-
h7t2t	1/1	Running	1	37m	apiserver-fm48r
openshift-apiserver	1/1	Running	0	30m	apiserver-fxkvv
openshift-apiserver	1/1	Running	0	29m	apiserver-q85nm
openshift-apiserver	1/1	Running	0	29m	...
openshift-service-ca-operator	1/1	Running	0	37m	openshift-service-ca-operator-66ff6dc6cd-
9r257					apiservice-cabundle-injector-695b6bcfc-cl5hm
openshift-service-ca	1/1	Running	0	35m	configmap-cabundle-injector-8498544d7-
25qn6	1/1	Running	0	35m	service-serving-cert-signer-6445fc9c6-wqdqn
openshift-service-ca	1/1	Running	0	35m	openshift-service-catalog-apiserver-
openshift-service-catalog-apiserver-operator	1/1	Running	0	32m	operator-549f44668b-b5q2w
openshift-service-catalog-controller-manager-operator	1/1	Running	0	31m	controller-manager-operator-b78cr2lnm

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

10.24. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.18, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to [OpenShift Cluster Manager](#).

After you confirm that your [OpenShift Cluster Manager](#) inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See [About remote health monitoring](#) for more information about the Telemetry service

10.25. NEXT STEPS

- [Customize your cluster](#)
- If necessary, you can [opt out of remote health reporting](#)
- [Configuring Global Access for an Ingress Controller on GCP](#)

CHAPTER 11. INSTALLING A CLUSTER INTO A SHARED VPC ON GCP USING DEPLOYMENT MANAGER TEMPLATES

In OpenShift Container Platform version 4.18, you can install a cluster into a shared Virtual Private Cloud (VPC) on Google Cloud Platform (GCP) that uses infrastructure that you provide. In this context, a cluster installed into a shared VPC is a cluster that is configured to use a VPC from a project different from where the cluster is being deployed.

A shared VPC enables an organization to connect resources from multiple projects to a common VPC network. You can communicate within the organization securely and efficiently by using internal IPs from that network. For more information about shared VPC, see [Shared VPC overview](#) in the GCP documentation.

The steps for performing a user-provided infrastructure installation into a shared VPC 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.



IMPORTANT

The steps for performing a user-provisioned infrastructure installation are provided as an example only. Installing a cluster with infrastructure you provide requires knowledge of the cloud provider and the installation process of OpenShift Container Platform. 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.

11.1. PREREQUISITES

- You reviewed details about the [OpenShift Container Platform installation and update processes](#).
- You read the documentation on [selecting a cluster installation method and preparing it for users](#).
- If you use a firewall and plan to use the Telemetry service, you [configured the firewall to allow the sites](#) that your cluster requires access to.
- If the cloud identity and access management (IAM) APIs are not accessible in your environment, or if you do not want to store an administrator-level credential secret in the **kube-system** namespace, you can [manually create and maintain long-term credentials](#).
- If you want to provide your own private hosted zone, you must have created one in the service project with the DNS pattern **cluster-name.baseDomain**, for example **testCluster.example.com**. The private hosted zone must be bound to the VPC in the host project. For more information about cross-project binding, see [Create a zone with cross-project binding](#) (Google documentation). If you do not provide a private hosted zone, the installation program will provision one automatically.



NOTE

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

11.2. 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.

11.3. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM

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

You must have internet access to:

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



IMPORTANT

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

11.4. CONFIGURING THE GCP PROJECT THAT HOSTS YOUR CLUSTER

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

11.4.1. 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.



IMPORTANT

Your GCP project must use the Premium Network Service Tier if you are using installer-provisioned infrastructure. The Standard Network Service Tier is not supported for clusters installed using the installation program. The installation program configures internal load balancing for the **api-int.<cluster_name>. <base_domain>** URL; the Premium Tier is required for internal load balancing.

11.4.2. 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. You may also enable optional API services which are not required for installation. See [Enabling services](#) in the GCP documentation.

Table 11.1. Required API services

API service	Console service name
Compute Engine API	compute.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 Usage API	serviceusage.googleapis.com

Table 11.2. Optional API services

API service	Console service name
Cloud Deployment Manager V2 API	deploymentmanager.googleapis.com
Google Cloud APIs	clouddapis.googleapis.com
Service Management API	servicemanagement.googleapis.com
Google Cloud Storage JSON API	storage-api.googleapis.com
Cloud Storage	storage-component.googleapis.com

11.4.3. 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 11.3. GCP resources used in a default cluster

Service	Component	Location	Total resources required	Resources removed after bootstrap
Service account	IAM	Global	6	1
Firewall rules	Networking	Global	11	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	2	0
Subnetworks	Compute	Global	2	0
Target pools	Networking	Global	2	0



NOTE

If any of the quotas are insufficient during installation, the installation program displays an error that states both which quota was exceeded and the region.

Be sure to consider your actual cluster size, planned cluster growth, and any usage from other clusters that are associated with your account. The CPU, static IP addresses, and persistent disk SSD (storage) quotas are the ones that are most likely to be insufficient.

If you plan to deploy your cluster in one of the following regions, you will exceed the maximum storage quota and are likely to exceed the CPU quota limit:

- **asia-east2**
- **asia-northeast2**
- **asia-south1**

- **australia-southeast1**
- **europe-north1**
- **europe-west2**
- **europe-west3**
- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

You can increase resource quotas from the [GCP console](#), but you might need to file a support ticket. Be sure to plan your cluster size early so that you can allow time to resolve the support ticket before you install your OpenShift Container Platform cluster.

11.4.4. Creating a service account in GCP

OpenShift Container Platform requires a Google Cloud Platform (GCP) service account that provides authentication and authorization to access data in the Google APIs. If you do not have an existing IAM service account that contains the required roles in your project, you must create one.

Prerequisites

- You created a project to host your cluster.

Procedure

1. Create a 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](#).



NOTE

While making the service account an owner of the project is the easiest way to gain the required permissions, it means that service account has complete control over the project. You must determine if the risk that comes from offering that power is acceptable.

3. You can create the service account key in JSON format, or attach the service account to a GCP virtual machine. See [Creating service account keys](#) and [Creating and enabling service accounts for instances](#) in the GCP documentation.



NOTE

If you use a virtual machine with an attached service account to create your cluster, you must set **credentialsMode: Manual** in the **install-config.yaml** file before installation.

11.4.4.1. Required GCP roles

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. If your organization's security policies require a more restrictive set of permissions, you can create a service account with the following permissions. If you deploy your cluster into an existing virtual private cloud (VPC), the service account does not require certain networking permissions, which are noted in the following lists:

Required roles for the installation program

- Compute Admin
- Role Administrator
- Security Admin
- Service Account Admin
- Service Account Key Admin
- Service Account User
- Storage Admin

Required roles for creating network resources during installation

- DNS Administrator

Required roles for using the Cloud Credential Operator in passthrough mode

- Compute Load Balancer Admin
- Tag User

Required roles for user-provisioned GCP infrastructure

- Deployment Manager Editor

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

Table 11.4. GCP service account roles

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

Account	Roles
	roles/iam.serviceAccountUser
Compute	roles/compute.viewer
	roles/storage.admin
	roles/artifactregistry.reader

11.4.5. Supported GCP regions

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

- **africa-south1** (Johannesburg, South Africa)
- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-northeast3** (Seoul, South Korea)
- **asia-south1** (Mumbai, India)
- **asia-south2** (Delhi, India)
- **asia-southeast1** (Jurong West, Singapore)
- **asia-southeast2** (Jakarta, Indonesia)
- **australia-southeast1** (Sydney, Australia)
- **australia-southeast2** (Melbourne, Australia)
- **europe-central2** (Warsaw, Poland)
- **europe-north1** (Hamina, Finland)
- **europe-southwest1** (Madrid, Spain)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)
- **europe-west6** (Zürich, Switzerland)

- **europe-west8** (Milan, Italy)
- **europe-west9** (Paris, France)
- **europe-west12** (Turin, Italy)
- **me-central1** (Doha, Qatar, Middle East)
- **me-central2** (Dammam, Saudi Arabia, Middle East)
- **me-west1** (Tel Aviv, Israel)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **northamerica-northeast2** (Toronto, Ontario, Canada)
- **southamerica-east1** (São Paulo, Brazil)
- **southamerica-west1** (Santiago, Chile)
- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-east5** (Columbus, Ohio)
- **us-south1** (Dallas, Texas)
- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)
- **us-west3** (Salt Lake City, Utah, USA)
- **us-west4** (Las Vegas, Nevada, USA)



NOTE

To determine which machine type instances are available by region and zone, see the Google [documentation](#).

11.4.6. 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.

See [Authorizing with a service account](#) in the GCP documentation.

11.5. REQUIREMENTS FOR A CLUSTER WITH USER-PROVISIONED INFRASTRUCTURE

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

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

11.5.1. Required machines for cluster installation

The smallest OpenShift Container Platform clusters require the following hosts:

Table 11.5. Minimum required hosts

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



IMPORTANT

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

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

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

11.5.2. Minimum resource requirements for cluster installation

Each cluster machine must meet the following minimum requirements:

Table 11.6. Minimum resource requirements

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

1. One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.
2. OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.
3. As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.



NOTE

For OpenShift Container Platform version 4.18, RHCOS is based on RHEL version 9.4, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see [Architectures](#) (RHEL documentation).

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

Additional resources

- [Optimizing storage](#)

11.5.3. Tested instance types for GCP

The following Google Cloud Platform instance types have been tested with OpenShift Container Platform.



NOTE

Not all instance types are available in all regions and zones. For a detailed breakdown of which instance types are available in which zones, see [regions and zones](#) (Google documentation).

Some instance types require the use of Hyperdisk storage. If you use an instance type that requires Hyperdisk storage, all of the nodes in your cluster must support Hyperdisk storage, and you must change the default storage class to use Hyperdisk storage. For more information, see [machine series support for Hyperdisk](#) (Google documentation). For instructions on modifying storage classes, see the "GCE PersistentDisk (gcePD) object definition" section in the Dynamic Provisioning page in *Storage*.

Example 11.1. Machine series

- A2
- A3
- C2
- C2D
- C3
- C3D
- C4
- E2
- M1
- N1
- N2
- N2D
- N4
- Tau T2D

11.5.4. Using custom machine types

Using a custom machine type to install a OpenShift Container Platform cluster is supported.

Consider the following when using a custom machine type:

- Similar to predefined instance types, custom machine types must meet the minimum resource requirements for control plane and compute machines. For more information, see "Minimum resource requirements for cluster installation".
- The name of the custom machine type must adhere to the following syntax:
custom-<number_of_cpus>-<amount_of_memory_in_mb>

For example, **custom-6-20480**.

11.6. CONFIGURING THE GCP PROJECT THAT HOSTS YOUR SHARED VPC NETWORK

If you use a shared Virtual Private Cloud (VPC) to host your OpenShift Container Platform cluster in Google Cloud Platform (GCP), you must configure the project that hosts it.



NOTE

If you already have a project that hosts the shared VPC network, review this section to ensure that the project meets all of the requirements to install an OpenShift Container Platform cluster.

Procedure

1. Create a project to host the shared VPC for your OpenShift Container Platform cluster. See [Creating and Managing Projects](#) in the GCP documentation.
2. Create a service account in the project that hosts your shared VPC. See [Creating a service account](#) in the GCP documentation.
3. 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](#).



NOTE

While making the service account an owner of the project is the easiest way to gain the required permissions, it means that service account has complete control over the project. You must determine if the risk that comes from offering that power is acceptable.

The service account for the project that hosts the shared VPC network requires the following roles:

- Compute Network User
- Compute Security Admin
- Deployment Manager Editor
- DNS Administrator
- Security Admin
- Network Management Admin

11.6.1. 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 project that hosts the shared VPC that you install the cluster into. 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 migrated your root domain to Google Cloud DNS, migrate your DNS records. See [Migrating to Cloud DNS](#) in the GCP documentation.
6. If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain. This process might include a request to your company's IT department or the division that controls the root domain and DNS services for your company.

11.6.2. 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

- You have defined the variables in the *Exporting common variables* section.

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. Export the following variables required by the resource definition:

a. Export the control plane CIDR:

```
$ export MASTER_SUBNET_CIDR='10.0.0.0/17'
```

b. Export the compute CIDR:

```
$ export WORKER_SUBNET_CIDR='10.0.128.0/17'
```

c. Export the region to deploy the VPC network and cluster to:

```
$ export REGION='<region>'
```

3. Export the variable for the ID of the project that hosts the shared VPC:

```
$ export HOST_PROJECT=<host_project>
```

4. Export the variable for the email of the service account that belongs to host project:

```
$ export HOST_PROJECT_ACCOUNT=<host_service_account_email>
```

5. 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: '<prefix>' 1
  region: '${REGION}' 2
  master_subnet_cidr: '${MASTER_SUBNET_CIDR}' 3
  worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' 4
EOF
```

1 **infra_id** is the prefix of the network name.

2 **region** is the region to deploy the cluster into, for example **us-central1**.

3 **master_subnet_cidr** is the CIDR for the master subnet, for example **10.0.0.0/17**.

4 **worker_subnet_cidr** is the CIDR for the worker subnet, for example **10.0.128.0/17**.

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

```
$ gcloud deployment-manager deployments create <vpc_deployment_name> --config 01_vpc.yaml --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} 1
```

- 1 For <vpc_deployment_name>, specify the name of the VPC to deploy.

7. Export the VPC variable that other components require:

- a. Export the name of the host project network:

```
$ export HOST_PROJECT_NETWORK=<vpc_network>
```

- b. Export the name of the host project control plane subnet:

```
$ export HOST_PROJECT_CONTROL_SUBNET=<control_plane_subnet>
```

- c. Export the name of the host project compute subnet:

```
$ export HOST_PROJECT_COMPUTE_SUBNET=<compute_subnet>
```

8. Set up the shared VPC. See [Setting up Shared VPC](#) in the GCP documentation.

11.6.2.1. 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:

Example 11.2. 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'] + '-router',
  'type': 'compute.v1.router',
  'properties': {
    'region': context.properties['region'],
    'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
    'nats': [
      {
        'name': context.properties['infra_id'] + '-nat-master',
        'natIpAllocateOption': 'AUTO_ONLY',
        'minPortsPerVm': 7168,
        'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
        'subnetworks': [
          {
            'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
            'sourceIpRangesToNat': ['ALL_IP_RANGES']
          }
        ]
      },
      {
        'name': context.properties['infra_id'] + '-nat-worker',
        'natIpAllocateOption': 'AUTO_ONLY',
        'minPortsPerVm': 512,
        'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
        'subnetworks': [
          {
            'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
            'sourceIpRangesToNat': ['ALL_IP_RANGES']
          }
        ]
      }
    ]
  }
}

return {'resources': resources}

```

11.7. 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. You also have the option to first set up a separate **var** partition during the preparation phases of installation.

11.7.1. Manually creating the installation configuration file

Installing the cluster requires that you manually create the installation configuration file.

Prerequisites

- You have an SSH public key on your local machine to provide to the installation program. The key will be used for SSH authentication onto your cluster nodes for debugging and disaster recovery.
- You have obtained the OpenShift Container Platform installation program and the pull secret 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 sample **install-config.yaml** file template that is provided 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.

Additional resources

- [Installation configuration parameters for GCP](#)

11.7.2. Enabling Shielded VMs

You can use Shielded VMs when installing your cluster. Shielded VMs have extra security features including secure boot, firmware and integrity monitoring, and rootkit detection. For more information, see Google's documentation on [Shielded VMs](#).



NOTE

Shielded VMs are currently not supported on clusters with 64-bit ARM infrastructures.

Procedure

- Use a text editor to edit the **install-config.yaml** file prior to deploying your cluster and add one of the following stanzas:
 - a. To use shielded VMs for only control plane machines:

```
controlPlane:  
  platform:  
    gcp:  
      secureBoot: Enabled
```

- b. To use shielded VMs for only compute machines:

```
compute:
- platform:
  gcp:
    secureBoot: Enabled
```

- c. To use shielded VMs for all machines:

```
platform:
gcp:
  defaultMachinePlatform:
    secureBoot: Enabled
```

11.7.3. Enabling Confidential VMs

You can use Confidential VMs when installing your cluster. Confidential VMs encrypt data while it is being processed. For more information, see Google's documentation on [Confidential Computing](#). You can enable Confidential VMs and Shielded VMs at the same time, although they are not dependent on each other.



NOTE

Confidential VMs are currently not supported on 64-bit ARM architectures.

Procedure

- Use a text editor to edit the **install-config.yaml** file prior to deploying your cluster and add one of the following stanzas:

- a. To use confidential VMs for only control plane machines:

```
controlPlane:
platform:
gcp:
  confidentialCompute: Enabled ①
  type: n2d-standard-8 ②
  onHostMaintenance: Terminate ③
```

- ① Enable confidential VMs.
- ② Specify a machine type that supports Confidential VMs. Confidential VMs require the N2D or C2D series of machine types. For more information on supported machine types, see [Supported operating systems and machine types](#).
- ③ Specify the behavior of the VM during a host maintenance event, such as a hardware or software update. For a machine that uses Confidential VM, this value must be set to **Terminate**, which stops the VM. Confidential VMs do not support live VM migration.

- b. To use confidential VMs for only compute machines:

```
compute:
- platform:
```

```

gcp:
  confidentialCompute: Enabled
  type: n2d-standard-8
  onHostMaintenance: Terminate

```

- c. To use confidential VMs for all machines:

```

platform:
  gcp:
    defaultMachinePlatform:
      confidentialCompute: Enabled
      type: n2d-standard-8
      onHostMaintenance: Terminate

```

11.7.4. 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:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
      tags: 5
        - control-plane-tag1
        - control-plane-tag2
  replicas: 3
compute: 6
  - hyperthreading: Enabled 7
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
      tags: 8
        - compute-tag1
        - compute-tag2
  replicas: 0
metadata:

```

```

name: test-cluster
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  machineNetwork:
    - cidr: 10.0.0.0/16
  networkType: OVNKubernetes 9
  serviceNetwork:
    - 172.30.0.0/16
platform:
  gcp:
    defaultMachinePlatform:
      tags: 10
        - global-tag1
        - global-tag2
      projectId: openshift-production 11
      region: us-central1 12
    pullSecret: '{"auths": ...}'
    fips: false 13
    sshKey: ssh-ed25519 AAAA... 14
    publish: Internal 15

```

1 Specify the public DNS on the host project.

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

5 **8** **10** Optional: A set of network tags to apply to the control plane or compute machine sets. The **platform.gcp.defaultMachinePlatform.tags** parameter applies to both control plane and compute machines. If the **compute.platform.gcp.tags** or **controlPlane.platform.gcp.tags** parameters are set, they override the **platform.gcp.defaultMachinePlatform.tags** parameter.

9 The cluster network plugin to install. The default value **OVNKubernetes** is the only supported value.

- 11 Specify the main project where the VM instances reside.
- 12 Specify the region that your VPC network is in.
- 13 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.



IMPORTANT

To enable FIPS mode for your cluster, you must run the installation program from a Red Hat Enterprise Linux (RHEL) computer configured to operate in FIPS mode. For more information about configuring FIPS mode on RHEL, see [Switching RHEL to FIPS mode](#).

When running Red Hat Enterprise Linux (RHEL) or Red Hat Enterprise Linux CoreOS (RHCOS) booted in FIPS mode, OpenShift Container Platform core components use the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the x86_64, ppc64le, and s390x architectures.

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

- 15 How to publish the user-facing endpoints of your cluster. Set **publish** to **Internal** to deploy a private cluster, which cannot be accessed from the internet. The default value is **External**. To use a shared VPC in a cluster that uses infrastructure that you provision, you must set **publish** to **Internal**. The installation program will no longer be able to access the public DNS zone for the base domain in the host project.

11.7.5. 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

- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (169.254.169.254).

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: https://<username>:<pswd>@<ip>:<port> ②
  noProxy: example.com ③
  additionalTrustBundle: | ④
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> ⑤
```

- 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.
- 3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with . to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use * to bypass the proxy for all destinations.
- 4 If provided, the installation program generates a config map 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** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **trustedCA** field of the **Proxy** object. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.
- 5 Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.



NOTE

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

**NOTE**

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

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.

11.7.6. 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 configure the machines.

The installation configuration file transforms into the Kubernetes manifests. The manifests wrap into the Ignition configuration files, which are later used to configure the cluster machines.

**IMPORTANT**

- The Ignition config files that the OpenShift Container Platform installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

Procedure

1. Change to the directory that contains the OpenShift Container Platform installation program and generate the Kubernetes manifests for the cluster:

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

- 1 For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

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

```
$ rm -f <installation_directory>/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 control plane machine set:

```
$ rm -f <installation_directory>/openshift/99_openshift-machine-api_master-control-plane-machine-set.yaml
```

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

```
$ rm -f <installation_directory>/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.

5. Check that the **mastersSchedulable** parameter in the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes manifest file is set to **false**. This setting prevents pods from being scheduled on the control plane machines:

- a. Open the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** file.
- b. Locate the **mastersSchedulable** parameter and ensure that it is set to **false**.
- c. Save and exit the file.

6. Remove the **privateZone** sections from the **<installation_directory>/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
  status: {}
```

- 1 Remove this section completely.

7. Configure the cloud provider for your VPC.

- a. Open the **<installation_directory>/manifests/cloud-provider-config.yaml** file.
- b. Add the **network-project-id** parameter and set its value to the ID of project that hosts the shared VPC network.
- c. Add the **network-name** parameter and set its value to the name of the shared VPC network that hosts the OpenShift Container Platform cluster.

- d. Replace the value of the **subnetwork-name** parameter with the value of the shared VPC subnet that hosts your compute machines.

The contents of the `<installation_directory>/manifests/cloud-provider-config.yaml` resemble the following example:

```
config: |+
[global]
project-id = example-project
regional = true
multizone = true
node-tags = opensh-ptzzx-master
node-tags = opensh-ptzzx-worker
node-instance-prefix = opensh-ptzzx
external-instance-groups-prefix = opensh-ptzzx
network-project-id = example-shared-vpc
network-name = example-network
subnetwork-name = example-worker-subnet
```

8. If you deploy a cluster that is not on a private network, open the `<installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml` file and replace the value of the **scope** parameter with **External**. The contents of the file resemble the following example:

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  creationTimestamp: null
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      scope: External
      type: LoadBalancerService
  status:
    availableReplicas: 0
    domain: ""
    selector: ""
```

9. To create the Ignition configuration files, run the following command from the directory that contains the installation program:

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

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

Ignition config files are created for the bootstrap, control plane, and compute nodes in the installation directory. The **kubeadmin-password** and **kubeconfig** files are created in the `./<installation_directory>/auth` directory:

```
.
  └── auth
      └── kubeadmin-password
```

```

    └── kubeconfig
        ├── bootstrap.ign
        ├── master.ign
        ├── metadata.json
        └── worker.ign

```

11.8. EXPORTING COMMON VARIABLES

11.8.1. Extracting the infrastructure name

The Ignition config files contain a unique cluster identifier that you can use to uniquely identify your cluster in Google Cloud Platform (GCP). The infrastructure name is also used to locate the appropriate GCP resources during an OpenShift Container Platform installation. The provided Deployment Manager templates contain references to this infrastructure name, so you must extract it.

Prerequisites

- You installed 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
```

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

Example output

```
openshift-vw9j6 ①
```

- ① The output of this command is your cluster name and a random string.

11.8.2. 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.

Procedure

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

```
$ export BASE_DOMAIN='<base_domain>' ①
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>' ②
$ export NETWORK_CIDR='10.0.0.0/16'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig ③
$ 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`
```

① ② Supply the values for the host project.

③ For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

11.9. NETWORKING REQUIREMENTS FOR USER-PROVISIONED INFRASTRUCTURE

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

11.9.1. Setting the cluster node hostnames through DHCP

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

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

11.9.2. Network connectivity requirements

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

This section provides details about the ports that are required.



IMPORTANT

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

Table 11.7. Ports used for all-machine to all-machine communications

Protocol	Port	Description
ICMP	N/A	Network reachability tests

Protocol	Port	Description
TCP	1936	Metrics
	9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .
	10250-10259	The default ports that Kubernetes reserves
UDP	4789	VXLAN
	6081	Geneve
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
	500	IPsec IKE packets
	4500	IPsec NAT-T packets
	123	Network Time Protocol (NTP) on UDP port 123 If an external NTP time server is configured, you must open UDP port 123 .
TCP/UDP	30000-32767	Kubernetes node port
ESP	N/A	IPsec Encapsulating Security Payload (ESP)

Table 11.8. Ports used for all-machine to control plane communications

Protocol	Port	Description
TCP	6443	Kubernetes API

Table 11.9. Ports used for control plane machine to control plane machine communications

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

11.10. CREATING LOAD BALANCERS IN GCP

You must configure load balancers 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

- You have defined the variables in the *Exporting common variables* section.

Procedure

1. Copy the template from the **Deployment Manager template for the internal load balancer** section of this topic and save it as **02_lb_int.py** on your computer. This template describes the internal load balancing objects that your cluster requires.
2. For an external cluster, also copy the template from the **Deployment Manager template for the external load balancer** section of this topic and save it as **02_lb_ext.py** on your computer. This template describes the external load balancing objects that your cluster requires.
3. Export the variables that the deployment template uses:

- a. Export the cluster network location:

```
$ export CLUSTER_NETWORK=`gcloud compute networks describe
${HOST_PROJECT_NETWORK} --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT} --format json | jq -r .selfLink`
```

- b. Export the control plane subnet location:

```
$ export CONTROL_SUBNET=`gcloud compute networks subnets describe
${HOST_PROJECT_CONTROL_SUBNET} --region=${REGION} --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} --format json | jq -r
.selfLink`
```

- c. Export the three zones that the cluster uses:

```
$ 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`
```

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

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py ①
resources:
```

```

- name: cluster-lb-ext 2
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}' 3
    region: '${REGION}' 4
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' 5
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: 6
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF

```

1 **2** Required only when deploying an external cluster.

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

4 **region** is the region to deploy the cluster into, for example **us-central1**.

5 **control_subnet** is the URI to the control subnet.

6 **zones** are the zones to deploy the control plane instances into, like **us-east1-b**, **us-east1-c**, and **us-east1-d**.

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

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

6. Export the cluster IP address:

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

7. For an external cluster, also export the cluster public IP address:

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

11.10.1. Deployment Manager template for the external load balancer

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

Example 11.3. 02_lb_ext.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']
        }
    },
    # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
    # probe for kube-apiserver
    {
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        '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'
        }
    }
]

return {'resources': resources}

```

11.10.2. Deployment Manager template for the internal load balancer

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

Example 11.4. 02_lb_int.py Deployment Manager template

```

def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '$(ref.' + context.properties['infra_id'] + '-master-' + zone + '-ig' + '.selfLink)'
        })

    resources = [
        {
            'name': context.properties['infra_id'] + '-cluster-ip',
            'type': 'compute.v1.address',
            'properties': {

```

```

        'addressType': 'INTERNAL',
        'region': context.properties['region'],
        'subnetwork': context.properties['control_subnet']
    }
}, {
    # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
    # probe for kube-apiserver
    'name': context.properties['infra_id'] + '-api-internal-health-check',
    'type': 'compute.v1.healthCheck',
    'properties': {
        'httpsHealthCheck': {
            'port': 6443,
            'requestPath': '/readyz'
        },
        'type': "HTTPS"
    }
}, {
    'name': context.properties['infra_id'] + '-api-internal',
    'type': 'compute.v1.regionBackendService',
    'properties': {
        'backends': backends,
        'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-internal-health-
check.selfLink)'],
        'loadBalancingScheme': 'INTERNAL',
        'region': context.properties['region'],
        'protocol': 'TCP',
        'timeoutSec': 120
    }
}, {
    'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
    'type': 'compute.v1.forwardingRule',
    'properties': {
        'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal.selfLink)',
        'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
        'loadBalancingScheme': 'INTERNAL',
        'ports': ['6443','22623'],
        'region': context.properties['region'],
        'subnetwork': context.properties['control_subnet']
    }
}
]

for zone in context.properties['zones']:
    resources.append({
        'name': context.properties['infra_id'] + '-master-' + zone + '-ig',
        'type': 'compute.v1.instanceGroup',
        'properties': {
            'namedPorts': [
                {
                    'name': 'ignition',
                    'port': 22623
                }, {
                    'name': 'https',
                    'port': 6443
                }
            ],
            'network': context.properties['cluster_network'],
        }
    })

```

```

        'zone': zone
    }
})

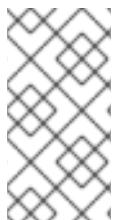
return {'resources': resources}

```

You will need this template in addition to the **02_lb_ext.py** template when you create an external cluster.

11.11. CREATING A PRIVATE DNS ZONE IN GCP

You must configure a private DNS zone in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create this component 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

- Ensure you defined the variables in the *Exporting common variables* and *Creating load balancers in GCP* sections.

Procedure

- 1 Copy the template from the **Deployment Manager template for the private DNS** section of this topic and save it as **02_dns.py** on your computer. This template describes the private DNS objects that your cluster requires.
- 2 Create a **02_dns.yaml** resource definition file:

```

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

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' 1
    cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' 2
    cluster_network: '${CLUSTER_NETWORK}' 3
EOF

```

- 1** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 2** **cluster_domain** is the domain for the cluster, for example **openshift.example.com**.
- 3** **cluster_network** is the **selfLink** URL to the cluster network.

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

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

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

- a. Add the internal DNS entries:

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

- b. For an external cluster, also add the external DNS entries:

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

11.11.1. Deployment Manager template for the private DNS

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

Example 11.5. 02_dns.py Deployment Manager template

```
def GenerateConfig(context):
    resources = [
        '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']
                ]
            }
        }
    ]
```

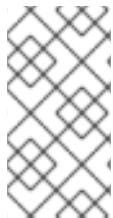
```

        }
    }]
    return {'resources': resources}

```

11.12. CREATING FIREWALL RULES IN GCP

You must create firewall rules 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

- Ensure you defined the variables in the *Exporting common variables* and *Creating load balancers in GCP* sections.

Procedure

- 1 Copy the template from the **Deployment Manager template for firewall rules** section of this topic and save it as **03_firewall.py** on your computer. This template describes the security groups that your cluster requires.
- 2 Create a **03_firewall.yaml** resource definition file:

```

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

resources:
- name: cluster-firewall
  type: 03_firewall.py
  properties:
    allowed_external_cidr: '0.0.0.0/0' 1
    infra_id: '${INFRA_ID}' 2
    cluster_network: '${CLUSTER_NETWORK}' 3
    network_cidr: '${NETWORK_CIDR}' 4
EOF

```

- 1** **allowed_external_cidr** is the CIDR range that can access the cluster API and SSH to the bootstrap host. For an internal cluster, set this value to **\${NETWORK_CIDR}**.
- 2** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 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**.

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

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

11.12.1. Deployment Manager template for firewall rules

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

Example 11.6. 03_firewall.py Deployment Manager template

```
def GenerateConfig(context):
    resources = [
        {
            'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
            'type': 'compute.v1.firewall',
            'properties': {
                'network': context.properties['cluster_network'],
                'allowed': [
                    {
                        'IPProtocol': 'tcp',
                        'ports': ['22']
                    }
                ],
                'sourceRanges': [context.properties['allowed_external_cidr']],
                'targetTags': [context.properties['infra_id'] + '-bootstrap']
            }
        },
        {
            'name': context.properties['infra_id'] + '-api',
            'type': 'compute.v1.firewall',
            'properties': {
                'network': context.properties['cluster_network'],
                'allowed': [
                    {
                        'IPProtocol': 'tcp',
                        'ports': ['6443']
                    }
                ],
                'sourceRanges': [context.properties['allowed_external_cidr']],
                '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', '6443', '22624']
                    }
                ],
                'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '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', '10250']
                    }
                ],
                'sourceRanges': [context.properties['allowed_external_cidr']],
                'targetTags': [context.properties['infra_id'] + '-etcd']
            }
        }
    ]
    return resources
```

```

'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']
                },
                {
                    'IPProtocol': 'tcp',
                    'ports': ['22623']
                }
            ],
            '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': 'udp',
      'ports': ['500', '4500']
    },{
      'IPProtocol': 'esp',
    },{
      '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'
    ]
  }
}

return {'resources': resources}

```

11.13. CREATING IAM ROLES IN GCP

You must create IAM 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

- You have defined the variables in the *Exporting common variables* section.

Procedure

1. Copy the template from the **Deployment Manager template for IAM roles** section of this topic and save it as **03_iam.py** on your computer. This template describes the IAM roles that your cluster requires.
2. Create a **03_iam.yaml** resource definition file:

```
$ cat <<EOF >03_iam.yaml
imports:
- path: 03_iam.py
resources:
- name: cluster-iam
  type: 03_iam.py
  properties:
    infra_id: '${INFRA_ID}' 1
EOF
```

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

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

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
```

4. Export the variable for the master service account:

```
$ export MASTER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

5. Export the variable for the worker service account:

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

6. Assign the permissions that the installation program requires to the service accounts for the subnets that host the control plane and compute subnets:

- a. Grant the **networkViewer** role of the project that hosts your shared VPC to the master service account:

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
projects add-iam-policy-binding ${HOST_PROJECT} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role
"roles/compute.networkViewer"
```

- b. Grant the **networkUser** role to the master service account for the control plane subnet:

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_CONTROL_SUBNET}" --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkUser"
--region ${REGION}
```

- c. Grant the **networkUser** role to the worker service account for the control plane subnet:

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_CONTROL_SUBNET}" --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role
"roles/compute.networkUser" --region ${REGION}
```

- d. Grant the **networkUser** role to the master service account for the compute subnet:

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_COMPUTE_SUBNET}" --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkUser"
--region ${REGION}
```

- e. Grant the **networkUser** role to the worker service account for the compute subnet:

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_COMPUTE_SUBNET}" --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role
"roles/compute.networkUser" --region ${REGION}
```

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

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. 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_SERVICE_ACCOUNT}
```

11.13.1. Deployment Manager template for IAM roles

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

Example 11.7. 03_iam.py Deployment Manager template

```

def GenerateConfig(context):

    resources = [
        {
            '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}

```

11.14. 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 [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>-<arch>-gcp.<arch>.tar.gz**.

2. Create the Google storage bucket:

```
$ gsutil mb gs://<bucket_name>
```

3. Upload the RHCOS image to the Google storage bucket:

```
$ gsutil cp <downloaded_image_file_path>/rhcoss-<version>-x86_64-gcp.x86_64.tar.gz
gs://<bucket_name>
```

4. Export the uploaded RHCOS image location as a variable:

-

```
$ export IMAGE_SOURCE=gs://<bucket_name>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
```

5. Create the cluster image:

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

11.15. 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

- Ensure you defined the variables in the *Exporting common variables* and *Creating load balancers in GCP* sections.
- Ensure you installed pyOpenSSL.

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 location of the Red Hat Enterprise Linux CoreOS (RHCOS) image that the installation program requires:

```
$ export CLUSTER_IMAGE=`gcloud compute images describe ${INFRA_ID}-rhcos-image --format json | jq -r .selfLink`
```

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

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

```
$ gsutil cp <installation_directory>/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-central1**.
- 3** **zone** is the zone to deploy the bootstrap instance into, for example **us-central1-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** **root_volume_size** is the boot disk size for the bootstrap machine.
- 9** **bootstrap_ign** is the URL output when creating a signed URL.

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

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

7. Add the bootstrap instance to the internal load balancer instance group:

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-bootstrap-ig --
zone=${ZONE_0} --instances=${INFRA_ID}-bootstrap
```

8. Add the bootstrap instance group to the internal load balancer backend service:

```
$ gcloud compute backend-services add-backend ${INFRA_ID}-api-internal --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-ig --instance-group-zone=${ZONE_0}
```

11.15.1. 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:

Example 11.8. 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',
            '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'] +
                            '"}},"version":"3.2.0"}},'
                        }
                    ]
                },
                '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']
            }
        },
    ]
```

```

'name': context.properties['infra_id'] + '-bootstrap-ig',
'type': 'compute.v1.instanceGroup',
'properties': {
  'namedPorts': [
    {
      'name': 'ignition',
      'port': 22623
    },
    {
      'name': 'https',
      'port': 6443
    }
  ],
  'network': context.properties['cluster_network'],
  'zone': context.properties['zone']
}
}

return {'resources': resources}

```

11.16. 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

- Ensure you defined the variables in the *Exporting common variables*, *Creating load balancers in GCP*, *Creating IAM roles in GCP*, and *Creating the bootstrap machine in GCP* sections.
- 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 variable required by the resource definition:

```
$ export MASTER_IGNITION=`cat <installation_directory>/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: '${INFRA_ID}' 1
    zones: 2
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' 3
    image: '${CLUSTER_IMAGE}' 4
    machine_type: 'n1-standard-4' 5
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}' 6

    ignition: '${MASTER_IGNITION}' 7
EOF

```

- 1** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 2** **zones** are the zones to deploy the control plane instances into, for example **us-central1-a**, **us-central1-b**, and **us-central1-c**.
- 3** **control_subnet** is the **selfLink** URL to the control subnet.
- 4** **image** is the **selfLink** URL to the RHCOS image.
- 5** **machine_type** is the machine type of the instance, for example **n1-standard-4**.
- 6** **service_account_email** is the email address for the master service account that you created.
- 7** **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 load balancer membership due to limitations of Deployment Manager, so you must add the control plane machines manually.

- Run the following commands to add the control plane machines to the appropriate instance groups:

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_0}-ig --zone=${ZONE_0} --instances=${INFRA_ID}-master-0
```

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_1}-ig --zone=${ZONE_1} --instances=${INFRA_ID}-master-1
```

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-${ZONE_2}-ig --zone=${ZONE_2} --instances=${INFRA_ID}-master-2
```

- For an external cluster, you must also run the following commands to add the control plane machines to the target pools:

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-zone="${ZONE_0}" --instances=${INFRA_ID}-master-0
```

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-zone="${ZONE_1}" --instances=${INFRA_ID}-master-1
```

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-zone="${ZONE_2}" --instances=${INFRA_ID}-master-2
```

11.16.1. 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:

Example 11.9. 05_control_plane.py Deployment Manager template

```
def GenerateConfig(context):
    resources = [
        {
            'name': context.properties['infra_id'] + '-master-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'] + '-master-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'] + '-master-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}

```

11.17. 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

- Ensure you defined the variables in the *Exporting common variables* and *Creating load balancers in GCP* sections.
- 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 backend-services remove-backend ${INFRA_ID}-api-internal --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-ig --instance-group-zone=${ZONE_0}

$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign

$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition

$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

11.18. 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

- Ensure you defined the variables in the *Exporting common variables*, *Creating load balancers in GCP*, and *Creating the bootstrap machine in GCP* sections.
- 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 variables that the resource definition uses.
 - a. Export the subnet that hosts the compute machines:

```
$ export COMPUTE_SUBNET=$(gcloud compute networks subnets describe ${HOST_PROJECT_COMPUTE_SUBNET} --region=${REGION} --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} --format json | jq -r
```

```
    .selfLink')
```

- b. Export the email address for your service account:

```
$ export WORKER_SERVICE_ACCOUNT=`gcloud iam service-accounts list --filter "email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '[0].email'`
```

- c. Export the location of the compute machine Ignition config file:

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
```

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

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

resources:
- name: 'worker-0' 1
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' 2
    zone: '${ZONE_0}' 3
    compute_subnet: '${COMPUTE_SUBNET}' 4
    image: '${CLUSTER_IMAGE}' 5
    machine_type: 'n1-standard-4' 6
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' 7
    ignition: '${WORKER_IGNITION}' 8
- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' 9
    zone: '${ZONE_1}' 10
    compute_subnet: '${COMPUTE_SUBNET}' 11
    image: '${CLUSTER_IMAGE}' 12
    machine_type: 'n1-standard-4' 13
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' 14
    ignition: '${WORKER_IGNITION}' 15
EOF
```

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

2 **9** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

3 **10** **zone** is the zone to deploy the worker machine into, for example **us-central1-a**.

4 **11** **compute_subnet** is the **selfLink** URL to the compute subnet.

5 **12** **image** is the **selfLink** URL to the RHCOS image.¹

6 **13** **machine_type** is the machine type of the instance, for example **n1-standard-4**.

7 14 **service_account_email** is the email address for the worker service account that you created.

8 15 **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
```

1. To use a GCP Marketplace image, specify the offer to use:

- OpenShift Container Platform:
<https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-413-x86-64-202305021736>
- OpenShift Platform Plus: <https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-413-x86-64-202305021736>
- OpenShift Kubernetes Engine:
<https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-413-x86-64-202305021736>

11.18.1. 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:

Example 11.10. 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}

```

11.19. INSTALLING THE OPENSHIFT CLI

You can install the OpenShift CLI (**oc**) to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.18. Download and install the new version of **oc**.

Installing the OpenShift CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** drop-down list.
3. Select the appropriate version from the **Version** drop-down list.
4. Click **Download Now** next to the **OpenShift v4.18 Linux Clients** entry and save the file.
5. Unpack the archive:

```
$ tar xvf <file>
```

6. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

Verification

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

```
$ oc <command>
```

Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.18 Windows Client** entry and save the file.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.

To check your **PATH**, open the command prompt and execute the following command:

```
C:> path
```

Verification

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

```
C:> oc <command>
```

Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.18 macOS Clients** entry and save the file.



NOTE

For macOS arm64, choose the **OpenShift v4.18 macOS arm64 Client** entry.

4. Unpack and unzip the archive.

5. Move the **oc** binary to a directory on your **PATH**.

To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

Verification

- Verify your installation by using an **oc** command:

```
$ oc <command>
```

11.20. LOGGING IN TO THE CLUSTER BY USING THE CLI

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

- You installed the **oc** CLI.
- Ensure the bootstrap process completed successfully.

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
```

Example output

```
system:admin
```

11.21. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	63m	v1.31.3
master-1	Ready	master	63m	v1.31.3
master-2	Ready	master	64m	v1.31.3

The output lists all of the machines that you created.



NOTE

The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

- Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

NAME	AGE	REQUESTOR	CONDITION
csr-8b2br	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
csr-8vnps	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
...			

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 the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.



NOTE

For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- 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.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}\n{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



NOTE

Some Operators might not become available until some CSRs are approved.

4. Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

NAME	AGE	REQUESTOR	CONDITION
csr-bfd72	5m26s	system:node:ip-10-0-50-126.us-east-2.compute.internal	Pending
csr-c57lv	5m26s	system:node:ip-10-0-95-157.us-east-2.compute.internal	Pending
...			

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- 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.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}\n{{end}}{{end}}' | xargs oc adm certificate approve
```

6. After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	73m	v1.31.3
master-1	Ready	master	73m	v1.31.3
master-2	Ready	master	74m	v1.31.3
worker-0	Ready	worker	11m	v1.31.3
worker-1	Ready	worker	11m	v1.31.3



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- [Certificate Signing Requests](#)

11.22. ADDING THE INGRESS DNS RECORDS

DNS zone configuration is removed 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

- Ensure you defined the variables in the *Exporting common variables* section.
- Remove the DNS Zone configuration when creating Kubernetes manifests and generating Ignition configs.
- Ensure the bootstrap process completed successfully.

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
```

Example output

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 zones:

- To use A records:

- i. Export the variable for the router IP address:

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

- ii. Add the A record to the private zones:

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

- iii. For an external cluster, also add the A record to the public zones:

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

- To add explicit domains instead of using a wildcard, 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
```

Example output

```
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
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

11.23. ADDING INGRESS FIREWALL RULES

The cluster requires several firewall rules. If you do not use a shared VPC, these rules are created by the Ingress Controller via the GCP cloud provider. When you use a shared VPC, you can either create cluster-wide firewall rules for all services now or create each rule based on events, when the cluster requests access. By creating each rule when the cluster requests access, you know exactly which firewall rules are required. By creating cluster-wide firewall rules, you can apply the same rule set across multiple clusters.

If you choose to create each rule based on events, you must create firewall rules after you provision the cluster and during the life of the cluster when the console notifies you that rules are missing. Events that are similar to the following event are displayed, and you must add the firewall rules that are required:

```
$ oc get events -n openshift-ingress --field-selector="reason=LoadBalancerManualChange"
```

Example output

```
Firewall change required by security admin: `gcloud compute firewall-rules create k8s-fw-a26e631036a3f46cba28f8df67266d55 --network example-network --description "{\"kubernetes.io/service-name\":\"\"}:\\"openshift-ingress/router-default\\\", \"kubernetes.io/service-ip\\\":\"35.237.236.234\\\\"\\" --allow tcp:443,tcp:80 --source-ranges 0.0.0.0/0 --target-tags exempl-fqzq7-master,exempl-fqzq7-worker --project example-project`
```

If you encounter issues when creating these rule-based events, you can configure the cluster-wide firewall rules while your cluster is running.

11.23.1. Creating cluster-wide firewall rules for a shared VPC in GCP

You can create cluster-wide firewall rules to allow the access that the OpenShift Container Platform cluster requires.



WARNING

If you do not choose to create firewall rules based on cluster events, you must create cluster-wide firewall rules.

Prerequisites

- You exported the variables that the Deployment Manager templates require to deploy your cluster.
- You created the networking and load balancing components in GCP that your cluster requires.

Procedure

1. Add a single firewall rule to allow the Google Cloud Engine health checks to access all of the services. This rule enables the ingress load balancers to determine the health status of their instances.

```
$ gcloud compute firewall-rules create --allow='tcp:30000-32767,udp:30000-32767' --network="${CLUSTER_NETWORK}" --source-
```

```
ranges='130.211.0.0/22,35.191.0.0/16,209.85.152.0/22,209.85.204.0/22' --target-tags="${INFRA_ID}-master,${INFRA_ID}-worker" ${INFRA_ID}-ingress-hc --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
```

2. Add a single firewall rule to allow access to all cluster services:

- For an external cluster:

```
$ gcloud compute firewall-rules create --allow='tcp:80,tcp:443' --network="${CLUSTER_NETWORK}" --source-ranges="0.0.0.0/0" --target-tags="${INFRA_ID}-master,${INFRA_ID}-worker" ${INFRA_ID}-ingress --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
```

- For a private cluster:

```
$ gcloud compute firewall-rules create --allow='tcp:80,tcp:443' --network="${CLUSTER_NETWORK}" --source-ranges=${NETWORK_CIDR} --target-tags="${INFRA_ID}-master,${INFRA_ID}-worker" ${INFRA_ID}-ingress --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
```

Because this rule only allows traffic on TCP ports **80** and **443**, ensure that you add all the ports that your services use.

11.24. 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

- Ensure the bootstrap process completed successfully.

Procedure

1. Complete the cluster installation:

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

Example output

```
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, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

2. Observe the running state of your cluster.

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

```
$ oc get clusterversion
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	SINCE	STATUS
version		False	True	24m	Working towards 4.5.4: 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
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED
SINCE				
authentication	4.5.4	True	False	False 7m56s
cloud-credential	4.5.4	True	False	False 31m
cluster-autoscaler	4.5.4	True	False	False 16m
console	4.5.4	True	False	False 10m
csi-snapshot-controller	4.5.4	True	False	False 16m
dns	4.5.4	True	False	False 22m
etcd	4.5.4	False	False	False 25s
image-registry	4.5.4	True	False	False 16m
ingress	4.5.4	True	False	False 16m
insights	4.5.4	True	False	False 17m
kube-apiserver	4.5.4	True	False	False 19m
kube-controller-manager	4.5.4	True	False	False 20m
kube-scheduler	4.5.4	True	False	False 20m
kube-storage-version-migrator	4.5.4	True	False	False 16m
machine-api	4.5.4	True	False	False 22m
machine-config	4.5.4	True	False	False 22m
marketplace	4.5.4	True	False	False 16m

monitoring	4.5.4	True	False	False	10m
network	4.5.4	True	False	False	23m
node-tuning	4.5.4	True	False	False	23m
openshift-apiserver	4.5.4	True	False	False	17m
openshift-controller-manager	4.5.4	True	False	False	15m
openshift-samples	4.5.4	True	False	False	16m
operator-lifecycle-manager	4.5.4	True	False	False	22m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	22m
operator-lifecycle-manager-packageserver	4.5.4	True	False	False	18m
service-ca	4.5.4	True	False	False	23m
service-catalog-apiserver	4.5.4	True	False	False	23m
service-catalog-controller-manager	4.5.4	True	False	False	23m
storage	4.5.4	True	False	False	17m

c. Run the following command to view your cluster pods:

```
$ oc get pods --all-namespaces
```

Example output

NAMESPACE	READY	STATUS	RESTARTS	AGE	NAME
kube-system					etcd-member-ip-10-0-3-111.us-east-
2.compute.internal	1/1	Running	0	35m	etcd-member-ip-10-0-3-239.us-east-
kube-system					etcd-member-ip-10-0-3-24.us-east-
2.compute.internal	1/1	Running	0	37m	openshift-apiserver-operator-6d6674f4f4-
h7t2t	1/1	Running	1	37m	apiserver-fm48r
openshift-apiserver	1/1	Running	0	30m	apiserver-fxkvv
openshift-apiserver	1/1	Running	0	29m	apiserver-q85nm
openshift-apiserver	1/1	Running	0	29m	...
openshift-service-ca-operator	1/1	Running	0	37m	openshift-service-ca-operator-66ff6dc6cd-
9r257					apiservice-cabundle-injector-695b6bcfc-cl5hm
openshift-service-ca	1/1	Running	0	35m	configmap-cabundle-injector-8498544d7-
25qn6	1/1	Running	0	35m	service-serving-cert-signer-6445fc9c6-wqdqn
openshift-service-ca	1/1	Running	0	35m	openshift-service-catalog-apiserver-
openshift-service-catalog-apiserver-operator	1/1	Running	0	32m	operator-549f44668b-b5q2w
openshift-service-catalog-controller-manager-operator	1/1	Running	0	31m	controller-manager-operator-b78cr2lnm

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

11.25. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.18, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to [OpenShift Cluster Manager](#).

After you confirm that your [OpenShift Cluster Manager](#) inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See [About remote health monitoring](#) for more information about the Telemetry service

11.26. NEXT STEPS

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

CHAPTER 12. INSTALLING A CLUSTER ON GCP IN A DISCONNECTED ENVIRONMENT WITH USER-PROVISIONED INFRASTRUCTURE

In OpenShift Container Platform version 4.18, you can install a cluster on Google Cloud Platform (GCP) that uses 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 GCP APIs.

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.



IMPORTANT

The steps for performing a user-provisioned infrastructure installation are provided as an example only. Installing a cluster with infrastructure you provide requires knowledge of the cloud provider and the installation process of OpenShift Container Platform. 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.

12.1. PREREQUISITES

- You reviewed details about the [OpenShift Container Platform installation and update processes](#).
- You read the documentation on [selecting a cluster installation method and preparing it for users](#).
- You [created a registry on your mirror host](#) and obtained the **imageContentSources** data for your version of OpenShift Container Platform.



IMPORTANT

Because the installation media is on the mirror host, you can use that computer to complete all installation steps.

- If you use a firewall, you [configured it to allow the sites](#) that your cluster requires access to. While you might need to grant access to more sites, you must grant access to ***.googleapis.com** and **accounts.google.com**.
- If the cloud identity and access management (IAM) APIs are not accessible in your environment, or if you do not want to store an administrator-level credential secret in the **kube-system** namespace, you can [manually create and maintain long-term credentials](#).

12.2. ABOUT INSTALLATIONS IN RESTRICTED NETWORKS

In OpenShift Container Platform 4.18, you can perform an installation that does not require an active connection to the internet to obtain software components. Restricted network installations can be completed using installer-provisioned infrastructure or user-provisioned infrastructure, depending on the cloud platform to which you are installing the cluster.

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 Route 53 DNS and IAM services, require internet access. Depending on your network, you might require less internet access for an installation on bare metal hardware, Nutanix, or on VMware vSphere.

To complete a restricted network installation, you must create a registry that mirrors the contents of the OpenShift image registry and contains the installation media. You can create this registry on a mirror host, which can access both the internet and your closed network, or by using other methods that meet your restrictions.



IMPORTANT

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 using user-provisioned infrastructure.

Completing this test installation might make it easier to isolate and troubleshoot any issues that might arise during your installation in a restricted network.

12.2.1. 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 image stream tags.

12.3. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.18, you require access to the internet to obtain the images that are necessary to install your cluster.

You must have internet access to:

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

12.4. CONFIGURING YOUR GCP PROJECT

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

12.4.1. 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.



IMPORTANT

Your GCP project must use the Premium Network Service Tier if you are using installer-provisioned infrastructure. The Standard Network Service Tier is not supported for clusters installed using the installation program. The installation program configures internal load balancing for the **api-int.<cluster_name>. <base_domain>** URL; the Premium Tier is required for internal load balancing.

12.4.2. 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. You may also enable optional API services which are not required for installation. See [Enabling services](#) in the GCP documentation.

Table 12.1. Required API services

API service	Console service name
Compute Engine API	compute.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 Usage API	serviceusage.googleapis.com

Table 12.2. Optional API services

API service	Console service name
Google Cloud APIs	cloudapis.googleapis.com
Service Management API	servicemanagement.googleapis.com
Google Cloud Storage JSON API	storage-api.googleapis.com
Cloud Storage	storage-component.googleapis.com

12.4.3. 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 migrated your root domain to Google Cloud DNS, migrate your DNS records. See [Migrating to Cloud DNS](#) in the GCP documentation.
6. If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain. This process might include a request to your company's IT department or the division that controls the root domain and DNS services for your company.

12.4.4. 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 12.3. GCP resources used in a default cluster

Service	Component	Location	Total resources required	Resources removed after bootstrap
Service account	IAM	Global	6	1
Firewall rules	Networking	Global	11	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	2	0
Subnetworks	Compute	Global	2	0
Target pools	Networking	Global	2	0



NOTE

If any of the quotas are insufficient during installation, the installation program displays an error that states both which quota was exceeded and the region.

Be sure to consider your actual cluster size, planned cluster growth, and any usage from other clusters that are associated with your account. The CPU, static IP addresses, and persistent disk SSD (storage) quotas are the ones that are most likely to be insufficient.

If you plan to deploy your cluster in one of the following regions, you will exceed the maximum storage quota and are likely to exceed the CPU quota limit:

- **asia-east2**
- **asia-northeast2**
- **asia-south1**

- **australia-southeast1**
- **europe-north1**
- **europe-west2**
- **europe-west3**
- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

You can increase resource quotas from the [GCP console](#), but you might need to file a support ticket. Be sure to plan your cluster size early so that you can allow time to resolve the support ticket before you install your OpenShift Container Platform cluster.

12.4.5. Creating a service account in GCP

OpenShift Container Platform requires a Google Cloud Platform (GCP) service account that provides authentication and authorization to access data in the Google APIs. If you do not have an existing IAM service account that contains the required roles in your project, you must create one.

Prerequisites

- You created a project to host your cluster.

Procedure

1. Create a 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](#).



NOTE

While making the service account an owner of the project is the easiest way to gain the required permissions, it means that service account has complete control over the project. You must determine if the risk that comes from offering that power is acceptable.

3. You can create the service account key in JSON format, or attach the service account to a GCP virtual machine. See [Creating service account keys](#) and [Creating and enabling service accounts for instances](#) in the GCP documentation.



NOTE

If you use a virtual machine with an attached service account to create your cluster, you must set **credentialsMode: Manual** in the **install-config.yaml** file before installation.

12.4.6. Required GCP roles

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. If your organization's security policies require a more restrictive set of permissions, you can create a service account with the following permissions. If you deploy your cluster into an existing virtual private cloud (VPC), the service account does not require certain networking permissions, which are noted in the following lists:

Required roles for the installation program

- Compute Admin
- Role Administrator
- Security Admin
- Service Account Admin
- Service Account Key Admin
- Service Account User
- Storage Admin

Required roles for creating network resources during installation

- DNS Administrator

Required roles for using the Cloud Credential Operator in passthrough mode

- Compute Load Balancer Admin
- Tag User

Required roles for user-provisioned GCP infrastructure

- Deployment Manager Editor

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

Table 12.4. GCP service account roles

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

Account	Roles
	roles/iam.serviceAccountUser
Compute	roles/compute.viewer
	roles/storage.admin
	roles/artifactregistry.reader

12.4.7. Required GCP permissions for user-provisioned infrastructure

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.

If your organization's security policies require a more restrictive set of permissions, you can create [custom roles](#) with the necessary permissions. The following permissions are required for the user-provisioned infrastructure for creating and deleting the OpenShift Container Platform cluster.

Example 12.1. Required permissions for creating network resources

- **compute.addresses.create**
- **compute.addresses.createInternal**
- **compute.addresses.delete**
- **compute.addresses.get**
- **compute.addresses.list**
- **compute.addresses.use**
- **compute.addresses.useInternal**
- **compute.firewalls.create**
- **compute.firewalls.delete**
- **compute.firewalls.get**
- **compute.firewalls.list**
- **compute.forwardingRules.create**
- **compute.forwardingRules.get**
- **compute.forwardingRules.list**
- **compute.forwardingRules.setLabels**
- **compute.globalAddresses.create**
- **compute.globalAddresses.get**

- `compute.globalAddresses.use`
- `compute.globalForwardingRules.create`
- `compute.globalForwardingRules.get`
- `compute.globalForwardingRules.setLabels`
- `compute.networks.create`
- `compute.networks.get`
- `compute.networks.list`
- `compute.networks.updatePolicy`
- `compute.networks.use`
- `compute.routers.create`
- `compute.routers.get`
- `compute.routers.list`
- `compute.routers.update`
- `compute.routes.list`
- `compute.subnetworks.create`
- `compute.subnetworks.get`
- `compute.subnetworks.list`
- `compute.subnetworks.use`
- `compute.subnetworks.useExternalIp`

Example 12.2. Required permissions for creating load balancer resources

- `compute.backendServices.create`
- `compute.backendServices.get`
- `compute.backendServices.list`
- `compute.backendServices.update`
- `compute.backendServices.use`
- `compute.regionBackendServices.create`
- `compute.regionBackendServices.get`
- `compute.regionBackendServices.list`
- `compute.regionBackendServices.update`

- `compute.regionBackendServices.use`
- `compute.targetPools.addInstance`
- `compute.targetPools.create`
- `compute.targetPools.get`
- `compute.targetPools.list`
- `compute.targetPools.removeInstance`
- `compute.targetPools.use`
- `compute.targetTcpProxies.create`
- `compute.targetTcpProxies.get`
- `compute.targetTcpProxies.use`

Example 12.3. Required permissions for creating DNS resources

- `dns.changes.create`
- `dns.changes.get`
- `dns.managedZones.create`
- `dns.managedZones.get`
- `dns.managedZones.list`
- `dns.networks.bindPrivateDNSZone`
- `dns.resourceRecordSets.create`
- `dns.resourceRecordSets.list`
- `dns.resourceRecordSets.update`

Example 12.4. Required permissions for creating Service Account resources

- `iam.serviceAccountKeys.create`
- `iam.serviceAccountKeys.delete`
- `iam.serviceAccountKeys.get`
- `iam.serviceAccountKeys.list`
- `iam.serviceAccounts.actAs`
- `iam.serviceAccounts.create`
- `iam.serviceAccounts.delete`

- **iam.serviceAccounts.get**
- **iam.serviceAccounts.list**
- **resourcemanager.projects.get**
- **resourcemanager.projects.getIamPolicy**
- **resourcemanager.projects.setIamPolicy**

Example 12.5. Required permissions for creating compute resources

- **compute.disks.create**
- **compute.disks.get**
- **compute.disks.list**
- **compute.instanceGroups.create**
- **compute.instanceGroups.delete**
- **compute.instanceGroups.get**
- **compute.instanceGroups.list**
- **compute.instanceGroups.update**
- **compute.instanceGroups.use**
- **compute.instances.create**
- **compute.instances.delete**
- **compute.instances.get**
- **compute.instances.list**
- **compute.instances.setLabels**
- **compute.instances.setMetadata**
- **compute.instances.setServiceAccount**
- **compute.instances.setTags**
- **compute.instances.use**
- **compute.machineTypes.get**
- **compute.machineTypes.list**

Example 12.6. Required for creating storage resources

- **storage.buckets.create**

- **storage.buckets.delete**
- **storage.buckets.get**
- **storage.buckets.list**
- **storage.objects.create**
- **storage.objects.delete**
- **storage.objects.get**
- **storage.objects.list**

Example 12.7. Required permissions for creating health check resources

- **compute.healthChecks.create**
- **compute.healthChecks.get**
- **compute.healthChecks.list**
- **compute.healthChecks.useReadOnly**
- **compute.httpHealthChecks.create**
- **compute.httpHealthChecks.get**
- **compute.httpHealthChecks.list**
- **compute.httpHealthChecks.useReadOnly**
- **compute.regionHealthChecks.create**
- **compute.regionHealthChecks.get**
- **compute.regionHealthChecks.useReadOnly**

Example 12.8. Required permissions to get GCP zone and region related information

- **compute.globalOperations.get**
- **compute.regionOperations.get**
- **compute.regions.get**
- **compute.regions.list**
- **compute.zoneOperations.get**
- **compute.zones.get**
- **compute.zones.list**

Example 12.9. Required permissions for checking services and quotas

- **monitoring.timeSeries.list**
- **serviceusage.quotas.get**
- **serviceusage.services.list**

Example 12.10. Required IAM permissions for installation

- **iam.roles.get**

Example 12.11. Required permissions when authenticating without a service account key

- **iam.serviceAccounts.signBlob**

Example 12.12. Required Images permissions for installation

- **compute.images.create**
- **compute.images.delete**
- **compute.images.get**
- **compute.images.list**

Example 12.13. Optional permission for running gather bootstrap

- **compute.instances.getSerialPortOutput**

Example 12.14. Required permissions for deleting network resources

- **compute.addresses.delete**
- **compute.addresses.deleteInternal**
- **compute.addresses.list**
- **compute.addresses.setLabels**
- **compute.firewalls.delete**
- **compute.firewalls.list**
- **compute.forwardingRules.delete**
- **compute.forwardingRules.list**
- **compute.globalAddresses.delete**
- **compute.globalAddresses.list**

- **compute.globalForwardingRules.delete**
- **compute.globalForwardingRules.list**
- **compute.networks.delete**
- **compute.networks.list**
- **compute.networks.updatePolicy**
- **compute.routers.delete**
- **compute.routers.list**
- **compute.routes.list**
- **compute.subnetworks.delete**
- **compute.subnetworks.list**

Example 12.15. Required permissions for deleting load balancer resources

- **compute.backendServices.delete**
- **compute.backendServices.list**
- **compute.regionBackendServices.delete**
- **compute.regionBackendServices.list**
- **compute.targetPools.delete**
- **compute.targetPools.list**
- **compute.targetTcpProxies.delete**
- **compute.targetTcpProxies.list**

Example 12.16. Required permissions for deleting DNS resources

- **dns.changes.create**
- **dns.managedZones.delete**
- **dns.managedZones.get**
- **dns.managedZones.list**
- **dns.resourceRecordSets.delete**
- **dns.resourceRecordSets.list**

Example 12.17. Required permissions for deleting Service Account resources

- **iam.serviceAccounts.delete**
- **iam.serviceAccounts.get**
- **iam.serviceAccounts.list**
- **resourcemanager.projects.getIamPolicy**
- **resourcemanager.projects.setIamPolicy**

Example 12.18. Required permissions for deleting compute resources

- **compute.disks.delete**
- **compute.disks.list**
- **compute.instanceGroups.delete**
- **compute.instanceGroups.list**
- **compute.instances.delete**
- **compute.instances.list**
- **compute.instances.stop**
- **compute.machineTypes.list**

Example 12.19. Required for deleting storage resources

- **storage.buckets.delete**
- **storage.buckets.getIamPolicy**
- **storage.buckets.list**
- **storage.objects.delete**
- **storage.objects.list**

Example 12.20. Required permissions for deleting health check resources

- **compute.healthChecks.delete**
- **compute.healthChecks.list**
- **compute.httpHealthChecks.delete**
- **compute.httpHealthChecks.list**
- **compute.regionHealthChecks.delete**
- **compute.regionHealthChecks.list**

Example 12.21. Required Images permissions for deletion

- **compute.images.delete**
- **compute.images.list**

Example 12.22. Required permissions to get Region related information

- **compute.regions.get**

Example 12.23. Required Deployment Manager permissions

- **deploymentmanager.deployments.create**
- **deploymentmanager.deployments.delete**
- **deploymentmanager.deployments.get**
- **deploymentmanager.deployments.list**
- **deploymentmanager.manifests.get**
- **deploymentmanager.operations.get**
- **deploymentmanager.resources.list**

Additional resources

- [Optimizing storage](#)

12.4.8. Supported GCP regions

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

- **africa-south1** (Johannesburg, South Africa)
- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-northeast3** (Seoul, South Korea)
- **asia-south1** (Mumbai, India)
- **asia-south2** (Delhi, India)

- **asia-southeast1** (Jurong West, Singapore)
- **asia-southeast2** (Jakarta, Indonesia)
- **australia-southeast1** (Sydney, Australia)
- **australia-southeast2** (Melbourne, Australia)
- **europe-central2** (Warsaw, Poland)
- **europe-north1** (Hamina, Finland)
- **europe-southwest1** (Madrid, Spain)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)
- **europe-west6** (Zürich, Switzerland)
- **europe-west8** (Milan, Italy)
- **europe-west9** (Paris, France)
- **europe-west12** (Turin, Italy)
- **me-central1** (Doha, Qatar, Middle East)
- **me-central2** (Dammam, Saudi Arabia, Middle East)
- **me-west1** (Tel Aviv, Israel)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **northamerica-northeast2** (Toronto, Ontario, Canada)
- **southamerica-east1** (São Paulo, Brazil)
- **southamerica-west1** (Santiago, Chile)
- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-east5** (Columbus, Ohio)
- **us-south1** (Dallas, Texas)
- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)

- **us-west3** (Salt Lake City, Utah, USA)
- **us-west4** (Las Vegas, Nevada, USA)



NOTE

To determine which machine type instances are available by region and zone, see the Google [documentation](#).

12.4.9. 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.
See [Authorizing with a service account](#) in the GCP documentation.

12.5. REQUIREMENTS FOR A CLUSTER WITH USER-PROVISIONED INFRASTRUCTURE

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

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

12.5.1. Required machines for cluster installation

The smallest OpenShift Container Platform clusters require the following hosts:

Table 12.5. Minimum required hosts

Hosts	Description
One temporary bootstrap machine	The cluster requires the bootstrap machine to deploy the OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster.

Hosts	Description
Three control plane machines	The control plane machines run the Kubernetes and OpenShift Container Platform services that form the control plane.
At least two compute machines, which are also known as worker machines.	The workloads requested by OpenShift Container Platform users run on the compute machines.



IMPORTANT

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

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

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

12.5.2. Minimum resource requirements for cluster installation

Each cluster machine must meet the following minimum requirements:

Table 12.6. Minimum resource requirements

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

- One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.
- OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.
- As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance,

including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.



NOTE

For OpenShift Container Platform version 4.18, RHCOS is based on RHEL version 9.4, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see [Architectures](#) (RHEL documentation).

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

12.5.3. Tested instance types for GCP

The following Google Cloud Platform instance types have been tested with OpenShift Container Platform.



NOTE

Not all instance types are available in all regions and zones. For a detailed breakdown of which instance types are available in which zones, see [regions and zones](#) (Google documentation).

Some instance types require the use of Hyperdisk storage. If you use an instance type that requires Hyperdisk storage, all of the nodes in your cluster must support Hyperdisk storage, and you must change the default storage class to use Hyperdisk storage. For more information, see [machine series support for Hyperdisk](#) (Google documentation). For instructions on modifying storage classes, see the "GCE PersistentDisk (gcePD) object definition" section in the Dynamic Provisioning page in [Storage](#).

Example 12.24. Machine series

- **A2**
- **A3**
- **C2**
- **C2D**
- **C3**
- **C3D**

- C4
- E2
- M1
- N1
- N2
- N2D
- N4
- Tau T2D

12.5.4. Using custom machine types

Using a custom machine type to install a OpenShift Container Platform cluster is supported.

Consider the following when using a custom machine type:

- Similar to predefined instance types, custom machine types must meet the minimum resource requirements for control plane and compute machines. For more information, see "Minimum resource requirements for cluster installation".
- The name of the custom machine type must adhere to the following syntax:
custom-<number_of_cpus>-<amount_of_memory_in_mb>

For example, **custom-6-20480**.

12.6. 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. You also have the option to first set up a separate **var** partition during the preparation phases of installation.

12.6.1. Optional: Creating a separate /var partition

It is recommended that disk partitioning for OpenShift Container Platform be left to the installer. However, there are cases where you might want to create separate partitions in a part of the filesystem that you expect to grow.

OpenShift Container Platform supports the addition of a single partition to attach storage to either the **/var** partition or a subdirectory of **/var**. For example:

- **/var/lib/containers**: Holds container-related content that can grow as more images and containers are added to a system.
- **/var/lib/etcd**: Holds data that you might want to keep separate for purposes such as performance optimization of etcd storage.

- **/var**: Holds data that you might want to keep separate for purposes such as auditing.

Storing the contents of a **/var** directory separately makes it easier to grow storage for those areas as needed and reinstall OpenShift Container Platform at a later date and keep that data intact. With this method, you will not have to pull all your containers again, nor will you have to copy massive log files when you update systems.

Because **/var** must be in place before a fresh installation of Red Hat Enterprise Linux CoreOS (RHCOS), the following procedure sets up the separate **/var** partition by creating a machine config manifest that is inserted during the **openshift-install** preparation phases of an OpenShift Container Platform installation.



IMPORTANT

If you follow the steps to create a separate **/var** partition in this procedure, it is not necessary to create the Kubernetes manifest and Ignition config files again as described later in this section.

Procedure

1. Create a directory to hold the OpenShift Container Platform installation files:

```
$ mkdir $HOME/clusterconfig
```

2. Run **openshift-install** to create a set of files in the **manifest** and **openshift** subdirectories. Answer the system questions as you are prompted:

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

Example output

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. Optional: Confirm that the installation program created manifests in the **clusterconfig/openshift** directory:

```
$ ls $HOME/clusterconfig/openshift/
```

Example output

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. Create a Butane config that configures the additional partition. For example, name the file **\$HOME/clusterconfig/98-var-partition.bu**, change the disk device name to the name of the storage device on the **worker** systems, and set the storage size as appropriate. This example

places the `/var` directory on a separate partition:

```
variant: openshift
version: 4.18.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
    - device: /dev/disk/by-id/<device_name> ①
      partitions:
        - label: var
          start_mib: <partition_start_offset> ②
          size_mib: <partition_size> ③
          number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] ④
      with_mount_unit: true
```

- ① The storage device name of the disk that you want to partition.
- ② When adding a data partition to the boot disk, a minimum value of 25000 MiB (Mebibytes) is recommended. The root file system is automatically resized to fill all available space up to the specified offset. If no value is specified, or if the specified value is smaller than the recommended minimum, the resulting root file system will be too small, and future reinstalls of RHCOS might overwrite the beginning of the data partition.
- ③ The size of the data partition in mebibytes.
- ④ The **prjquota** mount option must be enabled for filesystems used for container storage.



NOTE

When creating a separate `/var` partition, you cannot use different instance types for worker nodes, if the different instance types do not have the same device name.

5. Create a manifest from the Butane config and save it to the **clusterconfig/openshift** directory. For example, run the following command:

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. Run **openshift-install** again to create Ignition configs from a set of files in the **manifest** and **openshift** subdirectories:

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth bootstrap.ign master.ign metadata.json worker.ign
```

Now you can use the Ignition config files as input to the installation procedures to install Red Hat Enterprise Linux CoreOS (RHCOS) systems.

12.6.2. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP).

Prerequisites

- You have the OpenShift Container Platform installation program and the pull secret for your cluster. For a restricted network installation, these files are on your mirror host.
- You have the **imageContentSources** values that were generated during mirror registry creation.
- You have obtained the contents of the certificate for your mirror registry.
- Configure a GCP account.

Procedure

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

- a. Change to the directory that contains the installation program and 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.

When specifying the directory:

- Verify that the directory has the **execute** permission. This permission is required to run Terraform binaries under the installation directory.
- Use an empty directory. Some installation assets, such as bootstrap X.509 certificates, have short expiration intervals, therefore 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, 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.
2. Edit the **install-config.yaml** file to give 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": {"<mirror_host_name>:5000": {"auth": "<credentials>", "email": "you@example.com"}}}
```
- For **<mirror\_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.
- ```
additionalTrustBundle: |
-----BEGIN CERTIFICATE-----
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
-----END CERTIFICATE-----
```
- The value must be the contents of the certificate file that you used for your mirror registry. The certificate file can be an existing, trusted certificate authority, or the self-signed certificate that you generated for the mirror registry.
- c. Define the network and subnets for the VPC to install the cluster in under the parent **platform.gcp** field:
- ```
network: <existing_vpc>
controlPlaneSubnet: <control_plane_subnet>
computeSubnet: <compute_subnet>
```
- For **platform.gcp.network**, specify the name for the existing Google VPC. For **platform.gcp.controlPlaneSubnet** and **platform.gcp.computeSubnet**, specify the existing subnets to deploy the control plane machines and compute machines, respectively.
- d. Add the image content resources, which resemble the following YAML excerpt:
- ```
imageContentSources:
- mirrors:
  - <mirror_host_name>:<5000/><repo_name>/release
```

```

source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
source: registry.redhat.io/ocp/release

```

For these values, use the **imageContentSources** that you recorded during mirror registry creation.

- e. Optional: Set the publishing strategy to **Internal**:

```

publish: Internal

```

By setting this option, you create an internal Ingress Controller and a private load balancer.

3. Make any other modifications to the **install-config.yaml** file that you require. For more information about the parameters, see "Installation configuration parameters".
4. 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.

Additional resources

- [Installation configuration parameters for GCP](#)

12.6.3. Enabling Shielded VMs

You can use Shielded VMs when installing your cluster. Shielded VMs have extra security features including secure boot, firmware and integrity monitoring, and rootkit detection. For more information, see Google's documentation on [Shielded VMs](#).



NOTE

Shielded VMs are currently not supported on clusters with 64-bit ARM infrastructures.

Procedure

- Use a text editor to edit the **install-config.yaml** file prior to deploying your cluster and add one of the following stanzas:
 - a. To use shielded VMs for only control plane machines:

```

controlPlane:
platform:
gcp:
  secureBoot: Enabled

```

- b. To use shielded VMs for only compute machines:

```

compute:
- platform:

```

```
gcp:
  secureBoot: Enabled
```

- c. To use shielded VMs for all machines:

```
platform:
gcp:
  defaultMachinePlatform:
    secureBoot: Enabled
```

12.6.4. Enabling Confidential VMs

You can use Confidential VMs when installing your cluster. Confidential VMs encrypt data while it is being processed. For more information, see Google's documentation on [Confidential Computing](#). You can enable Confidential VMs and Shielded VMs at the same time, although they are not dependent on each other.



NOTE

Confidential VMs are currently not supported on 64-bit ARM architectures.

Procedure

- Use a text editor to edit the **install-config.yaml** file prior to deploying your cluster and add one of the following stanzas:

- a. To use confidential VMs for only control plane machines:

```
controlPlane:
  platform:
    gcp:
      confidentialCompute: Enabled ①
      type: n2d-standard-8 ②
      onHostMaintenance: Terminate ③
```

- ① Enable confidential VMs.
- ② Specify a machine type that supports Confidential VMs. Confidential VMs require the N2D or C2D series of machine types. For more information on supported machine types, see [Supported operating systems and machine types](#).
- ③ Specify the behavior of the VM during a host maintenance event, such as a hardware or software update. For a machine that uses Confidential VM, this value must be set to **Terminate**, which stops the VM. Confidential VMs do not support live VM migration.

- b. To use confidential VMs for only compute machines:

```
compute:
- platform:
  gcp:
    confidentialCompute: Enabled
    type: n2d-standard-8
    onHostMaintenance: Terminate
```

- c. To use confidential VMs for all machines:

```

platform:
gcp:
  defaultMachinePlatform:
    confidentialCompute: Enabled
    type: n2d-standard-8
    onHostMaintenance: Terminate

```

12.6.5. 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

- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

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: https://<username>:<pswd>@<ip>:<port> ②
  noProxy: example.com ③
  additionalTrustBundle: | ④
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> ⑤

```

- 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.

- 3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with . to match subdomains only. For example, exclude .example.com to match subdomains like www.example.com.
- 4 If provided, the installation program generates a config map 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** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **trustedCA** field of the **Proxy** object. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.
- 5 Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.



NOTE

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



NOTE

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

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**.



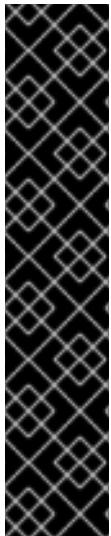
NOTE

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

12.6.6. 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 configure the machines.

The installation configuration file transforms into the Kubernetes manifests. The manifests wrap into the Ignition configuration files, which are later used to configure the cluster machines.



IMPORTANT

- The Ignition config files that the OpenShift Container Platform installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

Procedure

1. Change to the directory that contains the OpenShift Container Platform installation program and generate the Kubernetes manifests for the cluster:

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

- ① For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

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

```
$ rm -f <installation_directory>/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 control plane machine set:

```
$ rm -f <installation_directory>/openshift/99_openshift-machine-api_master-control-plane-machine-set.yaml
```

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

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



IMPORTANT

If you disabled the **MachineAPI** capability when installing a cluster on user-provisioned infrastructure, you must remove the Kubernetes manifest files that define the worker machines. Otherwise, your cluster fails to install.

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

5. Check that the **mastersSchedulable** parameter in the

<installation_directory>/manifests/cluster-scheduler-02-config.yml Kubernetes manifest file is set to **false**. This setting prevents pods from being scheduled on the control plane machines:

- a. Open the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** file.
 - b. Locate the **mastersSchedulable** parameter and ensure that it is set to **false**.
 - c. Save and exit the file.
6. Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the **privateZone** and **publicZone** sections from the **<installation_directory>/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 this section completely.

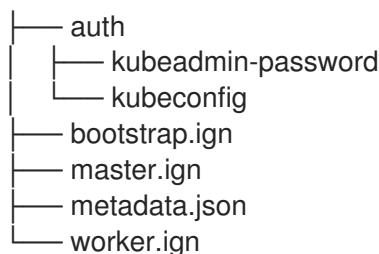
If you do so, you must add ingress DNS records manually in a later step.

7. To create the Ignition configuration files, run the following command from the directory that contains the installation program:

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

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

Ignition config files are created for the bootstrap, control plane, and compute nodes in the installation directory. The **kubeadmin-password** and **kubeconfig** files are created in the **./<installation_directory>/auth** directory:



Additional resources

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

12.7. EXPORTING COMMON VARIABLES

12.7.1. Extracting the infrastructure name

The Ignition config files contain a unique cluster identifier that you can use to uniquely identify your cluster in Google Cloud Platform (GCP). The infrastructure name is also used to locate the appropriate GCP resources during an OpenShift Container Platform installation. The provided Deployment Manager templates contain references to this infrastructure name, so you must extract it.

Prerequisites

- You installed 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
```

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

Example output

```
openshift-vw9j6 ①
```

- ① The output of this command is your cluster name and a random string.

12.7.2. 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.

Procedure

1. 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='10.0.0.0/17'
$ export WORKER_SUBNET_CIDR='10.0.128.0/17'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig ①
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json` 
$ export INFRA_ID=`jq -r .infrайд <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.

12.8. 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

- You have defined the variables in the *Exporting common variables* section.

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-central1**.
- 3 **master_subnet_cidr** is the CIDR for the master subnet, for example **10.0.0.0/17**.
- 4 **worker_subnet_cidr** is the CIDR for the worker subnet, for example **10.0.128.0/17**.

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

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

12.8.1. 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:

Example 12.25. 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'] + '-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': 'AUTO_ONLY',
                        'minPortsPerVm': 7168,
                        'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                        'subnetworks': [

```

```

        'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
        'sourceIpRangesToNat': ['ALL_IP_RANGES']
    }]
}, {
    'name': context.properties['infra_id'] + '-nat-worker',
    'natIpAllocateOption': 'AUTO_ONLY',
    'minPortsPerVm': 512,
    'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
    'subnetworks': [
        {
            'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
            'sourceIpRangesToNat': ['ALL_IP_RANGES']
        }
    ]
}
]

return {'resources': resources}

```

12.9. NETWORKING REQUIREMENTS FOR USER-PROVISIONED INFRASTRUCTURE

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

12.9.1. Setting the cluster node hostnames through DHCP

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

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

12.9.2. Network connectivity requirements

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

This section provides details about the ports that are required.

Table 12.7. Ports used for all-machine to all-machine communications

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	1936	Metrics

Protocol	Port	Description
	9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .
	10250-10259	The default ports that Kubernetes reserves
UDP	4789	VXLAN
	6081	Geneve
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
	500	IPsec IKE packets
	4500	IPsec NAT-T packets
	123	Network Time Protocol (NTP) on UDP port 123 If an external NTP time server is configured, you must open UDP port 123 .
TCP/UDP	30000-32767	Kubernetes node port
ESP	N/A	IPsec Encapsulating Security Payload (ESP)

Table 12.8. Ports used for all-machine to control plane communications

Protocol	Port	Description
TCP	6443	Kubernetes API

Table 12.9. Ports used for control plane machine to control plane machine communications

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

12.10. CREATING LOAD BALANCERS IN GCP

You must configure load balancers 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

- You have defined the variables in the *Exporting common variables* section.

Procedure

1. Copy the template from the **Deployment Manager template for the internal load balancer** section of this topic and save it as **02_lb_int.py** on your computer. This template describes the internal load balancing objects that your cluster requires.
2. For an external cluster, also copy the template from the **Deployment Manager template for the external load balancer** section of this topic and save it as **02_lb_ext.py** on your computer. This template describes the external load balancing objects that your cluster requires.
3. Export the variables that the deployment template uses:
 - a. Export the cluster network location:


```
$ export CLUSTER_NETWORK=(`gcloud compute networks describe ${INFRA_ID}-network --format json | jq -r .selfLink`)
```
 - b. Export the control plane subnet location:


```
$ export CONTROL_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-master-subnet --region=${REGION} --format json | jq -r .selfLink`)
```
 - c. Export the three zones that the cluster uses:


```
$ 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`)
```
4. Create a **02_infra.yaml** resource definition file:


```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py ①
resources:
- name: cluster-lb-ext ②
  type: 02_lb_ext.py
EOF
```

```

properties:
  infra_id: '${INFRA_ID}' 3
  region: '${REGION}' 4
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' 5
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: 6
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF

```

1 **2** Required only when deploying an external cluster.

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

4 **region** is the region to deploy the cluster into, for example **us-central1**.

5 **control_subnet** is the URI to the control subnet.

6 **zones** are the zones to deploy the control plane instances into, like **us-east1-b, us-east1-c, and us-east1-d**.

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

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

6. Export the cluster IP address:

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

7. For an external cluster, also export the cluster public IP address:

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

12.10.1. Deployment Manager template for the external load balancer

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

Example 12.26. 02_lb_ext.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']
}
}, {
    # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
    # probe for kube-apiserver
    'name': context.properties['infra_id'] + '-api-http-health-check',
    'type': 'compute.v1.httpHealthCheck',
    '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'
    }
}]
}

return {'resources': resources}

```

12.10.2. Deployment Manager template for the internal load balancer

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

Example 12.27. 02_lb_int.py Deployment Manager template

```

def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '$(ref.' + context.properties['infra_id'] + '-master-' + zone + '-ig' + '.selfLink)'
        })

    resources = [
        {
            'name': context.properties['infra_id'] + '-cluster-ip',
            'type': 'compute.v1.address',
            'properties': {
                'addressType': 'INTERNAL',
                'region': context.properties['region'],

```

```

'subnetwork': context.properties['control_subnet']
}
},
# Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
{
  'name': context.properties['infra_id'] + '-api-internal-health-check',
  'type': 'compute.v1.healthCheck',
  'properties': {
    'httpsHealthCheck': {
      'port': 6443,
      'requestPath': '/readyz'
    },
    'type': "HTTPS"
  }
},
{
  'name': context.properties['infra_id'] + '-api-internal',
  'type': 'compute.v1.regionBackendService',
  'properties': {
    'backends': backends,
    'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-internal-health-
check.selfLink)'],
    'loadBalancingScheme': 'INTERNAL',
    'region': context.properties['region'],
    'protocol': 'TCP',
    'timeoutSec': 120
  }
},
{
  'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
  'type': 'compute.v1.forwardingRule',
  'properties': {
    'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal.selfLink)',
    'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
    'loadBalancingScheme': 'INTERNAL',
    'ports': ['6443','22623'],
    'region': context.properties['region'],
    'subnetwork': context.properties['control_subnet']
  }
}
]

for zone in context.properties['zones']:
  resources.append({
    'name': context.properties['infra_id'] + '-master-' + zone + '-ig',
    'type': 'compute.v1.instanceGroup',
    'properties': {
      'namedPorts': [
        {
          'name': 'ignition',
          'port': 22623
        },
        {
          'name': 'https',
          'port': 6443
        }
      ],
      'network': context.properties['cluster_network'],
      'zone': zone
    }
  })

```

```

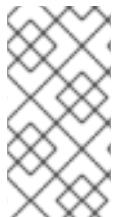
    })
return {'resources': resources}

```

You will need this template in addition to the **02_lb_ext.py** template when you create an external cluster.

12.11. CREATING A PRIVATE DNS ZONE IN GCP

You must configure a private DNS zone in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create this component 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

- Ensure you defined the variables in the *Exporting common variables* and *Creating load balancers in GCP* sections.

Procedure

1. Copy the template from the **Deployment Manager template for the private DNS** section of this topic and save it as **02_dns.py** on your computer. This template describes the private DNS objects that your cluster requires.
2. Create a **02_dns.yaml** resource definition file:

```

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

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' 1
    cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' 2
    cluster_network: '${CLUSTER_NETWORK}' 3
EOF

```

- 1** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 2** **cluster_domain** is the domain for the cluster, for example **openshift.example.com**.
- 3** **cluster_network** is the **selfLink** URL to the cluster network.

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

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml
```

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

- a. Add the 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
```

- b. For an external cluster, also add the 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 ${CLUSTER_PUBLIC_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
```

12.11.1. Deployment Manager template for the private DNS

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

Example 12.28. 02_dns.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [
        {
            '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}
```

12.12. CREATING FIREWALL RULES IN GCP

You must create firewall rules 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

- Ensure you defined the variables in the *Exporting common variables* and *Creating load balancers in GCP* sections.

Procedure

1. Copy the template from the **Deployment Manager template for firewall rules** section of this topic and save it as **03_firewall.py** on your computer. This template describes the security groups that your cluster requires.
2. Create a **03_firewall.yaml** resource definition file:

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

resources:
- name: cluster-firewall
  type: 03_firewall.py
  properties:
    allowed_external_cidr: '0.0.0.0/0' 1
    infra_id: '${INFRA_ID}' 2
    cluster_network: '${CLUSTER_NETWORK}' 3
    network_cidr: '${NETWORK_CIDR}' 4
EOF
```

- 1** **allowed_external_cidr** is the CIDR range that can access the cluster API and SSH to the bootstrap host. For an internal cluster, set this value to **\${NETWORK_CIDR}**.
- 2** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 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**.

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

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml
```

12.12.1. Deployment Manager template for firewall rules

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

Example 12.29. 03_firewall.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [
        {
            'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
            'type': 'compute.v1.firewall',
            'properties': {
                'network': context.properties['cluster_network'],
                'allowed': [
                    {
                        'IPProtocol': 'tcp',
                        'ports': ['22']
                    },
                    {
                        'sourceRanges': [context.properties['allowed_external_cidr']],
                        'targetTags': [context.properties['infra_id'] + '-bootstrap']
                    }
                ],
                'sourceRanges': [context.properties['allowed_external_cidr']],
                'targetTags': [context.properties['infra_id'] + '-bootstrap']
            }
        },
        {
            'name': context.properties['infra_id'] + '-api',
            'type': 'compute.v1.firewall',
            'properties': {
                'network': context.properties['cluster_network'],
                'allowed': [
                    {
                        'IPProtocol': 'tcp',
                        'ports': ['6443']
                    },
                    {
                        'sourceRanges': [context.properties['allowed_external_cidr']],
                        'targetTags': [context.properties['infra_id'] + '-master']
                    }
                ],
                'sourceRanges': [context.properties['allowed_external_cidr']],
                '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', '6443', '22624']
                    },
                    {
                        'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
                        'targetTags': [context.properties['infra_id'] + '-master']
                    }
                ],
                'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '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']
                    }
                ],
                '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']
          },
          {
            'IPProtocol': 'tcp',
            'ports': ['22623']
          }
        ],
        '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': 'udp',
            'ports': ['500', '4500']
          },
          {
            'IPProtocol': 'esp',
          },
          {
            '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'
    ]
}
}]

return {'resources': resources}

```

12.13. CREATING IAM ROLES IN GCP

You must create IAM 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

- You have defined the variables in the *Exporting common variables* section.

Procedure

1. Copy the template from the **Deployment Manager template for IAM roles** section of this topic and save it as **03_iam.py** on your computer. This template describes the IAM roles that your cluster requires.
2. Create a **03_iam.yaml** resource definition file:

```

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

```

```

resources:
- name: cluster-iam
  type: 03_iam.py
  properties:
    infra_id: '${INFRA_ID}' 1
EOF

```

- 1** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

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

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
```

4. Export the variable for the master service account:

```
$ export MASTER_SERVICE_ACCOUNT=`gcloud iam service-accounts list --filter "email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '[0].email'`
```

5. Export the variable for the worker service account:

```
$ export WORKER_SERVICE_ACCOUNT=`gcloud iam service-accounts list --filter "email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '[0].email'`
```

6. Export the variable for the subnet that hosts the compute machines:

```
$ export COMPUTE_SUBNET=`gcloud compute networks subnets describe ${INFRA_ID}-worker-subnet --region=${REGION} --format json | jq -r .selfLink`
```

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

```

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

```

8. 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_SERVICE_ACCOUNT}
```

12.13.1. Deployment Manager template for IAM roles

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

Example 12.30. 03_iam.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [
        {
            '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}
```

12.14. 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 [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>-<arch>-gcp.<arch>.tar.gz**.

2. Create the Google storage bucket:

```
$ gsutil mb gs://<bucket_name>
```

3. Upload the RHCOS image to the Google storage bucket:

```
$ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
gs://<bucket_name>
```

4. Export the uploaded RHCOS image location as a variable:

```
$ export IMAGE_SOURCE=gs://<bucket_name>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
```

5. Create the cluster image:

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

12.15. 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

- Ensure you defined the variables in the *Exporting common variables* and *Creating load balancers in GCP* sections.
- Ensure you installed pyOpenSSL.

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 location of the Red Hat Enterprise Linux CoreOS (RHCOS) image that the installation program requires:

```
$ export CLUSTER_IMAGE=(`gcloud compute images describe ${INFRA_ID}-rhcos-image --
format json | jq -r .selfLink`)
```

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

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

```
$ gsutil cp <installation_directory>/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-central1**.
- 3** **zone** is the zone to deploy the bootstrap instance into, for example **us-central1-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** **root_volume_size** is the boot disk size for the bootstrap machine.
- 9** **bootstrap_ign** is the URL output when creating a signed URL.

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.

- a. Add the bootstrap instance to the internal load balancer instance group:

```
$ gcloud compute instance-groups unmanaged add-instances \
${INFRA_ID}-bootstrap-ig --zone=${ZONE_0} --instances=${INFRA_ID}-bootstrap
```

- b. Add the bootstrap instance group to the internal load balancer backend service:

```
$ gcloud compute backend-services add-backend \
${INFRA_ID}-api-internal --region=${REGION} --instance-group=${INFRA_ID}-
bootstrap-ig --instance-group-zone=${ZONE_0}
```

12.15.1. 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:

Example 12.31. 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',
            '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'] +
                            '"}},"version":"3.2.0"}}'
                        }
                    ],
                    'networkInterfaces': [
                        {
                            'subnetwork': context.properties['control_subnet'],
                            'accessConfigs': [
                                {
                                    'natIP': '$(ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address)'
                                }
                            ]
                        }
                    ]
                }
            }
        }
    ]
    return {'resources': resources}
```

```

        }],
        'tags': {
          'items': [
            context.properties['infra_id'] + '-master',
            context.properties['infra_id'] + '-bootstrap'
          ]
        },
        'zone': context.properties['zone']
      }
    ],
    'name': context.properties['infra_id'] + '-bootstrap-ig',
    'type': 'compute.v1.instanceGroup',
    'properties': {
      'namedPorts': [
        {
          'name': 'ignition',
          'port': 22623
        },
        {
          'name': 'https',
          'port': 6443
        }
      ],
      'network': context.properties['cluster_network'],
      'zone': context.properties['zone']
    }
  }
}

return {'resources': resources}

```

12.16. 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

- Ensure you defined the variables in the *Exporting common variables*, *Creating load balancers in GCP*, *Creating IAM roles in GCP*, and *Creating the bootstrap machine in GCP* sections.
- 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 variable required by the resource definition:

```
$ export MASTER_IGNITION=`cat <installation_directory>/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: '${INFRA_ID}' 1
    zones: 2
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' 3
    image: '${CLUSTER_IMAGE}' 4
    machine_type: 'n1-standard-4' 5
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}' 6

    ignition: '${MASTER_IGNITION}' 7
EOF
```

- 1** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 2** **zones** are the zones to deploy the control plane instances into, for example **us-central1-a**, **us-central1-b**, and **us-central1-c**.
- 3** **control_subnet** is the **selfLink** URL to the control subnet.
- 4** **image** is the **selfLink** URL to the RHCOS image.
- 5** **machine_type** is the machine type of the instance, for example **n1-standard-4**.
- 6** **service_account_email** is the email address for the master service account that you created.
- 7** **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 load balancer membership due to limitations of Deployment Manager, so you must add the control plane machines manually.

- Run the following commands to add the control plane machines to the appropriate instance groups:

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-${ZONE_0}-ig --zone=${ZONE_0} --instances=${INFRA_ID}-master-0
```

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-${ZONE_1}-ig --zone=${ZONE_1} --instances=${INFRA_ID}-master-1
```

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-${ZONE_2}-ig --zone=${ZONE_2} --instances=${INFRA_ID}-master-2
```

- For an external cluster, you must also run the following commands to add the control plane machines to the target pools:

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-zone="${ZONE_0}" --instances=${INFRA_ID}-master-0
```

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-zone="${ZONE_1}" --instances=${INFRA_ID}-master-1
```

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-zone="${ZONE_2}" --instances=${INFRA_ID}-master-2
```

12.16.1. 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:

Example 12.32. 05_control_plane.py Deployment Manager template

```
def GenerateConfig(context):
    resources = [
        {
            'name': context.properties['infra_id'] + '-master-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'] + '-master-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'] + '-master-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}

```

12.17. 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

- Ensure you defined the variables in the *Exporting common variables* and *Creating load balancers in GCP* sections.
- 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 backend-services remove-backend ${INFRA_ID}-api-internal --
region=${REGION} --instance-group=${INFRA_ID}-bootstrap-ig --instance-group-
zone=${ZONE_0}

$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign

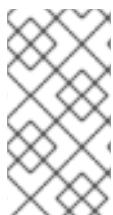
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition

$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

12.18. 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

- Ensure you defined the variables in the *Exporting common variables*, *Creating load balancers in GCP*, and *Creating the bootstrap machine in GCP* sections.
- 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 variables that the resource definition uses.
 - a. Export the subnet that hosts the compute machines:

```
$ export COMPUTE_SUBNET=`gcloud compute networks subnets describe
${INFRA_ID}-worker-subnet --region=${REGION} --format json | jq -r .selfLink`
```

- b. Export the email address for your service account:

```
$ export WORKER_SERVICE_ACCOUNT=`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '[0].email'`
```

- c. Export the location of the compute machine Ignition config file:

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
```

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

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

resources:
- name: 'worker-0' ①
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' ②
    zone: '${ZONE_0}' ③
    compute_subnet: '${COMPUTE_SUBNET}' ④
    image: '${CLUSTER_IMAGE}' ⑤
    machine_type: 'n1-standard-4' ⑥
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' ⑦
    ignition: '${WORKER_IGNITION}' ⑧
- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' ⑨
    zone: '${ZONE_1}' ⑩
    compute_subnet: '${COMPUTE_SUBNET}' ⑪
    image: '${CLUSTER_IMAGE}' ⑫
    machine_type: 'n1-standard-4' ⑬
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' ⑭
    ignition: '${WORKER_IGNITION}' ⑮
EOF
```

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

2 9 `infra_id` is the **INFRA_ID** infrastructure name from the extraction step.

3 10 `zone` is the zone to deploy the worker machine into, for example **us-central1-a**.

4 11 `compute_subnet` is the **selfLink** URL to the compute subnet.

5 12 `image` is the **selfLink** URL to the RHCOS image.¹

6 13 `machine_type` is the machine type of the instance, for example **n1-standard-4**.

7 14 `service_account_email` is the email address for the worker service account that you created.

8 15 `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
```

1. To use a GCP Marketplace image, specify the offer to use:

- OpenShift Container Platform:
<https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-ocp-413-x86-64-202305021736>
- OpenShift Platform Plus: <https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-opp-413-x86-64-202305021736>
- OpenShift Kubernetes Engine:
<https://www.googleapis.com/compute/v1/projects/redhat-marketplace-public/global/images/redhat-coreos-oke-413-x86-64-202305021736>

12.18.1. 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:

Example 12.33. 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}

```

12.19. LOGGING IN TO THE CLUSTER BY USING THE CLI

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

- You installed the **oc** CLI.
- Ensure the bootstrap process completed successfully.

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
```

Example output

```
system:admin
```

12.20. DISABLING THE DEFAULT OPERATORHUB CATALOG SOURCES

Operator catalogs that source content provided by Red Hat and community projects are configured for OperatorHub by default during an OpenShift Container Platform installation. In a restricted network environment, you must disable the default catalogs as a cluster administrator.

Procedure

- Disable the sources for the default catalogs by adding **disableAllDefaultSources: true** to the **OperatorHub** object:

```
$ oc patch OperatorHub cluster --type json \
-p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

TIP

Alternatively, you can use the web console to manage catalog sources. From the **Administration → Cluster Settings → Configuration → OperatorHub** page, click the **Sources** tab, where you can create, update, delete, disable, and enable individual sources.

12.21. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

- Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	63m	v1.31.3
master-1	Ready	master	63m	v1.31.3
master-2	Ready	master	64m	v1.31.3

The output lists all of the machines that you created.



NOTE

The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

- Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

NAME	AGE	REQUESTOR	CONDITION
csr-8b2br	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
csr-8vnps	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
...			

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 the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.



NOTE

For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- 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.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}\n{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



NOTE

Some Operators might not become available until some CSRs are approved.

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

NAME	AGE	REQUESTOR	CONDITION
csr-bfd72	5m26s	system:node:ip-10-0-50-126.us-east-2.compute.internal	Pending
csr-c57lv	5m26s	system:node:ip-10-0-95-157.us-east-2.compute.internal	Pending
...			

- If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- 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.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}\n{{end}}{{end}}' | xargs oc adm certificate approve
```

- After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	73m	v1.31.3
master-1	Ready	master	73m	v1.31.3

```
master-2 Ready master 74m v1.31.3
worker-0 Ready worker 11m v1.31.3
worker-1 Ready worker 11m v1.31.3
```

**NOTE**

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- [Certificate Signing Requests](#)

12.22. 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

- Ensure you defined the variables in the *Exporting common variables* section.
- Remove the DNS Zone configuration when creating Kubernetes manifests and generating Ignition configs.
- Ensure the bootstrap process completed successfully.

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
```

Example output

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 zones:

- To use A records:

- i. Export the variable for the router IP address:

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

- ii. Add the A record to the private zones:

```
$ 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
```

- iii. For an external cluster, also add the A record to the public zones:

```
$ 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}
```

- To add explicit domains instead of using a wildcard, 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
```

Example output

```
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
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

12.23. 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

- Ensure the bootstrap process completed successfully.

Procedure

1. Complete the cluster installation:

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

Example output

```
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, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

2. Observe the running state of your cluster.

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

```
$ oc get clusterversion
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	SINCE	STATUS
version	False	True	24m	Working towards 4.5.4: 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
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED
SINCE				
authentication	4.5.4	True	False	False 7m56s
cloud-credential	4.5.4	True	False	False 31m
cluster-autoscaler	4.5.4	True	False	False 16m
console	4.5.4	True	False	False 10m
csi-snapshot-controller	4.5.4	True	False	False 16m
dns	4.5.4	True	False	False 22m
etcd	4.5.4	False	False	False 25s
image-registry	4.5.4	True	False	False 16m
ingress	4.5.4	True	False	False 16m
insights	4.5.4	True	False	False 17m
kube-apiserver	4.5.4	True	False	False 19m
kube-controller-manager	4.5.4	True	False	False 20m
kube-scheduler	4.5.4	True	False	False 20m
kube-storage-version-migrator	4.5.4	True	False	False 16m
machine-api	4.5.4	True	False	False 22m
machine-config	4.5.4	True	False	False 22m
marketplace	4.5.4	True	False	False 16m

monitoring	4.5.4	True	False	False	10m
network	4.5.4	True	False	False	23m
node-tuning	4.5.4	True	False	False	23m
openshift-apiserver	4.5.4	True	False	False	17m
openshift-controller-manager	4.5.4	True	False	False	15m
openshift-samples	4.5.4	True	False	False	16m
operator-lifecycle-manager	4.5.4	True	False	False	22m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	22m
operator-lifecycle-manager-packageserver	4.5.4	True	False	False	18m
service-ca	4.5.4	True	False	False	23m
service-catalog-apiserver	4.5.4	True	False	False	23m
service-catalog-controller-manager	4.5.4	True	False	False	23m
storage	4.5.4	True	False	False	17m

c. Run the following command to view your cluster pods:

```
$ oc get pods --all-namespaces
```

Example output

NAMESPACE	READY	STATUS	RESTARTS	AGE	NAME
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					openshift-apiserver-operator-6d6674f4f4-
h7t2t	1/1	Running	1	37m	
openshift-apiserver					apiserver-fm48r
1/1	Running	0	30m		
openshift-apiserver					apiserver-fxkvv
1/1	Running	0	29m		
openshift-apiserver					apiserver-q85nm
1/1	Running	0	29m		
...					
openshift-service-ca-operator					openshift-service-ca-operator-66ff6dc6cd-
9r257	1/1	Running	0	37m	
openshift-service-ca					apiservice-cabundle-injector-695b6bcfc-cl5hm
1/1	Running	0	35m		
openshift-service-ca					configmap-cabundle-injector-8498544d7-
25qn6	1/1	Running	0	35m	
openshift-service-ca					service-serving-cert-signer-6445fc9c6-wqdqn
1/1	Running	0	35m		
openshift-service-catalog-apiserver-operator					openshift-service-catalog-apiserver-
operator-549f44668b-b5q2w	1/1	Running	0	32m	
openshift-service-catalog-controller-manager-operator					openshift-service-catalog-
controller-manager-operator-b78cr2lnm	1/1	Running	0	31m	

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

12.24. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.18, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to [OpenShift Cluster Manager](#).

After you confirm that your [OpenShift Cluster Manager](#) inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See [About remote health monitoring](#) for more information about the Telemetry service

12.25. NEXT STEPS

- [Customize your cluster.](#)
- [Configure image streams](#) for the Cluster Samples Operator and the **must-gather** tool.
- Learn how to [Use Operator Lifecycle Manager in disconnected environments](#) .
- If the mirror registry that you used to install your cluster has a trusted CA, add it to the cluster by [configuring additional trust stores](#).
- If necessary, you can [opt out of remote health reporting](#) .
- If necessary, see [Registering your disconnected cluster](#)

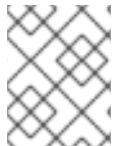
CHAPTER 13. INSTALLING A THREE-NODE CLUSTER ON GCP

In OpenShift Container Platform version 4.18, you can install a three-node cluster on Google Cloud Platform (GCP). A three-node cluster consists of three control plane machines, which also act as compute machines. This type of cluster provides a smaller, more resource efficient cluster, for cluster administrators and developers to use for testing, development, and production.

You can install a three-node cluster using either installer-provisioned or user-provisioned infrastructure.

13.1. CONFIGURING A THREE-NODE CLUSTER

You configure a three-node cluster by setting the number of worker nodes to **0** in the **install-config.yaml** file before deploying the cluster. Setting the number of worker nodes to **0** ensures that the control plane machines are schedulable. This allows application workloads to be scheduled to run from the control plane nodes.



NOTE

Because application workloads run from control plane nodes, additional subscriptions are required, as the control plane nodes are considered to be compute nodes.

Prerequisites

- You have an existing **install-config.yaml** file.

Procedure

1. Set the number of compute replicas to **0** in your **install-config.yaml** file, as shown in the following **compute** stanza:

Example **install-config.yaml** file for a three-node cluster

```
apiVersion: v1
baseDomain: example.com
compute:
- name: worker
  platform: {}
  replicas: 0
# ...
```

2. If you are deploying a cluster with user-provisioned infrastructure:

- After you create the Kubernetes manifest files, make sure that the **spec.mastersSchedulable** parameter is set to **true** in **cluster-scheduler-02-config.yml** file. You can locate this file in **<installation_directory>/manifests**. For more information, see "Creating the Kubernetes manifest and Ignition config files" in "Installing a cluster on user-provisioned infrastructure in GCP by using Deployment Manager templates".
- Do not create additional worker nodes.

Example **cluster-scheduler-02-config.yml** file for a three-node cluster

```
apiVersion: config.openshift.io/v1
kind: Scheduler
```

```
metadata:  
  creationTimestamp: null  
  name: cluster  
spec:  
  mastersSchedulable: true  
  policy:  
    name: ""  
status: {}
```

13.2. NEXT STEPS

- [Installing a cluster on GCP with customizations](#)
- [Installing a cluster on user-provisioned infrastructure in GCP by using Deployment Manager templates](#)

CHAPTER 14. INSTALLATION CONFIGURATION PARAMETERS FOR GCP

Before you deploy an OpenShift Container Platform cluster on Google Cloud Platform (GCP), you provide parameters to customize your cluster and the platform that hosts it. When you create the **install-config.yaml** file, you provide values for the required parameters through the command line. You can then modify the **install-config.yaml** file to customize your cluster further.

14.1. AVAILABLE INSTALLATION CONFIGURATION PARAMETERS FOR GCP

The following tables specify the required, optional, and GCP-specific installation configuration parameters that you can set as part of the installation process.



NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.

14.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 14.1. Required parameters

Parameter	Description	Values
apiVersion:	The API version for the install-config.yaml content. The current version is v1 . The installation program may also support older API versions.	String
baseDomain:	The base domain of your cloud provider. The base domain 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 .
metadata:	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object

Parameter	Description	Values
metadata: name:	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}}. {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform:	The configuration for the specific platform upon which to perform the installation: aws , baremetal , azure , gcp , ibmcloud , nutanix , openstack , powervs , vsphere , or {} . For additional information about platform . < platform > parameters, consult the table for your specific platform that follows.	Object
pullSecret:	Get a pull secret from Red Hat OpenShift Cluster Manager to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths": { "cloud.openshift.com": { "auth": "b3Blb=", "email": "you@example.com" }, "quay.io": { "auth": "b3Blb=", "email": "you@example.com" } } }</pre>

14.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 14.2. Network parameters

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

Parameter	Description	Values
networking:	The configuration for the cluster network.	<p>Object</p>  <p>NOTE</p> <p>You cannot modify parameters specified by the networking object after installation.</p>
networking: networkType:	The Red Hat OpenShift Networking network plugin to install.	OVNKubernetes. OVNKubernetes is a CNI plugin for Linux networks and hybrid networks that contain both Linux and Windows servers. The default value is OVNKubernetes .
networking: clusterNetwork:	<p>The IP address blocks for pods.</p> <p>The default value is 10.128.0.0/14 with a host prefix of /23.</p> <p>If you specify multiple IP address blocks, the blocks must not overlap.</p>	<p>An array of objects. For example:</p> <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>
networking: clusterNetwork: cidr:	<p>Required if you use networking.clusterNetwork. An IP address block.</p> <p>An IPv4 network.</p>	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking: clusterNetwork: hostPrefix:	<p>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. A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.</p>	<p>A subnet prefix.</p> <p>The default value is 23.</p>
networking: serviceNetwork:	<p>The IP address block for services. The default value is 172.30.0.0/16.</p> <p>The OVN-Kubernetes network plugins supports only a single IP address block for the service network.</p>	<p>An array with an IP address block in CIDR format. For example:</p> <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>

Parameter	Description	Values
networking: machineNetwork:	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: networking: machineNetwork: - cidr: 10.0.0.0/16
networking: machineNetwork: cidr:	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt and IBM Power® Virtual Server. For libvirt, the default value is 192.168.126.0/24 . For IBM Power® Virtual Server, the default value is 192.168.0.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  NOTE Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.
networking: ovnKubernetesConfig: ipv4: internalJoinSubnet:	Configures the IPv4 join subnet that is used internally by ovn-kubernetes . This subnet must not overlap with any other subnet that OpenShift Container Platform is using, including the node network. The size of the subnet must be larger than the number of nodes. You cannot change the value after installation.	An IP network block in CIDR notation. The default value is 100.64.0.0/16 .

14.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 14.3. Optional parameters

Parameter	Description	Values
additionalTrustBundle:	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String

Parameter	Description	Values
capabilities:	Controls the installation of optional core cluster components. You can reduce the footprint of your OpenShift Container Platform cluster by disabling optional components. For more information, see the "Cluster capabilities" page in <i>Installing</i> .	String array
capabilities: baselineCapabilitySet:	Selects an initial set of optional capabilities to enable. Valid values are None , v4.11 , v4.12 and vCurrent . The default value is vCurrent .	String
capabilities: additionalEnabledCapabilities:	Extends the set of optional capabilities beyond what you specify in baselineCapabilitySet . You may specify multiple capabilities in this parameter.	String array
cpuPartitioningMode:	Enables workload partitioning, which isolates OpenShift Container Platform services, cluster management workloads, and infrastructure pods to run on a reserved set of CPUs. Workload partitioning can only be enabled during installation and cannot be disabled after installation. While this field enables workload partitioning, it does not configure workloads to use specific CPUs. For more information, see the <i>Workload partitioning</i> page in the <i>Scalability and Performance</i> section.	None or AllNodes . None is the default value.
compute:	The configuration for the machines that comprise the compute nodes.	Array of MachinePool objects.
compute: architecture:	Determines the instruction set architecture of the machines in the pool. Currently, clusters with varied architectures are not supported. All pools must specify the same architecture. Valid values are amd64 and arm64 .	String

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: name:	Required if you use compute . The name of the machine pool.	worker
compute: platform:	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws , azure , gcp , ibmcloud , nutanix , openstack , powervs , vsphere , or {}
compute: replicas:	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
featureSet:	Enables the cluster for a feature set. A feature set is a collection of OpenShift Container Platform features that are not enabled by default. For more information about enabling a feature set during installation, see "Enabling features using feature gates".	String. The name of the feature set to enable, such as TechPreviewNoUpgrade .
controlPlane:	The configuration for the machines that comprise the control plane.	Array of MachinePool objects.

Parameter	Description	Values
controlPlane: architecture:	Determines the instruction set architecture of the machines in the pool. Currently, clusters with varied architectures are not supported. All pools must specify the same architecture. Valid values are amd64 and arm64 .	String
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: name:	Required if you use controlPlane . The name of the machine pool.	master
controlPlane: platform:	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, ibmcloud, nutanix, openstack, powervs, vsphere, or {}
controlPlane: replicas:	The number of control plane machines to provision.	Supported values are 3 , or 1 when deploying single-node OpenShift.

Parameter	Description	Values
credentialsMode:	<p>The Cloud Credential Operator (CCO) mode. If no mode is specified, the CCO dynamically tries to determine the capabilities of the provided credentials, with a preference for mint mode on the platforms where multiple modes are supported.</p>  <p>NOTE</p> <p>Not all CCO modes are supported for all cloud providers. For more information about CCO modes, see the "Managing cloud provider credentials" entry in the <i>Authentication and authorization</i> content.</p>	Mint, Passthrough, Manual or an empty string ("").

Parameter	Description	Values
<p><code>fips:</code></p> 	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p> <p>IMPORTANT</p> <p>To enable FIPS mode for your cluster, you must run the installation program from a Red Hat Enterprise Linux (RHEL) computer configured to operate in FIPS mode. For more information about configuring FIPS mode on RHEL, see Switching RHEL to FIPS mode.</p> <p>When running Red Hat Enterprise Linux (RHEL) or Red Hat Enterprise Linux CoreOS (RHCOS) booted in FIPS mode, OpenShift Container Platform core components use the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the x86_64, ppc64le, and s390x architectures.</p>	<p>false or true</p>
	<p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p>	

Parameter	Description	Values
imageContentSources:	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources: source:	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources: mirrors:	Specify one or more repositories that may also contain the same images.	Array of strings
publish:	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .
sshKey:	The SSH key to authenticate access to your cluster machines.	For example, sshKey: ssh-ed25519 AAAA...



NOTE

If you are installing on GCP into a shared virtual private cloud (VPC), **credentialsMode** must be set to **Passthrough** or **Manual**.



IMPORTANT

Setting this parameter to **Manual** enables alternatives to storing administrator-level secrets in the **kube-system** project, which require additional configuration steps. For more information, see "Alternatives to storing administrator-level secrets in the kube-system project".

14.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 14.4. Additional GCP parameters

Parameter	Description	Values
controlPlane.platform.gcp.osImage.project:	Optional. By default, the installation program downloads and installs the Red Hat Enterprise Linux CoreOS (RHCOS) image that is used to boot control plane machines. You can override the default behavior by specifying the location of a custom RHCOS image that the installation program is to use for control plane machines only. Control plane machines do not contribute to licensing costs when using the default image, but if you apply a GCP Marketplace image for a control plane machine, usage costs will apply.	String. The name of GCP project where the image is located.
controlPlane.platform.gcp.osImage.name:	The name of the custom RHCOS image that the installation program is to use to boot control plane machines. If you use controlPlane.platform.gcp.osImage.project , this field is required.	String. The name of the RHCOS image.

Parameter	Description	Values
compute: platform: gcp: osImage: project:	Optional. By default, the installation program downloads and installs the RHCOS image that is used to boot compute machines. You can override the default behavior by specifying the location of a custom RHCOS image that the installation program is to use for compute machines only.	String. The name of GCP project where the image is located.
compute: platform: gcp: osImage: name:	The name of the custom RHCOS image that the installation program is to use to boot compute machines. If you use compute.platform.gcp.osImage.project , this field is required.	String. The name of the RHCOS image.

Parameter	Description	Values
compute: platform: gcp: serviceAccount:	Specifies the email address of a GCP service account to be used during installations. This service account will be used to provision compute machines.	String. The email address of the service account.
platform: gcp: network:	The name of the existing Virtual Private Cloud (VPC) where you want to deploy your cluster. If you want to deploy your cluster into a shared VPC, you must set platform.gcp.networkProjectID with the name of the GCP project that contains the shared VPC.	String.
platform: gcp: networkProjectID:	Optional. The name of the GCP project that contains the shared VPC where you want to deploy your cluster.	String.

Parameter	Description	Values
platfor m: gcp : proj ectl D:	The name of the GCP project where the installation program installs the cluster.	String.
platfor m: gcp : regi on:	The name of the GCP region that hosts your cluster.	Any valid region name, such as us-central1 .
platfor m: gcp : con trol Pla ne Su bne t:	The name of the existing subnet where you want to deploy your control plane machines.	The subnet name.

Parameter	Description	Values
platfor m: gcp : co mp ute Su bne t:	The name of the existing subnet where you want to deploy your compute machines.	The subnet name.
platfor m: gcp : def ault Ma chi ne Plat for m: zon es:	The availability zones where the installation program creates machines.	<p>A list of valid GCP availability zones, such as us-central1-a, in a YAML sequence.</p> <div style="display: flex; align-items: center;">  IMPORTANT <p>When running your cluster on GCP 64-bit ARM infrastructure, ensure that you use a zone where Ampere Altra Arm CPU's are available. You can find which zones are compatible with 64-bit ARM processors in the "GCP availability zones" link.</p> </div>

Parameter	Description	Values
platfor m: gcp : def ault Ma chi ne Plat for m: os Dis k: disk Siz eG B:	The size of the disk in gigabytes (GB).	Any size between 16 GB and 65536 GB.

Parameter	Description	Values
plat for m: gcp : def ault Ma chi ne Plat for m: os Dis k: disk Typ e:	<p>The GCP disk type.</p>	<p>The default disk type for all machines. Valid values are pd-balanced, pd-ssd, pd-standard, or hyperdisk-balanced. The default value is pd-ssd. Control plane machines cannot use the pd-standard disk type, so if you specify pd-standard as the default machine platform disk type, you must specify a different disk type using the controlPlane.platform.gcp.osDisk.diskType parameter.</p>
plat for m: gcp : def ault Ma chi ne Plat for m: osl ma ge: proj ect:	<p>Optional. By default, the installation program downloads and installs the RHCOS image that is used to boot control plane and compute machines. You can override the default behavior by specifying the location of a custom RHCOS image that the installation program is to use for both types of machines.</p>	<p>String. The name of GCP project where the image is located.</p>

Parameter	Description	Values
plat for m: gcp : def ault Ma chi ne Plat for m: osl ma ge: na me:	<p>The name of the custom RHCOS image that the installation program is to use to boot control plane and compute machines.</p> <p>If you use platform.gcp.defaultMachinePlatform.osImage.project, this field is required.</p>	String. The name of the RHCOS image.
plat for m: gcp : def ault Ma chi ne Plat for m: tag s:	<p>Optional. Additional network tags to add to the control plane and compute machines.</p>	One or more strings, for example network-tag1 .

Parameter	Description	Values
platfor m: gcp : defaul tMa chine Platfor m: typ e:	The GCP machine type for control plane and compute machines.	The GCP machine type, for example n1-standard-4 .

Parameter	Description	Values
plat for m: gcp : def ault Ma chi ne Plat for m: os Dis k: enc rypt ion Key : km sKe y: na me:	The name of the customer managed encryption key to be used for machine disk encryption.	The encryption key name.

Parameter	Description	Values
platform: gcp: defaultMachinePlatform: osDisk: encryptionKey: kmssKey: keyRing:	The name of the Key Management Service (KMS) key ring to which the KMS key belongs.	The KMS key ring name.

Parameter	Description	Values
platfor m: gcp : defaul tMa chi ne Plat for m:	The GCP location in which the KMS key ring exists.	The GCP location.
osDis k:		
encrypt ion Key :		
km sKe y:		
locatio n:		

Parameter	Description	Values
platform: gcp: defaultMachinePlatform: osDisk: encryptionKey: kmssKey: projectID:	The ID of the project in which the KMS key ring exists. This value defaults to the value of the platform.gcp.projectID parameter if it is not set.	The GCP project ID.

Parameter	Description	Values
platfor m: gcp : defaul tMa chine Platfor m: os Dis k: enc rypt ion Key : km sKe ySe rvic eAc cou nt:	The GCP service account used for the encryption request for control plane and compute machines. If absent, the Compute Engine default service account is used. For more information about GCP service accounts, see Google's documentation on service accounts .	The GCP service account email, for example <code><service_account_name>@<project_id>.iam.gserviceaccount.com</code> .

Parameter	Description	Values
platfor m: gcp : defau ltMa chi ne Plat for m: sec ure Bo ot:	<p>Whether to enable Shielded VM secure boot for all machines in the cluster. Shielded VMs have additional security protocols such as secure boot, firmware and integrity monitoring, and rootkit protection. For more information on Shielded VMs, see Google's documentation on Shielded VMs.</p>	Enabled or Disabled . The default value is Disabled .
platfor m: gcp : defau ltMa chi ne Plat for m: confide ntia lCo mp ute:	<p>Whether to use Confidential VMs for all machines in the cluster. Confidential VMs provide encryption for data during processing. For more information on Confidential computing, see Google's documentation on Confidential computing.</p>	Enabled or Disabled . The default value is Disabled .

Parameter	Description	Values
platfor m:	Specifies the behavior of all VMs during a host maintenance event, such as a software or hardware update. For Confidential VMs, this parameter must be set to Terminate . Confidential VMs do not support live VM migration.	Terminate or Migrate . The default value is Migrate .
gcp: :		
defaultMa chi ne Platfor m:		
onHo stM ainten ance: :		

Parameter	Description	Values
controlPlane:	The name of the customer managed encryption key to be used for control plane machine disk encryption.	The encryption key name.
platform:		
gcp:		
osDisk:		
encryptionKey:		
kmssKey:		
name:		

Parameter	Description	Values
controlPlane:	For control plane machines, the name of the KMS key ring to which the KMS key belongs.	The KMS key ring name.
platform:		
gcp:		
osDisk:		
encryptionKey:		
kmssKey:		
keyRing:		

Parameter	Description	Values
controlPlane:	For control plane machines, the GCP location in which the key ring exists. For more information about KMS locations, see Google's documentation on Cloud KMS locations .	The GCP location for the key ring.
platform:		
gcp:		
osDisk:		
encryptionKey:		
kmssKey:		
location:		

Parameter	Description	Values
controlPlane:	For control plane machines, the ID of the project in which the KMS key ring exists. This value defaults to the VM project ID if not set.	The GCP project ID.
platform:		
gcp:		
osDisk:		
encryptionKey:		
kmssKey:		
projectID:		

Parameter	Description	Values
controlPlane:	The GCP service account used for the encryption request for control plane machines. If absent, the Compute Engine default service account is used. For more information about GCP service accounts, see Google's documentation on service accounts .	The GCP service account email, for example <code><service_account_name>@<project_id>.iam.gserviceaccount.com</code> .
platform:		
gcp:		
osDisk:		
encryptionKey:		
kmssKeyServiceAccount:		

Parameter	Description	Values
controlPlane: platform: gcp: osDisk: diskSizeGB:	The size of the disk in gigabytes (GB). This value applies to control plane machines.	Any integer between 16 and 65536.
controlPlane: platform: gcp: osDisk: diskType:	The GCP disk type for control plane machines.	Valid values are pd-balanced , pd-ssd , or hyperdisk-balanced . The default value is pd-ssd .

Parameter	Description	Values
controlPlane: platform: gcp: tags:	<p>Optional. Additional network tags to add to the control plane machines. If set, this parameter overrides the platform.gcp.defaultMachinePlatform.tags parameter for control plane machines.</p>	One or more strings, for example control-plane-tag1 .
controlPlane: platform: gcp: type:	<p>The GCP machine type for control plane machines. If set, this parameter overrides the platform.gcp.defaultMachinePlatform.type parameter.</p>	The GCP machine type, for example n1-standard-4 .

Parameter	Description	Values
controlPlane: platform: gcp: zones:	The availability zones where the installation program creates control plane machines.	A list of valid GCP availability zones , such as us-central1-a , in a YAML sequence .  <p>IMPORTANT</p> <p>When running your cluster on GCP 64-bit ARM infrastructures, ensure that you use a zone where Ampere Altra Arm CPU's are available. You can find which zones are compatible with 64-bit ARM processors in the "GCP availability zones" link.</p>
controlPlane: platform: gcp: secureBoot:	Whether to enable Shielded VM secure boot for control plane machines. Shielded VMs have additional security protocols such as secure boot, firmware and integrity monitoring, and rootkit protection. For more information on Shielded VMs, see Google's documentation on Shielded VMs .	Enabled or Disabled . The default value is Disabled .

Parameter	Description	Values
controlPlane: platform: gcp: confidentialCompute:	<p>Whether to enable Confidential VMs for control plane machines. Confidential VMs provide encryption for data while it is being processed. For more information on Confidential VMs, see Google's documentation on Confidential Computing.</p>	Enabled or Disabled . The default value is Disabled .
controlPlane: platform: gcp: onHostMaintenance:	<p>Specifies the behavior of control plane VMs during a host maintenance event, such as a software or hardware update. For Confidential VMs, this parameter must be set to Terminate. Confidential VMs do not support live VM migration.</p>	Terminate or Migrate . The default value is Migrate .

Parameter	Description	Values
controlPlane: platform: gcp: serviceAccount:	<p>Specifies the email address of a GCP service account to be used during installations. This service account will be used to provision control plane machines.</p> <p>IMPORTANT</p> <p>In the case of shared VPC installations, when the service account is not provided, the installer service account must have the resourcemanager.projects.getIamPolicy and resourcemanager.projects.setIamPolicy permissions in the host project.</p>	String. The email address of the service account.
compute: platform: gcp: osDisk: encryptionKey: kmSKey: name:	<p>The name of the customer managed encryption key to be used for compute machine disk encryption.</p>	The encryption key name.

Parameter	Description	Values
compute:	For compute machines, the name of the KMS key ring to which the KMS key belongs.	The KMS key ring name.
platform:		
gcp:		
osDisk:		
encryptionKey:		
kmssKey:		
keyRing:		

Parameter	Description	Values
compute:	For compute machines, the GCP location in which the key ring exists. For more information about KMS locations, see Google's documentation on Cloud KMS locations .	The GCP location for the key ring.
platform:		
gcp:		
osDisk:		
encryptionKey:		
kmsKey:		
location:		

Parameter	Description	Values
compute:	For compute machines, the ID of the project in which the KMS key ring exists. This value defaults to the VM project ID if not set.	The GCP project ID.
platform:		
gcp:		
osDisk:		
encryptionKey:		
kmssKey:		
projectID:		

Parameter	Description	Values
co mp ute: plat for m: gcp : os Dis k: enc rypt ion Key : km sKe ySe rvic eAc cou nt:	The GCP service account used for the encryption request for compute machines. If this value is not set, the Compute Engine default service account is used. For more information about GCP service accounts, see Google's documentation on service accounts .	The GCP service account email, for example <code><service_account_name>@<project_id>.iam.gserviceaccount.com</code> .
co mp ute: plat for m: gcp : os Dis k: disk Siz eG B:	The size of the disk in gigabytes (GB). This value applies to compute machines.	Any integer between 16 and 65536.

Parameter	Description	Values
co mp ute: plat for m: gcp : os Dis k: disk Typ e:	<p>The GCP disk type for compute machines.</p>	<p>Valid values are pd-balanced, pd-ssd, pd-standard, or hyperdisk-balanced. The default value is pd-ssd.</p>
co mp ute: plat for m: gcp : tag s:	<p>Optional. Additional network tags to add to the compute machines. If set, this parameter overrides the platform.gcp.defaultMachinePlatform.tags parameter for compute machines.</p>	<p>One or more strings, for example compute-network-tag1.</p>

Parameter	Description	Values
compute: platform: gcp: type:	The GCP machine type for compute machines. If set, this parameter overrides the platform.gcp.defaultMachinePlatform.type parameter.	The GCP machine type, for example n1-standard-4 .
compute: platform: gcp: zones:	The availability zones where the installation program creates compute machines.	<p>A list of valid GCP availability zones, such as us-central1-a, in a YAML sequence.</p> <p>IMPORTANT</p> <p>When running your cluster on GCP 64-bit ARM infrastructure, ensure that you use a zone where Ampere Altra Arm CPU's are available. You can find which zones are compatible with 64-bit ARM processors in the "GCP availability zones" link.</p> 

Parameter	Description	Values
compute: platform: gcp: secureBoot:	Whether to enable Shielded VM secure boot for compute machines. Shielded VMs have additional security protocols such as secure boot, firmware and integrity monitoring, and rootkit protection. For more information on Shielded VMs, see Google's documentation on Shielded VMs .	Enabled or Disabled . The default value is Disabled .
compute: platform: gcp: confidentialCompute:	Whether to enable Confidential VMs for compute machines. Confidential VMs provide encryption for data while it is being processed. For more information on Confidential VMs, see Google's documentation on Confidential Computing .	Enabled or Disabled . The default value is Disabled .

Parameter	Description	Values
compute:	Specifies the behavior of compute VMs during a host maintenance event, such as a software or hardware update. For Confidential VMs, this parameter must be set to Terminate . Confidential VMs do not support live VM migration.	Terminate or Migrate . The default value is Migrate .
platform:		
gcp:		
onHostMaintenance:		

CHAPTER 15. UNINSTALLING A CLUSTER ON GCP

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

15.1. REMOVING A CLUSTER THAT USES INSTALLER-PROVISIONED INFRASTRUCTURE

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



NOTE

After uninstallation, check your cloud provider for any resources not removed properly, especially with User Provisioned Infrastructure (UPI) clusters. There might be resources that the installer did not create or that the installer is unable to access. For example, some Google Cloud resources require [IAM permissions](#) in shared VPC host projects, or there might be unused [health checks that must be deleted](#).

Prerequisites

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

Procedure

1. From the directory that contains the installation program on the computer that you used to install the cluster, run the following command:

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

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



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.

15.2. DELETING GOOGLE CLOUD PLATFORM RESOURCES WITH THE CLOUD CREDENTIAL OPERATOR UTILITY

After uninstalling an OpenShift Container Platform cluster that uses short-term credentials managed outside the cluster, you can use the CCO utility (**ccctl**) to remove the Google Cloud Platform (GCP) resources that **ccctl** created during installation.

Prerequisites

- Extract and prepare the **ccctl** binary.
- Uninstall an OpenShift Container Platform cluster on GCP that uses short-term credentials.

Procedure

- 1 Set a **\$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

- 2 Extract the list of **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
--from=$RELEASE_IMAGE \
--credentials-requests \
--included ① \
--to=<path_to_directory_for_credentials_requests> ②
```

- ① The **--included** parameter includes only the manifests that your specific cluster configuration requires.
- ② Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

- 3 Delete the GCP resources that **ccctl** created by running the following command:

```
$ ccctl gcp delete \
--name=<name> ① \
--project=<gcp_project_id> ② \
--credentials-requests-dir=<path_to_credentials_requests_directory> \
--force-delete-custom-roles ③
```

- ① **<name>** matches the name that was originally used to create and tag the cloud resources.
- ② **<gcp_project_id>** is the GCP project ID in which to delete cloud resources.
- ③ Optional: This parameter deletes the custom roles that the **ccctl** utility creates during installation. GCP does not permanently delete custom roles immediately. For more information, see GCP documentation about [deleting a custom role](#).

Verification

- To verify that the resources are deleted, query GCP. For more information, refer to GCP documentation.

CHAPTER 16. INSTALLING A CLUSTER WITH THE SUPPORT FOR CONFIGURING MULTI-ARCHITECTURE COMPUTE MACHINES

An OpenShift Container Platform cluster with multi-architecture compute machines supports compute machines with different architectures.



NOTE

When you have nodes with multiple architectures in your cluster, the architecture of your image must be consistent with the architecture of the node. You must ensure that the pod is assigned to the node with the appropriate architecture and that it matches the image architecture. For more information on assigning pods to nodes, [Scheduling workloads on clusters with multi-architecture compute machines](#).

You can install a Google Cloud Platform (GCP) cluster with the support for configuring multi-architecture compute machines. After installing the GCP cluster, you can add multi-architecture compute machines to the cluster in the following ways:

- Adding 64-bit x86 compute machines to a cluster that uses 64-bit ARM control plane machines and already includes 64-bit ARM compute machines. In this case, 64-bit x86 is considered the secondary architecture.
- Adding 64-bit ARM compute machines to a cluster that uses 64-bit x86 control plane machines and already includes 64-bit x86 compute machines. In this case, 64-bit ARM is considered the secondary architecture.



NOTE

Before adding a secondary architecture node to your cluster, it is recommended to install the Multiarch Tuning Operator, and deploy a **ClusterPodPlacementConfig** custom resource. For more information, see "Managing workloads on multi-architecture clusters by using the Multiarch Tuning Operator".

16.1. INSTALLING A CLUSTER WITH MULTI-ARCHITECTURE SUPPORT

You can install a cluster with the support for configuring multi-architecture compute machines.

Prerequisites

- You installed the OpenShift CLI (**oc**).
- You have the OpenShift Container Platform installation program.
- You downloaded the pull secret for your cluster.

Procedure

1. Check that the **openshift-install** binary is using the **multi** payload by running the following command:

```
$ ./openshift-install version
```

Example output

```
./openshift-install 4.18.0
built from commit abc123etc
release image quay.io/openshift-release-dev/ocp-release@sha256:abc123wxyzetc
release architecture multi
default architecture amd64
```

The output must contain **release architecture multi** to indicate that the **openshift-install** binary is using the **multi** payload.

2. Update the **install-config.yaml** file to configure the architecture for the nodes.

Sample **install-config.yaml** file with multi-architecture configuration

```
apiVersion: v1
baseDomain: example.openshift.com
compute:
- architecture: amd64 1
  hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 3
controlPlane:
  architecture: arm64 2
  name: master
  platform: {}
  replicas: 3
# ...
```

- 1** Specify the architecture of the worker node. You can set this field to either **arm64** or **amd64**.
- 2** Specify the control plane node architecture. You can set this field to either **arm64** or **amd64**.

Next steps

- [Deploying the cluster](#)

Additional resources

- [Managing workloads on multi-architecture clusters by using the Multiarch Tuning Operator](#)