



OpenShift Container Platform 4.5

Installing on AWS

Installing OpenShift Container Platform AWS clusters

OpenShift Container Platform 4.5 Installing on AWS

Installing OpenShift Container Platform AWS clusters

Legal Notice

Copyright © 2021 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

This document provides instructions for installing and uninstalling OpenShift Container Platform clusters on Amazon Web Services.

Table of Contents

| | |
|---|----------|
| CHAPTER 1. INSTALLING ON AWS | 6 |
| 1.1. CONFIGURING AN AWS ACCOUNT | 6 |
| 1.1.1. Configuring Route 53 | 6 |
| 1.1.2. AWS account limits | 6 |
| 1.1.3. Required AWS permissions | 8 |
| 1.1.4. Creating an IAM user | 16 |
| 1.1.5. Supported AWS regions | 16 |
| 1.1.6. Next steps | 17 |
| 1.2. INSTALLING A CLUSTER QUICKLY ON AWS | 17 |
| 1.2.1. Prerequisites | 17 |
| 1.2.2. Internet and Telemetry access for OpenShift Container Platform | 18 |
| 1.2.3. Generating an SSH private key and adding it to the agent | 18 |
| 1.2.4. Obtaining the installation program | 20 |
| 1.2.5. Deploying the cluster | 20 |
| 1.2.6. Installing the CLI by downloading the binary | 22 |
| 1.2.6.1. Installing the CLI on Linux | 22 |
| 1.2.6.2. Installing the CLI on Windows | 23 |
| 1.2.6.3. Installing the CLI on macOS | 23 |
| 1.2.7. Logging in to the cluster | 24 |
| 1.2.8. Next steps | 24 |
| 1.3. INSTALLING A CLUSTER ON AWS WITH CUSTOMIZATIONS | 24 |
| 1.3.1. Prerequisites | 24 |
| 1.3.2. Internet and Telemetry access for OpenShift Container Platform | 25 |
| 1.3.3. Generating an SSH private key and adding it to the agent | 25 |
| 1.3.4. Obtaining the installation program | 27 |
| 1.3.5. Creating the installation configuration file | 27 |
| 1.3.5.1. Installation configuration parameters | 29 |
| 1.3.5.2. Sample customized install-config.yaml file for AWS | 36 |
| 1.3.6. Deploying the cluster | 38 |
| 1.3.7. Installing the CLI by downloading the binary | 39 |
| 1.3.7.1. Installing the CLI on Linux | 39 |
| 1.3.7.2. Installing the CLI on Windows | 40 |
| 1.3.7.3. Installing the CLI on macOS | 40 |
| 1.3.8. Logging in to the cluster | 41 |
| 1.3.9. Next steps | 41 |
| 1.4. INSTALLING A CLUSTER ON AWS WITH NETWORK CUSTOMIZATIONS | 41 |
| 1.4.1. Prerequisites | 42 |
| 1.4.2. Internet and Telemetry access for OpenShift Container Platform | 42 |
| 1.4.3. Generating an SSH private key and adding it to the agent | 43 |
| 1.4.4. Obtaining the installation program | 44 |
| 1.4.5. Creating the installation configuration file | 45 |
| 1.4.5.1. Installation configuration parameters | 46 |
| 1.4.5.2. Network configuration parameters | 53 |
| 1.4.5.3. Sample customized install-config.yaml file for AWS | 54 |
| 1.4.6. Modifying advanced network configuration parameters | 56 |
| 1.4.7. Cluster Network Operator configuration | 58 |
| 1.4.7.1. Configuration parameters for the OpenShift SDN default CNI network provider | 59 |
| 1.4.7.2. Configuration parameters for the OVN-Kubernetes default CNI network provider | 59 |
| 1.4.7.3. Cluster Network Operator example configuration | 60 |
| 1.4.8. Deploying the cluster | 60 |
| 1.4.9. Installing the CLI by downloading the binary | 61 |

| | |
|---|-----|
| 1.4.9.1. Installing the CLI on Linux | 62 |
| 1.4.9.2. Installing the CLI on Windows | 62 |
| 1.4.9.3. Installing the CLI on macOS | 63 |
| 1.4.10. Logging in to the cluster | 63 |
| 1.4.11. Next steps | 64 |
| 1.5. INSTALLING A CLUSTER ON AWS INTO AN EXISTING VPC | 64 |
| 1.5.1. Prerequisites | 64 |
| 1.5.2. About using a custom VPC | 64 |
| 1.5.2.1. Requirements for using your VPC | 65 |
| 1.5.2.2. VPC validation | 67 |
| 1.5.2.3. Division of permissions | 67 |
| 1.5.2.4. Isolation between clusters | 67 |
| 1.5.3. Internet and Telemetry access for OpenShift Container Platform | 68 |
| 1.5.4. Generating an SSH private key and adding it to the agent | 68 |
| 1.5.5. Obtaining the installation program | 69 |
| 1.5.6. Creating the installation configuration file | 70 |
| 1.5.6.1. Installation configuration parameters | 71 |
| 1.5.6.2. Sample customized install-config.yaml file for AWS | 79 |
| 1.5.6.3. Configuring the cluster-wide proxy during installation | 81 |
| 1.5.7. Deploying the cluster | 83 |
| 1.5.8. Installing the CLI by downloading the binary | 84 |
| 1.5.8.1. Installing the CLI on Linux | 84 |
| 1.5.8.2. Installing the CLI on Windows | 85 |
| 1.5.8.3. Installing the CLI on macOS | 85 |
| 1.5.9. Logging in to the cluster | 85 |
| 1.5.10. Next steps | 86 |
| 1.6. INSTALLING A PRIVATE CLUSTER ON AWS | 86 |
| 1.6.1. Prerequisites | 86 |
| 1.6.2. Private clusters | 87 |
| 1.6.2.1. Private clusters in AWS | 87 |
| 1.6.2.1.1. Limitations | 88 |
| 1.6.3. About using a custom VPC | 88 |
| 1.6.3.1. Requirements for using your VPC | 88 |
| 1.6.3.2. VPC validation | 90 |
| 1.6.3.3. Division of permissions | 91 |
| 1.6.3.4. Isolation between clusters | 91 |
| 1.6.4. Internet and Telemetry access for OpenShift Container Platform | 91 |
| 1.6.5. Generating an SSH private key and adding it to the agent | 92 |
| 1.6.6. Obtaining the installation program | 93 |
| 1.6.7. Manually creating the installation configuration file | 94 |
| 1.6.7.1. Installation configuration parameters | 94 |
| 1.6.7.2. Sample customized install-config.yaml file for AWS | 102 |
| 1.6.7.3. Configuring the cluster-wide proxy during installation | 104 |
| 1.6.8. Deploying the cluster | 106 |
| 1.6.9. Installing the CLI by downloading the binary | 107 |
| 1.6.9.1. Installing the CLI on Linux | 107 |
| 1.6.9.2. Installing the CLI on Windows | 107 |
| 1.6.9.3. Installing the CLI on macOS | 108 |
| 1.6.10. Logging in to the cluster | 108 |
| 1.6.11. Next steps | 109 |
| 1.7. INSTALLING A CLUSTER ON USER-PROVISIONED INFRASTRUCTURE IN AWS BY USING CLOUDFORMATION TEMPLATES | 109 |
| 1.7.1. Prerequisites | 109 |

| | |
|---|-----|
| 1.7.2. Internet and Telemetry access for OpenShift Container Platform | 110 |
| 1.7.3. Required AWS infrastructure components | 111 |
| 1.7.3.1. Cluster machines | 111 |
| 1.7.3.2. Certificate signing requests management | 112 |
| 1.7.3.3. Other infrastructure components | 112 |
| 1.7.3.4. Required AWS permissions | 118 |
| 1.7.4. Obtaining the installation program | 126 |
| 1.7.5. Generating an SSH private key and adding it to the agent | 126 |
| 1.7.6. Creating the installation files for AWS | 128 |
| 1.7.6.1. Creating the installation configuration file | 128 |
| 1.7.6.2. Configuring the cluster-wide proxy during installation | 129 |
| 1.7.6.3. Creating the Kubernetes manifest and Ignition config files | 130 |
| 1.7.7. Extracting the infrastructure name | 132 |
| 1.7.8. Creating a VPC in AWS | 133 |
| 1.7.8.1. CloudFormation template for the VPC | 135 |
| 1.7.9. Creating networking and load balancing components in AWS | 140 |
| 1.7.9.1. CloudFormation template for the network and load balancers | 143 |
| 1.7.10. Creating security group and roles in AWS | 151 |
| 1.7.10.1. CloudFormation template for security objects | 153 |
| 1.7.11. RHCOS AMIs for the AWS infrastructure | 162 |
| 1.7.12. Creating the bootstrap node in AWS | 163 |
| 1.7.12.1. CloudFormation template for the bootstrap machine | 167 |
| 1.7.13. Creating the control plane machines in AWS | 171 |
| 1.7.13.1. CloudFormation template for control plane machines | 176 |
| 1.7.14. Initializing the bootstrap node on AWS with user-provisioned infrastructure | 183 |
| 1.7.14.1. Creating the worker nodes in AWS | 184 |
| 1.7.14.1.1. CloudFormation template for worker machines | 187 |
| 1.7.15. Installing the CLI by downloading the binary | 190 |
| 1.7.15.1. Installing the CLI on Linux | 190 |
| 1.7.15.2. Installing the CLI on Windows | 190 |
| 1.7.15.3. Installing the CLI on macOS | 191 |
| 1.7.16. Logging in to the cluster | 191 |
| 1.7.17. Approving the certificate signing requests for your machines | 192 |
| 1.7.18. Initial Operator configuration | 194 |
| 1.7.18.1. Image registry storage configuration | 195 |
| 1.7.18.1.1. Configuring registry storage for AWS with user-provisioned infrastructure | 195 |
| 1.7.18.1.2. Configuring storage for the image registry in non-production clusters | 196 |
| 1.7.19. Deleting the bootstrap resources | 197 |
| 1.7.20. Creating the Ingress DNS Records | 197 |
| 1.7.21. Completing an AWS installation on user-provisioned infrastructure | 200 |
| 1.7.22. Next steps | 201 |
| 1.8. INSTALLING A CLUSTER ON AWS IN A RESTRICTED NETWORK WITH USER-PROVISIONED INFRASTRUCTURE | 201 |
| 1.8.1. Prerequisites | 201 |
| 1.8.2. About installations in restricted networks | 202 |
| 1.8.2.1. Additional limits | 202 |
| 1.8.3. Internet and Telemetry access for OpenShift Container Platform | 203 |
| 1.8.4. Required AWS infrastructure components | 203 |
| 1.8.4.1. Cluster machines | 203 |
| 1.8.4.2. Certificate signing requests management | 205 |
| 1.8.4.3. Other infrastructure components | 205 |
| 1.8.4.4. Required AWS permissions | 211 |
| 1.8.5. Generating an SSH private key and adding it to the agent | 218 |

| | |
|---|-----|
| 1.8.6. Creating the installation files for AWS | 219 |
| 1.8.6.1. Creating the installation configuration file | 219 |
| 1.8.6.2. Configuring the cluster-wide proxy during installation | 221 |
| 1.8.6.3. Creating the Kubernetes manifest and Ignition config files | 223 |
| 1.8.7. Extracting the infrastructure name | 225 |
| 1.8.8. Creating a VPC in AWS | 225 |
| 1.8.8.1. CloudFormation template for the VPC | 227 |
| 1.8.9. Creating networking and load balancing components in AWS | 233 |
| 1.8.9.1. CloudFormation template for the network and load balancers | 236 |
| 1.8.10. Creating security group and roles in AWS | 243 |
| 1.8.10.1. CloudFormation template for security objects | 246 |
| 1.8.11. RHCOS AMIs for the AWS infrastructure | 255 |
| 1.8.12. Creating the bootstrap node in AWS | 255 |
| 1.8.12.1. CloudFormation template for the bootstrap machine | 259 |
| 1.8.13. Creating the control plane machines in AWS | 263 |
| 1.8.13.1. CloudFormation template for control plane machines | 268 |
| 1.8.14. Initializing the bootstrap node on AWS with user-provisioned infrastructure | 275 |
| 1.8.14.1. Creating the worker nodes in AWS | 276 |
| 1.8.14.1.1. CloudFormation template for worker machines | 280 |
| 1.8.15. Logging in to the cluster | 282 |
| 1.8.16. Approving the certificate signing requests for your machines | 283 |
| 1.8.17. Initial Operator configuration | 285 |
| 1.8.17.1. Image registry storage configuration | 286 |
| 1.8.17.1.1. Configuring registry storage for AWS with user-provisioned infrastructure | 286 |
| 1.8.17.1.2. Configuring storage for the image registry in non-production clusters | 287 |
| 1.8.18. Deleting the bootstrap resources | 288 |
| 1.8.19. Creating the Ingress DNS Records | 288 |
| 1.8.20. Completing an AWS installation on user-provisioned infrastructure | 291 |
| 1.8.21. Next steps | 291 |
| 1.9. UNINSTALLING A CLUSTER ON AWS | 292 |
| 1.9.1. Removing a cluster that uses installer-provisioned infrastructure | 292 |

CHAPTER 1. INSTALLING ON AWS

1.1. CONFIGURING AN AWS ACCOUNT

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

1.1.1. Configuring Route 53

To install OpenShift Container Platform, the Amazon Web Services (AWS) account you use must have a dedicated public hosted zone in your Route 53 service. This zone must be authoritative for the domain. The Route 53 service provides cluster DNS resolution and name lookup for external connections to the cluster.

Procedure

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



NOTE


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

2. If you are using an existing domain and registrar, migrate its DNS to AWS. See [Making Amazon Route 53 the DNS Service for an Existing Domain](#) in the AWS documentation.
3. Create a public hosted zone for your domain or subdomain. See [Creating a Public Hosted Zone](#) in the AWS documentation.
Use an appropriate root domain, such as **openshiftcorp.com**, or subdomain, such as **clusters.openshiftcorp.com**.
4. Extract the new authoritative name servers from the hosted zone records. See [Getting the Name Servers for a Public Hosted Zone](#) in the AWS documentation.
5. Update the registrar records for the AWS Route 53 name servers that your domain uses. For example, if you registered your domain to a Route 53 service in a different accounts, see the following topic in the AWS documentation: [Adding or Changing Name Servers or Glue Records](#).
6. If you are using a subdomain, add its delegation records to the parent domain. This gives Amazon Route 53 responsibility for the subdomain. Follow the delegation procedure outlined by the DNS provider of the parent domain. See [Creating a subdomain that uses Amazon Route 53 as the DNS service without migrating the parent domain](#) in the AWS documentation for an example high level procedure.

1.1.2. AWS account limits

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

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

| Component | Number of clusters available by default | Default AWS limit | Description |
|--------------------|---|--------------------|---|
| Instance Limits | Varies | Varies | <p>By default, each cluster creates the following instances:</p> <ul style="list-style-type: none"> • One bootstrap machine, which is removed after installation • Three master nodes • Three worker nodes <p>These instance type counts are within a new account's default limit. To deploy more worker nodes, enable autoscaling, deploy large workloads, or use a different instance type, review your account limits to ensure that your cluster can deploy the machines that you need.</p> <p>In most regions, the bootstrap and worker machines use an m4.large machines and the master machines use m4.xlarge instances. In some regions, including all regions that do not support these instance types, m5.large and m5.xlarge instances are used instead.</p> |
| Elastic IPs (EIPs) | 0 to 1 | 5 EIPs per account | <p>To provision the cluster in a highly available configuration, the installation program creates a public and private subnet for each availability zone within a region. Each private subnet requires a NAT Gateway, and each NAT gateway requires a separate elastic IP. Review the AWS region map to determine how many availability zones are in each region. To take advantage of the default high availability, install the cluster in a region with at least three availability zones. To install a cluster in a region with more than five availability zones, you must increase the EIP limit.</p> <div>  <p>IMPORTANT</p> <p>To use the us-east-1 region, you must increase the EIP limit for your account.</p> </div> |

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

1.1.3. Required AWS permissions

When you attach the **AdministratorAccess** policy to the IAM user that you create in Amazon Web Services (AWS), you grant that user all of the required permissions. To deploy all components of an OpenShift Container Platform cluster, the IAM user requires the following permissions:

Example 1.1. Required EC2 permissions for installation

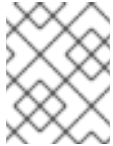
- **tag:TagResources**
- **tag:UntagResources**
- **ec2:AllocateAddress**
- **ec2:AssociateAddress**
- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2>CreateNetworkInterface**
- **ec2:AttachNetworkInterface**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2>DeleteSecurityGroup**
- **ec2>DeleteSnapshot**
- **ec2>DeleteTags**
- **ec2:DeregisterImage**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInternetGateways**
- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**
- **ec2:DescribeNetworkAcls**
- **ec2:DescribeNetworkInterfaces**

- **ec2:DescribePrefixLists**
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcAttribute**
- **ec2:DescribeVpcClassicLink**
- **ec2:DescribeVpcClassicLinkDnsSupport**
- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**
- **ec2:GetEbsDefaultKmsKeyId**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyNetworkInterfaceAttribute**
- **ec2:ReleaseAddress**
- **ec2:RevokeSecurityGroupEgress**
- **ec2:RevokeSecurityGroupIngress**
- **ec2:RunInstances**
- **ec2:TerminateInstances**

Example 1.2. Required permissions for creating network resources during installation

- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**
- **ec2:CreateDhcpOptions**
- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**

- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**



NOTE

If you use an existing VPC, your account does not require these permissions for creating network resources.

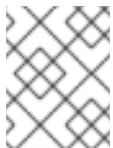
Example 1.3. Required Elastic Load Balancing permissions for installation

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

- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:RegisterTargets**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**

Example 1.4. Required IAM permissions for installation

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



NOTE

If you have not created an elastic load balancer (ELB) in your AWS account, the IAM user also requires the **iam:CreateServiceLinkedRole** permission.

Example 1.5. Required Route 53 permissions for installation

- **route53:ChangeResourceRecordSets**

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

Example 1.6. Required S3 permissions for installation

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

- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

Example 1.7. S3 permissions that cluster Operators require

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

Example 1.8. Required permissions to delete base cluster resources

- **autoscaling:DescribeAutoScalingGroups**
- **ec2:DeleteNetworkInterface**
- **ec2:DeleteVolume**
- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**
- **s3:DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

Example 1.9. Required permissions to delete network resources

- **ec2:DeleteDhcpOptions**

- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteRoute**
- **ec2:DeleteRouteTable**
- **ec2:DeleteSubnet**
- **ec2:DeleteVpc**
- **ec2:DeleteVpcEndpoints**
- **ec2:DetachInternetGateway**
- **ec2:DisassociateRouteTable**
- **ec2:ReplaceRouteTableAssociation**



NOTE

If you use an existing VPC, your account does not require these permissions to delete network resources.

Example 1.10. Additional IAM and S3 permissions that are required to create manifests

- **iam:CreateAccessKey**
- **iam:CreateUser**
- **iam>DeleteAccessKey**
- **iam>DeleteUser**
- **iam>DeleteUserPolicy**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **s3:PutBucketPublicAccessBlock**
- **s3:GetBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**
- **s3:HeadBucket**

- **s3:ListBucketMultipartUploads**
- **s3:AbortMultipartUpload**

1.1.4. Creating an IAM user

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

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

Procedure

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



NOTE

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

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



IMPORTANT

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

1.1.5. Supported AWS regions

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

- **ap-northeast-1** (Tokyo)
- **ap-northeast-2** (Seoul)

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

1.1.6. Next steps

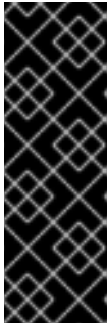
- Install an OpenShift Container Platform cluster:
 - [Quickly install a cluster](#) with default options on installer-provisioned infrastructure
 - [Install a cluster with cloud customizations on installer-provisioned infrastructure](#)
 - [Install a cluster with network customizations on installer-provisioned infrastructure](#)
 - [Installing a cluster on user-provisioned infrastructure in AWS by using CloudFormation templates](#)

1.2. INSTALLING A CLUSTER QUICKLY ON AWS

In OpenShift Container Platform version 4.5, you can install a cluster on Amazon Web Services (AWS) that uses the default configuration options.

1.2.1. Prerequisites

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



IMPORTANT

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

- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

1.2.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

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

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page 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 content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.

**NOTE**

In a production environment, you require disaster recovery and debugging.

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

**NOTE**

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

Procedure

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

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

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

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

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

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

Example output

```
Agent pid 31874
```

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

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

Example output

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

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

Next steps

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

1.2.4. Obtaining the installation program

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

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

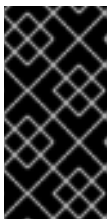
Procedure

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



IMPORTANT

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



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

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

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

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

1.2.5. 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

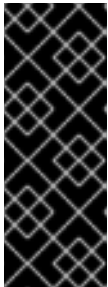
- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Run the installation program:

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

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



IMPORTANT

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

Provide values at the prompts:

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



NOTE

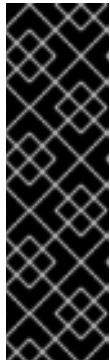
For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- b. Select **aws** as the platform to target.
- c. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.
- d. Select the AWS region to deploy the cluster to.
- e. Select the base domain for the Route 53 service that you configured for your cluster.
- f. Enter a descriptive name for your cluster.
- g. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.

**NOTE**

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

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

**IMPORTANT**

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, 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.

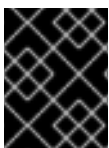
**IMPORTANT**

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

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

1.2.6. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order 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 4.5. Download and install the new version of **oc**.

1.2.6.1. Installing the CLI on Linux

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

Procedure

- Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
- Select your infrastructure provider, and, if applicable, your installation type.
- In the **Command-line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
- Unpack the archive:

```
$ tar xvfz <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

1.2.6.2. Installing the CLI on Windows

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

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command-line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
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
```

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

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

1.2.6.3. Installing the CLI on macOS

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

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command-line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
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
```

■

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

```
$ oc <command>
```

1.2.7. Logging in to the cluster

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

Prerequisites

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

Procedure

1. Export the **kubeadmin** credentials:

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

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

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

```
$ oc whoami
```

Example output

```
system:admin
```

1.2.8. Next steps

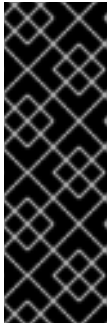
- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#) .
- If necessary, you can [remove cloud provider credentials](#).

1.3. INSTALLING A CLUSTER ON AWS WITH CUSTOMIZATIONS

In OpenShift Container Platform version 4.5, you can install a customized cluster on infrastructure that the installation program provisions on Amazon Web Services (AWS). To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

1.3.1. Prerequisites

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



IMPORTANT

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

- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

1.3.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

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

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page 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 content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.

**NOTE**

In a production environment, you require disaster recovery and debugging.

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

**NOTE**

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

Procedure

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

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

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

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

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

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

Example output

```
Agent pid 31874
```

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

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

Example output

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

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

Next steps

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

1.3.4. Obtaining the installation program

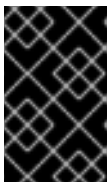
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

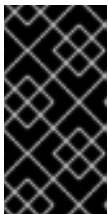
Procedure

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



IMPORTANT

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



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

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

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

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

1.3.5. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Amazon Web Services (AWS).

Prerequisites

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

Procedure

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

- a. Run the following command:

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

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



IMPORTANT

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

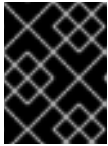
- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.



NOTE

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

- ii. Select **AWS** as the platform to target.
 - iii. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.
 - iv. Select the AWS region to deploy the cluster to.
 - v. Select the base domain for the Route 53 service that you configured for your cluster.
 - vi. Enter a descriptive name for your cluster.
 - vii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

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

1.3.5.1. Installation configuration parameters

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



NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.



IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.


Table 1.1. Required parameters

| Parameter | Description | Values |
|-------------------|--|--|
| apiVersion | The API version for the install-config.yaml content. The current version is v1 . The installer 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, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform. | Object |
| pullSecret | Get this pull secret from https://cloud.redhat.com/openshift/install/pull-secret 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> |


Table 1.2. 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 |
| compute | The configuration for the machines that comprise the compute nodes. | Array of machine-pool objects. For details, see the following "Machine-pool" table. |

| Parameter | Description | Values |
|----------------------------------|--|---|
| compute.architecture | Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default). | String |
| compute.hyperthreading | <p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> | Enabled or Disabled |
| compute.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, openstack, ovirt, 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 . |
| controlPlane | The configuration for the machines that comprise the control plane. | Array of MachinePool objects. For details, see the following "Machine-pool" table. |
| controlPlane.architecture | Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default). | String |

| Parameter | Description | Values |
|------------------------------------|---|--|
| controlPlane.hyperthreading | <p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> | Enabled or Disabled |
| controlPlane.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, openstack, ovirt, vsphere , or {} |
| controlPlane.replicas | The number of control plane machines to provision. | The only supported value is 3 , which is the default value. |

| Parameter | Description | Values |
|---------------------------------------|--|---|
| fips | <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> <div>  <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p> </div> | false or true |
| 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 |
| networking | The configuration for the pod network provider in the cluster. | Object |
| networking.clusterNetwork | The IP address pools for pods. The default is 10.128.0.0/14 with a host prefix of /23 . | Array of objects |
| networking.clusterNetwork.cidr | Required if you use networking.clusterNetwork . The IP block address pool. | IP network. IP networks are represented as strings using Classless Inter-Domain Routing (CIDR) notation with a traditional IP address or network number, followed by the forward slash (/) character, followed by a decimal value between 0 and 32 that describes the number of significant bits. For example, 10.0.0.0/16 represents IP addresses 10.0.0.0 through 10.0.255.255 . |

| Parameter | Description | Values |
|---|---|---|
| networking.clusterNetwork.hostPrefix | Required if you use networking.clusterNetwork . The prefix size to allocate to each node from the CIDR. For example, 24 would allocate $2^8=256$ addresses to each node. | Integer |
| networking.machineNetwork | The IP address pools for machines. | Array of objects |
| networking.machineNetwork.cidr | Required if you use networking.machineNetwork . An IP address block. The default is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default is 192.168.126.0/24 . | IP network in Classless Inter-Domain Routing (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.networkType | The type of network to install. The default is OpenShiftSDN . | String |
| networking.serviceNetwork | The IP address pools for services. The default is 172.30.0.0/16. | Array of IP networks. IP networks are represented as strings using Classless Inter-Domain Routing (CIDR) notation with a traditional IP address or network number, followed by the forward slash (/) character, followed by a decimal value between 0 and 32 that describes the number of significant bits. For example, 10.0.0.0/16 represents IP addresses 10.0.0.0 through 10.0.255.255 . |
| 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 . |


| Parameter | Description | Values |
|---------------|---|---|
| sshKey | <p>The SSH key to authenticate access to your cluster machines.</p> <div>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p> </div> | For example, sshKey: ssh-ed25519 AAAA... |

Table 1.3. Optional AWS parameters

| Parameter | Description | Values |
|---|--|--|
| compute.platform.aws.rootVolume.iops | The Input/Output Operations Per Second (IOPS) that is reserved for the root volume. | Integer, for example 4000 . |
| compute.platform.aws.rootVolume.size | The size in GiB of the root volume. | Integer, for example 500 . |
| compute.platform.aws.rootVolume.type | The instance type of the root volume. | Valid AWS EBS instance type , such as io1 . |
| compute.platform.aws.type | The EC2 instance type for the compute machines. | Valid AWS instance type , such as c5.9xlarge . |
| compute.platform.aws.zones | The availability zones where the installation program creates machines for the compute machine pool. If you provide your own VPC, you must provide a subnet in that availability zone. | A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence . |
| compute.aws.region | The AWS region that the installation program creates compute resources in. | Any valid AWS region , such as us-east-1 . |
| controlPlane.platform.aws.type | The EC2 instance type for the control plane machines. | Valid AWS instance type , such as c5.9xlarge . |

| Parameter | Description | Values |
|--|---|---|
| controlPlane.platform.aws.zones | The availability zones where the installation program creates machines for the control plane machine pool. | A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence . |
| controlPlane.aws.region | The AWS region that the installation program creates control plane resources in. | Valid AWS region , such as us-east-1 . |
| platform.aws.userTags | A map of keys and values that the installation program adds as tags to all resources that it creates. | Any valid YAML map, such as key value pairs in the <key>: <value> format. For more information about AWS tags, see Tagging Your Amazon EC2 Resources in the AWS documentation. |
| platform.aws.subnets | If you provide the VPC instead of allowing the installation program to create the VPC for you, specify the subnet for the cluster to use. The subnet must be part of the same machineNetwork[].cidr ranges that you specify. For a standard cluster, specify a public and a private subnet for each availability zone. For a private cluster, specify a private subnet for each availability zone. | Valid subnet IDs. |

1.3.5.2. Sample customized `install-config.yaml` file for AWS

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



IMPORTANT

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

```

apiVersion: v1
baseDomain: example.com ❶
controlPlane: ❷
  hyperthreading: Enabled ❸ ❹
  name: master
  platform:
    aws:
      zones:
        - us-west-2a

```



```

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

```

1 9 10 13 Required. The installation program prompts you for this value.

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

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

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

can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

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

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

11 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.

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

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.

1.3.6. 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

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

Procedure

1. Run the installation program:

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

- 1** For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.

- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

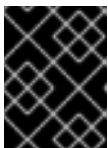
If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

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



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, 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.



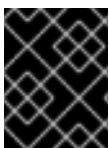
IMPORTANT

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

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

1.3.7. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order 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 4.5. Download and install the new version of **oc**.

1.3.7.1. Installing the CLI on Linux

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

Procedure

- Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
- Select your infrastructure provider, and, if applicable, your installation type.
- In the **Command-line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.

4. Unpack the archive:

```
$ tar xvfz <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

1.3.7.2. Installing the CLI on Windows

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

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command-line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
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
```

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

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

1.3.7.3. Installing the CLI on macOS

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

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command-line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
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
```

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

```
$ oc <command>
```

1.3.8. Logging in to the cluster

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

Prerequisites

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

Procedure

1. Export the **kubeadmin** credentials:

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

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

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

```
$ oc whoami
```

Example output

```
system:admin
```

1.3.9. Next steps

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

1.4. INSTALLING A CLUSTER ON AWS WITH NETWORK CUSTOMIZATIONS

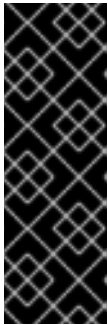
In OpenShift Container Platform version 4.5, you can install a cluster on Amazon Web Services (AWS) with customized network configuration options. By customizing your network configuration, your cluster

can coexist with existing IP address allocations in your environment and integrate with existing MTU and VXLAN configurations.

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

1.4.1. Prerequisites

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



IMPORTANT

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

- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

1.4.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

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

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page 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 content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

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



NOTE

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

Procedure

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

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

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

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

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

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

Example output

```
Agent pid 31874
```

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

-

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

Example output

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

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

Next steps

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

1.4.4. Obtaining the installation program

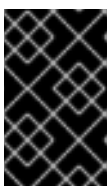
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

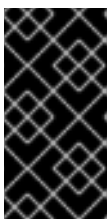
Procedure

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



IMPORTANT

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



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

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

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

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services

that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

1.4.5. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Amazon Web Services (AWS).

Prerequisites

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

Procedure

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

- a. Run the following command:

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

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



IMPORTANT

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

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



NOTE

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

- ii. Select **AWS** as the platform to target.
- iii. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.
- iv. Select the AWS region to deploy the cluster to.
- v. Select the base domain for the Route 53 service that you configured for your cluster.

- vi. Enter a descriptive name for your cluster.
 - vii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

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

1.4.5.1. Installation configuration parameters

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



NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.



IMPORTANT


The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.


Table 1.4. Required parameters


| Parameter | Description | Values |
|-------------------|---|--------|
| apiVersion | The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions. | String |


| Parameter | Description | Values |
|----------------------|--|---|
| 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 |
| 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 , openstack , ovirt , vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform. | Object |
| pullSecret | Get this pull secret from https://cloud.redhat.com/openshift/install/pull-secret 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> |

Table 1.5. 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 |
| compute | The configuration for the machines that comprise the compute nodes. | Array of machine-pool objects. For details, see the following "Machine-pool" table. |
| compute.architecture | Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default). | String |
| compute.hyperthreading | <p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> | Enabled or Disabled |
| compute.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 , openstack , ovirt , 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 . |

| Parameter | Description | Values |
|------------------------------------|--|---|
| controlPlane | The configuration for the machines that comprise the control plane. | Array of MachinePool objects. For details, see the following "Machine-pool" table. |
| controlPlane.architecture | Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default). | String |
| controlPlane.hyperthreading | <p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> | Enabled or Disabled |
| controlPlane.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, openstack, ovirt, vsphere , or {} |
| controlPlane.replicas | The number of control plane machines to provision. | The only supported value is 3 , which is the default value. |

| Parameter | Description | Values |
|---------------------------------------|--|---|
| fips | <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> <div>  <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p> </div> | false or true |
| 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 |
| networking | The configuration for the pod network provider in the cluster. | Object |
| networking.clusterNetwork | The IP address pools for pods. The default is 10.128.0.0/14 with a host prefix of /23 . | Array of objects |
| networking.clusterNetwork.cidr | Required if you use networking.clusterNetwork . The IP block address pool. | IP network. IP networks are represented as strings using Classless Inter-Domain Routing (CIDR) notation with a traditional IP address or network number, followed by the forward slash (/) character, followed by a decimal value between 0 and 32 that describes the number of significant bits. For example, 10.0.0.0/16 represents IP addresses 10.0.0.0 through 10.0.255.255 . |

| Parameter | Description | Values |
|---|---|---|
| networking.clusterNetwork.hostPrefix | Required if you use networking.clusterNetwork . The prefix size to allocate to each node from the CIDR. For example, 24 would allocate $2^8=256$ addresses to each node. | Integer |
| networking.machineNetwork | The IP address pools for machines. | Array of objects |
| networking.machineNetwork.cidr | Required if you use networking.machineNetwork . An IP address block. The default is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default is 192.168.126.0/24 . | <p>IP network in Classless Inter-Domain Routing (CIDR) notation. For example, 10.0.0.0/16.</p> <div>  <div> <p>NOTE</p> <p>Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.</p> </div> </div> |
| networking.networkType | The type of network to install. The default is OpenShiftSDN . | String |
| networking.serviceNetwork | The IP address pools for services. The default is 172.30.0.0/16. | Array of IP networks. IP networks are represented as strings using Classless Inter-Domain Routing (CIDR) notation with a traditional IP address or network number, followed by the forward slash (/) character, followed by a decimal value between 0 and 32 that describes the number of significant bits. For example, 10.0.0.0/16 represents IP addresses 10.0.0.0 through 10.0.255.255 . |
| 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 . |


| Parameter | Description | Values |
|---------------|---|---|
| sshKey | <p>The SSH key to authenticate access to your cluster machines.</p> <div>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p> </div> | For example, sshKey: ssh-ed25519 AAAA... |

Table 1.6. Optional AWS parameters

| Parameter | Description | Values |
|---|--|--|
| compute.platform.aws.rootVolume.iops | The Input/Output Operations Per Second (IOPS) that is reserved for the root volume. | Integer, for example 4000 . |
| compute.platform.aws.rootVolume.size | The size in GiB of the root volume. | Integer, for example 500 . |
| compute.platform.aws.rootVolume.type | The instance type of the root volume. | Valid AWS EBS instance type , such as io1 . |
| compute.platform.aws.type | The EC2 instance type for the compute machines. | Valid AWS instance type , such as c5.9xlarge . |
| compute.platform.aws.zones | The availability zones where the installation program creates machines for the compute machine pool. If you provide your own VPC, you must provide a subnet in that availability zone. | A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence . |
| compute.aws.region | The AWS region that the installation program creates compute resources in. | Any valid AWS region , such as us-east-1 . |
| controlPlane.platform.aws.type | The EC2 instance type for the control plane machines. | Valid AWS instance type , such as c5.9xlarge . |

| Parameter | Description | Values |
|--|---|---|
| controlPlane.platform.aws.zones | The availability zones where the installation program creates machines for the control plane machine pool. | A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence . |
| controlPlane.aws.region | The AWS region that the installation program creates control plane resources in. | Valid AWS region , such as us-east-1 . |
| platform.aws.userTags | A map of keys and values that the installation program adds as tags to all resources that it creates. | Any valid YAML map, such as key value pairs in the <key>: <value> format. For more information about AWS tags, see Tagging Your Amazon EC2 Resources in the AWS documentation. |
| platform.aws.subnets | If you provide the VPC instead of allowing the installation program to create the VPC for you, specify the subnet for the cluster to use. The subnet must be part of the same machineNetwork[].cidr ranges that you specify. For a standard cluster, specify a public and a private subnet for each availability zone. For a private cluster, specify a private subnet for each availability zone. | Valid subnet IDs. |



IMPORTANT

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

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

1.4.5.2. Network configuration parameters

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



NOTE

You cannot modify these parameters in the **install-config.yaml** file after installation.

Table 1.7. Required network parameters

| Parameter | Description | Value |
|---|---|---|
| networking.net workType | The default Container Network Interface (CNI) network provider plug-in to deploy. The OpenShiftSDN plug-in is the only plug-in supported in OpenShift Container Platform 4.5. The OVNKubernetes plug-in is available as a Technology Preview in OpenShift Container Platform 4.5. | Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN . |
| networking.clus terNetwork[].cid r | A block of IP addresses from which pod IP addresses are allocated. The OpenShiftSDN network plug-in supports multiple cluster networks. The address blocks for multiple cluster networks must not overlap. Select address pools large enough to fit your anticipated workload. | An IP address allocation in CIDR format. The default value is 10.128.0.0/14 . |
| networking.clus terNetwork[].ho stPrefix | The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 , then each node is assigned a /23 subnet out of the given cidr , allowing for $510 (2^{(32 - 23)} - 2)$ pod IP addresses. | A subnet prefix. The default value is 23 . |
| networking.serv iceNetwork[] | A block of IP addresses for services. OpenShiftSDN allows only one serviceNetwork block. The address block must not overlap with any other network block. | An IP address allocation in CIDR format. The default value is 172.30.0.0/16 . |
| networking.mac hineNetwork[].ci dr | A block of IP addresses assigned to nodes created by the OpenShift Container Platform installation program while installing the cluster. The address block must not overlap with any other network block. Multiple CIDR ranges may be specified. | An IP address allocation in CIDR format. The default value is 10.0.0.0/16 . |

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

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



IMPORTANT

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

```

apiVersion: v1
baseDomain: example.com 1
controlPlane: 2
  hyperthreading: Enabled 3 4
  name: master
  platform:

```

```

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

```

1 9 11 14 Required. The installation program prompts you for this value.

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

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

- 4 5** Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You

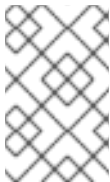


IMPORTANT

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

- 8** To configure faster storage for etcd, especially for larger clusters, set the storage type as **io1** and set **iops** to **2000**.
- 12** 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.
- 13** You can optionally provide the **sshKey** value that you use to access the machines in your cluster.

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.

1.4.6. Modifying advanced network configuration parameters

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



IMPORTANT

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

Prerequisites

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

Procedure

1. Use the following command to create manifests:

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

- 1 For **<installation_directory>**, specify the name of the directory that contains the **install-config.yaml** file for your cluster.
2. Create a file that is named **cluster-network-03-config.yml** in the **<installation_directory>/manifests/** directory:

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

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

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

```
$ ls <installation_directory>/manifests/cluster-network-*
```

Example output

```
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-network-03-config.yml
```

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

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec: 1
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  serviceNetwork:
    - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
```

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

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

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

1.4.7. 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 CR object that is named **cluster**. The CR specifies the parameters for the **Network** API in the **operator.openshift.io** API group.

You can specify the cluster network configuration for your OpenShift Container Platform cluster by setting the parameter values for the **defaultNetwork** parameter in the CNO CR. The following CR displays the default configuration for the CNO and explains both the parameters you can configure and the valid parameter values:

Cluster Network Operator CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork: ❶
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  serviceNetwork: ❷
    - 172.30.0.0/16
  defaultNetwork: ❸
  ...
  kubeProxyConfig: ❹
    iptablesSyncPeriod: 30s ❺
    proxyArguments:
      iptables-min-sync-period: ❻
      - 0s
```

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

❸ Configures the default Container Network Interface (CNI) network provider for the cluster network.

❹ The parameters for this object specify the **kube-proxy** configuration. If you do not specify the parameter values, the Cluster Network Operator applies the displayed default parameter values. If you are using the OVN-Kubernetes default CNI network provider, the kube-proxy configuration has no effect.

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



NOTE

Because of performance improvements introduced in OpenShift Container Platform 4.3 and greater, adjusting the **iptablesSyncPeriod** parameter is no longer necessary.

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

1.4.7.1. Configuration parameters for the OpenShift SDN default CNI network provider

The following YAML object describes the configuration parameters for the OpenShift SDN default Container Network Interface (CNI) network provider.

```
defaultNetwork:
  type: OpenShiftSDN ❶
  openshiftSDNConfig: ❷
    mode: NetworkPolicy ❸
    mtu: 1450 ❹
    vxlanPort: 4789 ❺
```

- ❶ Specified in the **install-config.yaml** file.
- ❷ Specify only if you want to override part of the OpenShift SDN configuration.
- ❸ Configures the network isolation mode for OpenShift SDN. The allowed values are **Multitenant**, **Subnet**, or **NetworkPolicy**. The default value is **NetworkPolicy**.
- ❹ The maximum transmission unit (MTU) for the VXLAN 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.

If the auto-detected value is not what you expected 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.

If your cluster requires different MTU values for different nodes, you must set this value to **50** 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 **1450**.

- ❺ The port to use for all VXLAN packets. The default value is **4789**. If you are running in a virtualized environment with existing nodes that are part of another VXLAN network, then you might be required to change this. For example, when running an OpenShift SDN overlay on top of VMware NSX-T, you must select an alternate port for VXLAN, since both SDNs use the same default VXLAN port number.

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

1.4.7.2. Configuration parameters for the OVN-Kubernetes default CNI network provider

The following YAML object describes the configuration parameters for the OVN-Kubernetes default CNI network provider.

```
defaultNetwork:
  type: OVNKubernetes ❶
  ovnKubernetesConfig: ❷
    mtu: 1400 ❸
    genevePort: 6081 ❹
```

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

- 2 Specify only if you want to override part of the OVN-Kubernetes configuration.
- 3 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.

If the auto-detected value is not what you expected 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.

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**.

- 4 The UDP port for the Geneve overlay network.

1.4.7.3. Cluster Network Operator example configuration

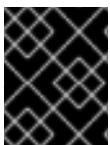
A complete CR object for the CNO is displayed in the following example:

Cluster Network Operator example CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  serviceNetwork:
    - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
  kubeProxyConfig:
    iptablesSyncPeriod: 30s
    proxyArguments:
      iptables-min-sync-period:
        - 0s
```

1.4.8. 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

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

Procedure

1. Run the installation program:

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

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



NOTE

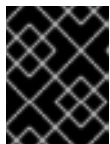
If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

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



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, 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-bootstrap** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.



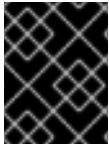
IMPORTANT

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

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

1.4.9. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order 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 4.5. Download and install the new version of **oc**.

1.4.9.1. Installing the CLI on Linux

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

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command-line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

1.4.9.2. Installing the CLI on Windows

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

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command-line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
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
```

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

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

1.4.9.3. Installing the CLI on macOS

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

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command-line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
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
```

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

```
$ oc <command>
```

1.4.10. Logging in to the cluster

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

Prerequisites

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

Procedure

1. Export the **kubeadmin** credentials:

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

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

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

```
$ oc whoami
```

Example output

```
system:admin
```

1.4.11. Next steps

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

1.5. INSTALLING A CLUSTER ON AWS INTO AN EXISTING VPC

In OpenShift Container Platform version 4.5, you can install a cluster into an existing Amazon Virtual Private Cloud (VPC) on Amazon Web Services (AWS). 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.

1.5.1. Prerequisites

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



IMPORTANT

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

- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

1.5.2. About using a custom VPC

In OpenShift Container Platform 4.5, you can deploy a cluster into existing subnets in an existing Amazon Virtual Private Cloud (VPC) in Amazon Web Services (AWS). By deploying OpenShift Container Platform into an existing AWS 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.

Because the installation program cannot know what other components are also in your existing subnets, it cannot choose subnet CIDRs and so forth on your behalf. You must configure networking for the subnets that you install your cluster to yourself.

1.5.2.1. Requirements for using your VPC

The installation program no longer creates the following components:

- Internet gateways
- NAT gateways
- Subnets
- Route tables
- VPCs
- VPC DHCP options
- VPC endpoints

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 must meet the following characteristics:

- The VPC's CIDR block must contain the **Networking.MachineCIDR** range, which is the IP address pool for cluster machines.
- The VPC must not use the **kubernetes.io/cluster/.*: owned** tag.
- You must enable the **enableDnsSupport** and **enableDnsHostnames** attributes in your VPC so that the cluster can use the Route 53 zones that are attached to the VPC to resolve cluster's internal DNS records. See [DNS Support in Your VPC](#) in the AWS documentation.

If you use a cluster with public access, you must create a public and a private subnet for each availability zone that your cluster uses.

The installation program modifies your subnets to add the **kubernetes.io/cluster/.*: shared** tag, so your subnets must have at least one free tag slot available for it. Review the current [Tag Restrictions](#) in the AWS documentation to ensure that the installation program can add a tag to each subnet that you specify.

If you are working in a disconnected environment, you are unable to reach the public IP addresses for EC2 and ELB endpoints. To resolve this, you must create a VPC endpoint and attach it to the subnet that the clusters are using. The endpoints should be named as follows:

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

Required VPC components

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

| Component | AWS type | Description | |
|------------------------|---|--|----------------------------|
| VPC | <ul style="list-style-type: none">• AWS::EC2::VPC• AWS::EC2::VPCEndpoint | You must provide a public VPC for the cluster to use. The VPC uses an endpoint that references the route tables for each subnet to improve communication with the registry that is hosted in S3. | |
| Public subnets | <ul style="list-style-type: none">• AWS::EC2::Subnet• AWS::EC2::SubnetNetworkAclAssociation | Your VPC must have public subnets for between 1 and 3 availability zones and associate them with appropriate Ingress rules. | |
| Internet gateway | <ul style="list-style-type: none">• AWS::EC2::InternetGateway• AWS::EC2::VPCGatewayAttachment• AWS::EC2::RouteTable• AWS::EC2::Route• AWS::EC2::SubnetRouteTableAssociation• AWS::EC2::NatGateway• AWS::EC2::EIP | You must have a public Internet gateway, with public routes, attached to the VPC. In the provided templates, each public subnet has a NAT gateway with an EIP address. These NAT gateways allow cluster resources, like private subnet instances, to reach the Internet and are not required for some restricted network or proxy scenarios. | |
| Network access control | <ul style="list-style-type: none">• AWS::EC2::NetworkAcl• AWS::EC2::NetworkAclEntry | You must allow the VPC to access the following ports: | |
| | | Port | Reason |
| | | 80 | Inbound HTTP traffic |
| | | 443 | Inbound HTTPS traffic |
| | | 22 | Inbound SSH traffic |
| | | 1024 - 65535 | Inbound ephemeral traffic |
| | | 0 - 65535 | Outbound ephemeral traffic |

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

1.5.2.2. 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 private subnets.
- The subnet CIDRs belong to the machine CIDR that you specified.
- You provide subnets for each availability zone. Each availability zone contains no more than one public and one private subnet. If you use a private cluster, provide only a private subnet for each availability zone. Otherwise, provide exactly one public and private subnet for each availability zone.
- You provide a public subnet for each private subnet availability zone. Machines are not provisioned in availability zones that you do not provide private subnets for.

If you destroy a cluster that uses an existing VPC, the VPC is not deleted. When you remove the OpenShift Container Platform cluster from a VPC, the **kubernetes.io/cluster/.*: shared** tag is removed from the subnets that it used.

1.5.2.3. 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 resource 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 AWS 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 ELBs, security groups, S3 buckets, and nodes.

1.5.2.4. 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 from 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.

1.5.3. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

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

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page 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.

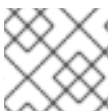


IMPORTANT

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 content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

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



NOTE

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

Procedure

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

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

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

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

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

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

Example output

```
Agent pid 31874
```

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

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

Example output

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

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

Next steps

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

1.5.5. Obtaining the installation program

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

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.

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

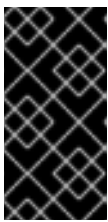
Procedure

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



IMPORTANT

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



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

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

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

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

1.5.6. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Amazon Web Services (AWS).

Prerequisites

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

Procedure

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

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

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



IMPORTANT

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

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



NOTE

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

- ii. Select **AWS** as the platform to target.
 - iii. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.
 - iv. Select the AWS region to deploy the cluster to.
 - v. Select the base domain for the Route 53 service that you configured for your cluster.
 - vi. Enter a descriptive name for your cluster.
 - vii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

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

1.5.6.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for

the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.



IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.


Table 1.8. Required parameters


| Parameter | Description | Values |
|----------------------|--|---|
| apiVersion | The API version for the install-config.yaml content. The current version is v1 . The installer 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 |
| 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 . |


| Parameter | Description | Values |
|-------------------|---|---|
| platform | The configuration for the specific platform upon which to perform the installation: aws , baremetal , azure , openstack , ovirt , vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform. | Object |
| pullSecret | Get this pull secret from https://cloud.redhat.com/openshift/install/pull-secret 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> |


Table 1.9. 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 |
| compute | The configuration for the machines that comprise the compute nodes. | Array of machine-pool objects. For details, see the following "Machine-pool" table. |
| compute.architecture | Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default). | String |

| Parameter | Description | Values |
|----------------------------------|---|---|
| compute.hyperthreading | <p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> | Enabled or Disabled |
| compute.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, openstack, ovirt, 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 . |
| controlPlane | The configuration for the machines that comprise the control plane. | Array of MachinePool objects. For details, see the following "Machine-pool" table. |
| controlPlane.architecture | Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default). | String |

| Parameter | Description | Values |
|------------------------------------|---|--|
| controlPlane.hyperthreading | <p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> | Enabled or Disabled |
| controlPlane.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, openstack, ovirt, vsphere , or {} |
| controlPlane.replicas | The number of control plane machines to provision. | The only supported value is 3 , which is the default value. |

| Parameter | Description | Values |
|---------------------------------------|---|---|
| fips | <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> <div>  <div> <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p> </div> </div> | false or true |
| 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 |
| networking | The configuration for the pod network provider in the cluster. | Object |
| networking.clusterNetwork | The IP address pools for pods. The default is 10.128.0.0/14 with a host prefix of /23 . | Array of objects |
| networking.clusterNetwork.cidr | Required if you use networking.clusterNetwork . The IP block address pool. | IP network. IP networks are represented as strings using Classless Inter-Domain Routing (CIDR) notation with a traditional IP address or network number, followed by the forward slash (/) character, followed by a decimal value between 0 and 32 that describes the number of significant bits. For example, 10.0.0.0/16 represents IP addresses 10.0.0.0 through 10.0.255.255 . |

| Parameter | Description | Values |
|---|---|---|
| networking.clusterNetwork.hostPrefix | Required if you use networking.clusterNetwork . The prefix size to allocate to each node from the CIDR. For example, 24 would allocate $2^8=256$ addresses to each node. | Integer |
| networking.machineNetwork | The IP address pools for machines. | Array of objects |
| networking.machineNetwork.cidr | Required if you use networking.machineNetwork . An IP address block. The default is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default is 192.168.126.0/24 . | IP network in Classless Inter-Domain Routing (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.networkType | The type of network to install. The default is OpenShiftSDN . | String |
| networking.serviceNetwork | The IP address pools for services. The default is 172.30.0.0/16. | Array of IP networks. IP networks are represented as strings using Classless Inter-Domain Routing (CIDR) notation with a traditional IP address or network number, followed by the forward slash (/) character, followed by a decimal value between 0 and 32 that describes the number of significant bits. For example, 10.0.0.0/16 represents IP addresses 10.0.0.0 through 10.0.255.255 . |
| 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 . |


| Parameter | Description | Values |
|---------------|---|---|
| sshKey | <p>The SSH key to authenticate access to your cluster machines.</p> <div>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p> </div> | For example, sshKey: ssh-ed25519 AAAA... |

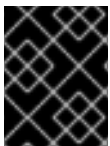
Table 1.10. Optional AWS parameters

| Parameter | Description | Values |
|---|--|--|
| compute.platform.aws.rootVolume.iops | The Input/Output Operations Per Second (IOPS) that is reserved for the root volume. | Integer, for example 4000 . |
| compute.platform.aws.rootVolume.size | The size in GiB of the root volume. | Integer, for example 500 . |
| compute.platform.aws.rootVolume.type | The instance type of the root volume. | Valid AWS EBS instance type , such as io1 . |
| compute.platform.aws.type | The EC2 instance type for the compute machines. | Valid AWS instance type , such as c5.9xlarge . |
| compute.platform.aws.zones | The availability zones where the installation program creates machines for the compute machine pool. If you provide your own VPC, you must provide a subnet in that availability zone. | A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence . |
| compute.aws.region | The AWS region that the installation program creates compute resources in. | Any valid AWS region , such as us-east-1 . |
| controlPlane.platform.aws.type | The EC2 instance type for the control plane machines. | Valid AWS instance type , such as c5.9xlarge . |

| Parameter | Description | Values |
|--|---|---|
| controlPlane.platform.aws.zones | The availability zones where the installation program creates machines for the control plane machine pool. | A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence . |
| controlPlane.aws.region | The AWS region that the installation program creates control plane resources in. | Valid AWS region , such as us-east-1 . |
| platform.aws.userTags | A map of keys and values that the installation program adds as tags to all resources that it creates. | Any valid YAML map, such as key value pairs in the <key>: <value> format. For more information about AWS tags, see Tagging Your Amazon EC2 Resources in the AWS documentation. |
| platform.aws.subnets | If you provide the VPC instead of allowing the installation program to create the VPC for you, specify the subnet for the cluster to use. The subnet must be part of the same machineNetwork[].cidr ranges that you specify. For a standard cluster, specify a public and a private subnet for each availability zone. For a private cluster, specify a private subnet for each availability zone. | Valid subnet IDs. |

1.5.6.2. Sample customized `install-config.yaml` file for AWS

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



IMPORTANT

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

```

apiVersion: v1
baseDomain: example.com ❶
controlPlane: ❷
  hyperthreading: Enabled ❸ ❹
  name: master
  platform:
    aws:
      zones:
        - us-west-2a

```

```

- us-west-2b
rootVolume:
  iops: 4000
  size: 500
  type: io1
  type: m5.xlarge 5
replicas: 3
compute: 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 8
        type: c5.4xlarge
      zones:
        - us-west-2c
      replicas: 3
  metadata:
    name: test-cluster 9
  networking:
    clusterNetwork:
      - cidr: 10.128.0.0/14
        hostPrefix: 23
    machineNetwork:
      - cidr: 10.0.0.0/16
    networkType: OpenShiftSDN
    serviceNetwork:
      - 172.30.0.0/16
  platform:
    aws:
      region: us-west-2 10
      userTags:
        adminContact: jdoe
        costCenter: 7536
      subnets: 11
      - subnet-1
      - subnet-2
      - subnet-3
  fips: false 12
  sshKey: ssh-ed25519 AAAA... 13
  pullSecret: '{"auths": ...}' 14

```

1 9 10 14 Required. The installation program prompts you for this value.

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

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

control plane pool is used.

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



IMPORTANT

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

- 8** To configure faster storage for etcd, especially for larger clusters, set the storage type as **io1** and set **iops** to **2000**.
- 11** If you provide your own VPC, specify subnets for each availability zone that your cluster uses.
- 12** 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.
- 13** You can optionally provide the **sshKey** value that you use to access the machines in your cluster.



NOTE

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

1.5.6.3. Configuring the cluster-wide proxy during installation

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

Prerequisites

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



NOTE

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

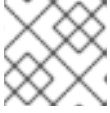
For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

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

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass 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 **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.

**NOTE**

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

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

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

**NOTE**

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

1.5.7. 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

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

Procedure

1. Run the installation program:

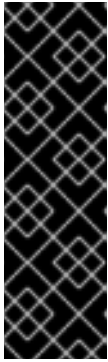
```
$ ./openshift-install create cluster --dir=<installation_directory> \ ❶
--log-level=info ❷
```

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

**NOTE**

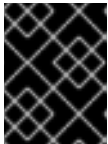
If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

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



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, 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.



IMPORTANT

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

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

1.5.8. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order 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 4.5. Download and install the new version of **oc**.

1.5.8.1. Installing the CLI on Linux

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

Procedure

- Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
- Select your infrastructure provider, and, if applicable, your installation type.
- In the **Command-line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
- Unpack the archive:

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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


```
$ oc <command>
```

1.5.8.2. Installing the CLI on Windows

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

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command-line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
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
```

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

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

1.5.8.3. Installing the CLI on macOS

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

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command-line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
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
```

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

```
$ oc <command>
```

1.5.9. Logging in to the cluster

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

Prerequisites

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

Procedure

1. Export the **kubeadmin** credentials:

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

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

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

```
$ oc whoami
```

Example output

```
system:admin
```

1.5.10. Next steps

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

1.6. INSTALLING A PRIVATE CLUSTER ON AWS

In OpenShift Container Platform version 4.5, you can install a private cluster into an existing VPC on Amazon Web Services (AWS). 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.

1.6.1. Prerequisites

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



IMPORTANT

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

- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

1.6.2. Private clusters

If your environment does not require an external Internet connection, 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.

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.

Additionally, you must deploy a private cluster from a machine that has access the API services for the cloud you provision to, the hosts on the network that you provision, and to 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.

1.6.2.1. Private clusters in AWS

To create a private cluster on Amazon Web Services (AWS), 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 access from only the private network.

The cluster still requires access to Internet to access the AWS APIs.

The following items are not required or created when you install a private cluster:

- Public subnets
- Public load balancers, which support public ingress
- A public Route 53 zone that matches the **baseDomain** for the cluster

The installation program does use the **baseDomain** that you specify to create a private Route 53 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.

1.6.2.1.1. Limitations

The ability to add public functionality to a private cluster is limited.

- You cannot make the Kubernetes API endpoints public after installation without taking additional actions, including creating public subnets in the VPC for each availability zone in use, creating a public load balancer, and configuring the control plane security groups to allow traffic from Internet on 6443 (Kubernetes API port).
- If you use a public Service type load balancer, you must tag a public subnet in each availability zone with **kubernetes.io/cluster/<cluster-infra-id>: shared** so that AWS can use them to create public load balancers.

1.6.3. About using a custom VPC

In OpenShift Container Platform 4.5, you can deploy a cluster into existing subnets in an existing Amazon Virtual Private Cloud (VPC) in Amazon Web Services (AWS). By deploying OpenShift Container Platform into an existing AWS 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.

Because the installation program cannot know what other components are also in your existing subnets, it cannot choose subnet CIDRs and so forth on your behalf. You must configure networking for the subnets that you install your cluster to yourself.

1.6.3.1. Requirements for using your VPC

The installation program no longer creates the following components:

- Internet gateways
- NAT gateways
- Subnets
- Route tables
- VPCs
- VPC DHCP options
- VPC endpoints

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 must meet the following characteristics:

- The VPC's CIDR block must contain the **Networking.MachineCIDR** range, which is the IP address pool for cluster machines.
- The VPC must not use the **kubernetes.io/cluster/.*: owned** tag.

- You must enable the **enableDnsSupport** and **enableDnsHostnames** attributes in your VPC so that the cluster can use the Route 53 zones that are attached to the VPC to resolve cluster's internal DNS records. See [DNS Support in Your VPC](#) in the AWS documentation.

If you use a cluster with public access, you must create a public and a private subnet for each availability zone that your cluster uses.

The installation program modifies your subnets to add the **kubernetes.io/cluster/.*: shared** tag, so your subnets must have at least one free tag slot available for it. Review the current [Tag Restrictions](#) in the AWS documentation to ensure that the installation program can add a tag to each subnet that you specify.

If you are working in a disconnected environment, you are unable to reach the public IP addresses for EC2 and ELB endpoints. To resolve this, you must create a VPC endpoint and attach it to the subnet that the clusters are using. The endpoints should be named as follows:

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

Required VPC components

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

| Component | AWS type | Description |
|------------------|---|--|
| VPC | <ul style="list-style-type: none"> • AWS::EC2::VPC • AWS::EC2::VPCEndpoint | You must provide a public VPC for the cluster to use. The VPC uses an endpoint that references the route tables for each subnet to improve communication with the registry that is hosted in S3. |
| Public subnets | <ul style="list-style-type: none"> • AWS::EC2::Subnet • AWS::EC2::SubnetNetworkACLAssociation | Your VPC must have public subnets for between 1 and 3 availability zones and associate them with appropriate Ingress rules. |
| Internet gateway | <ul style="list-style-type: none"> • AWS::EC2::InternetGateway • AWS::EC2::VPCGatewayAttachment • AWS::EC2::RouteTable • AWS::EC2::Route • AWS::EC2::SubnetRouteTableAssociation • AWS::EC2::NatGateway • AWS::EC2::EIP | You must have a public Internet gateway, with public routes, attached to the VPC. In the provided templates, each public subnet has a NAT gateway with an EIP address. These NAT gateways allow cluster resources, like private subnet instances, to reach the Internet and are not required for some restricted network or proxy scenarios. |

| Component | AWS type | Description | |
|------------------------|--|--|----------------------------|
| Network access control | <ul style="list-style-type: none"> ● AWS::EC2::NetworkAcl ● AWS::EC2::NetworkAclEntry | You must allow the VPC to access the following ports: | |
| | | Port | Reason |
| | | 80 | Inbound HTTP traffic |
| | | 443 | Inbound HTTPS traffic |
| | | 22 | Inbound SSH traffic |
| | | 1024 - 65535 | Inbound ephemeral traffic |
| | | 0 - 65535 | Outbound ephemeral traffic |
| Private subnets | <ul style="list-style-type: none"> ● AWS::EC2::Subnet ● AWS::EC2::RouteTable ● AWS::EC2::SubnetRouteTableAssociation | Your VPC can have private subnets. The provided CloudFormation templates can create private subnets for between 1 and 3 availability zones. If you use private subnets, you must provide appropriate routes and tables for them. | |

1.6.3.2. 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 private subnets.
- The subnet CIDRs belong to the machine CIDR that you specified.
- You provide subnets for each availability zone. Each availability zone contains no more than one public and one private subnet. If you use a private cluster, provide only a private subnet for each availability zone. Otherwise, provide exactly one public and private subnet for each availability zone.
- You provide a public subnet for each private subnet availability zone. Machines are not provisioned in availability zones that you do not provide private subnets for.

If you destroy a cluster that uses an existing VPC, the VPC is not deleted. When you remove the OpenShift Container Platform cluster from a VPC, the **kubernetes.io/cluster/.*: shared** tag is removed from the subnets that it used.

1.6.3.3. 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 resource 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 AWS 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 ELBs, security groups, S3 buckets, and nodes.

1.6.3.4. 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 from 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.

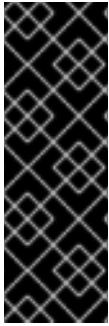
1.6.4. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

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

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page 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.

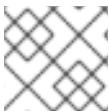


IMPORTANT

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 content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

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



NOTE

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

Procedure

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

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

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

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

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

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

Example output

```
Agent pid 31874
```

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

-


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

Example output

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

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

Next steps

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

1.6.6. Obtaining the installation program

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

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

Procedure

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



IMPORTANT

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



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

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

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

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services

that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

1.6.7. Manually creating the installation configuration file

For installations of a private OpenShift Container Platform cluster that are only accessible from an internal network and are not visible to the Internet, you must manually generate your installation configuration file.

Prerequisites

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

Procedure

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

```
$ mkdir <installation_directory>
```



IMPORTANT

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

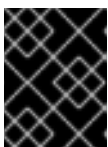
2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.



NOTE

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

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



IMPORTANT

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

1.6.7.1. Installation configuration parameters

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

**NOTE**

After installation, you cannot modify these parameters in the **install-config.yaml** file.

**IMPORTANT**

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.


Table 1.11. Required parameters


| Parameter | Description | Values |
|----------------------|--|---|
| apiVersion | The API version for the install-config.yaml content. The current version is v1 . The installer 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 |
| 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 . |


| Parameter | Description | Values |
|-------------------|---|---|
| platform | The configuration for the specific platform upon which to perform the installation: aws , baremetal , azure , openstack , ovirt , vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform. | Object |
| pullSecret | Get this pull secret from https://cloud.redhat.com/openshift/install/pull-secret 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> |


Table 1.12. 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 |
| compute | The configuration for the machines that comprise the compute nodes. | Array of machine-pool objects. For details, see the following "Machine-pool" table. |
| compute.architecture | Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default). | String |

| Parameter | Description | Values |
|----------------------------------|---|---|
| compute.hyperthreading | <p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> | Enabled or Disabled |
| compute.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, openstack, ovirt, 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 . |
| controlPlane | The configuration for the machines that comprise the control plane. | Array of MachinePool objects. For details, see the following "Machine-pool" table. |
| controlPlane.architecture | Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default). | String |

| Parameter | Description | Values |
|-------------------------------------|---|--|
| controlPlane.hypert hreading | <p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> | Enabled or Disabled |
| controlPlane.name | Required if you use controlPlane . The name of the machine pool. | master |
| controlPlane.platfor m | 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, openstack, ovirt, vsphere , or {} |
| controlPlane.replica s | The number of control plane machines to provision. | The only supported value is 3 , which is the default value. |

| Parameter | Description | Values |
|---------------------------------------|--|---|
| fips | <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> <div>  <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p> </div> | false or true |
| 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 |
| networking | The configuration for the pod network provider in the cluster. | Object |
| networking.clusterNetwork | The IP address pools for pods. The default is 10.128.0.0/14 with a host prefix of /23 . | Array of objects |
| networking.clusterNetwork.cidr | Required if you use networking.clusterNetwork . The IP block address pool. | IP network. IP networks are represented as strings using Classless Inter-Domain Routing (CIDR) notation with a traditional IP address or network number, followed by the forward slash (/) character, followed by a decimal value between 0 and 32 that describes the number of significant bits. For example, 10.0.0.0/16 represents IP addresses 10.0.0.0 through 10.0.255.255 . |

| Parameter | Description | Values |
|---|---|---|
| networking.clusterNetwork.hostPrefix | Required if you use networking.clusterNetwork . The prefix size to allocate to each node from the CIDR. For example, 24 would allocate $2^8=256$ addresses to each node. | Integer |
| networking.machineNetwork | The IP address pools for machines. | Array of objects |
| networking.machineNetwork.cidr | Required if you use networking.machineNetwork . An IP address block. The default is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default is 192.168.126.0/24 . | IP network in Classless Inter-Domain Routing (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.networkType | The type of network to install. The default is OpenShiftSDN . | String |
| networking.serviceNetwork | The IP address pools for services. The default is 172.30.0.0/16. | Array of IP networks. IP networks are represented as strings using Classless Inter-Domain Routing (CIDR) notation with a traditional IP address or network number, followed by the forward slash (/) character, followed by a decimal value between 0 and 32 that describes the number of significant bits. For example, 10.0.0.0/16 represents IP addresses 10.0.0.0 through 10.0.255.255 . |
| 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 . |


| Parameter | Description | Values |
|---------------|---|---|
| sshKey | <p>The SSH key to authenticate access to your cluster machines.</p> <div>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p> </div> | For example, sshKey: ssh-ed25519 AAAA... |

Table 1.13. Optional AWS parameters

| Parameter | Description | Values |
|---|--|--|
| compute.platform.aws.rootVolume.iops | The Input/Output Operations Per Second (IOPS) that is reserved for the root volume. | Integer, for example 4000 . |
| compute.platform.aws.rootVolume.size | The size in GiB of the root volume. | Integer, for example 500 . |
| compute.platform.aws.rootVolume.type | The instance type of the root volume. | Valid AWS EBS instance type , such as io1 . |
| compute.platform.aws.type | The EC2 instance type for the compute machines. | Valid AWS instance type , such as c5.9xlarge . |
| compute.platform.aws.zones | The availability zones where the installation program creates machines for the compute machine pool. If you provide your own VPC, you must provide a subnet in that availability zone. | A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence . |
| compute.aws.region | The AWS region that the installation program creates compute resources in. | Any valid AWS region , such as us-east-1 . |
| controlPlane.platform.aws.type | The EC2 instance type for the control plane machines. | Valid AWS instance type , such as c5.9xlarge . |

| Parameter | Description | Values |
|--|---|---|
| controlPlane.platform.aws.zones | The availability zones where the installation program creates machines for the control plane machine pool. | A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence . |
| controlPlane.aws.region | The AWS region that the installation program creates control plane resources in. | Valid AWS region , such as us-east-1 . |
| platform.aws.userTags | A map of keys and values that the installation program adds as tags to all resources that it creates. | Any valid YAML map, such as key value pairs in the <key>: <value> format. For more information about AWS tags, see Tagging Your Amazon EC2 Resources in the AWS documentation. |
| platform.aws.subnets | If you provide the VPC instead of allowing the installation program to create the VPC for you, specify the subnet for the cluster to use. The subnet must be part of the same machineNetwork[].cidr ranges that you specify. For a standard cluster, specify a public and a private subnet for each availability zone. For a private cluster, specify a private subnet for each availability zone. | Valid subnet IDs. |

1.6.7.2. Sample customized `install-config.yaml` file for AWS

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



IMPORTANT

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

```

apiVersion: v1
baseDomain: example.com ❶
controlPlane: ❷
  hyperthreading: Enabled ❸ ❹
  name: master
  platform:
    aws:
      zones:
        - us-west-2a

```

```

- us-west-2b
rootVolume:
  iops: 4000
  size: 500
  type: io1
  type: m5.xlarge 5
replicas: 3
compute: 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 8
        type: c5.4xlarge
      zones:
        - us-west-2c
      replicas: 3
  metadata:
    name: test-cluster 9
  networking:
    clusterNetwork:
      - cidr: 10.128.0.0/14
        hostPrefix: 23
    machineNetwork:
      - cidr: 10.0.0.0/16
    networkType: OpenShiftSDN
    serviceNetwork:
      - 172.30.0.0/16
  platform:
    aws:
      region: us-west-2 10
      userTags:
        adminContact: jdoe
        costCenter: 7536
      subnets: 11
      - subnet-1
      - subnet-2
      - subnet-3
  fips: false 12
  sshKey: ssh-ed25519 AAAA... 13
  publish: Internal 14
  pullSecret: '{"auths": ...}' 15

```

1 9 10 15 Required. The installation program prompts you for this value.

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

3 7 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift

Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

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



IMPORTANT

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

- 8** To configure faster storage for etcd, especially for larger clusters, set the storage type as **io1** and set **iops** to **2000**.
- 11** If you provide your own VPC, specify subnets for each availability zone that your cluster uses.
- 12** 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.
- 13** You can optionally provide the **sshKey** value that you use to access the machines in your cluster.



NOTE

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

- 14** 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**.

1.6.7.3. Configuring the cluster-wide proxy during installation

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

Prerequisites

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



NOTE

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

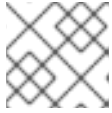
For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

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

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass 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 **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.

**NOTE**

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

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

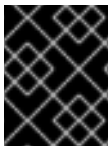
The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

**NOTE**

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

1.6.8. 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

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

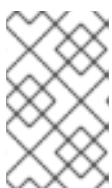
Procedure

1. Run the installation program:

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

1 For **<installation_directory>**, specify the

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

**NOTE**

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

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



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, 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.



IMPORTANT

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

1.6.9. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order 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 4.5. Download and install the new version of **oc**.

1.6.9.1. Installing the CLI on Linux

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

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command-line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvfz <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

1.6.9.2. Installing the CLI on Windows

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

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command-line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
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
```

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

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

1.6.9.3. Installing the CLI on macOS

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

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command-line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
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
```

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

```
$ oc <command>
```

1.6.10. Logging in to the cluster

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

Prerequisites

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

Procedure

1. Export the **kubeadmin** credentials:

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

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

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

```
$ oc whoami
```

Example output

```
system:admin
```

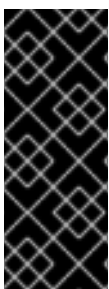
1.6.11. Next steps

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

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

In OpenShift Container Platform version 4.5, you can install a cluster on Amazon Web Services (AWS) that uses infrastructure that you provide.

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



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 CloudFormation 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.

1.7.1. Prerequisites

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



IMPORTANT

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

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



NOTE

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

- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

1.7.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

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

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page 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 content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

1.7.3. Required AWS infrastructure components

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

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

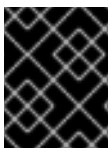
You can use the provided Cloud Formation templates to create this infrastructure, you can manually create the components, or you can reuse existing infrastructure that meets the cluster requirements. Review the Cloud Formation templates for more details about how the components interrelate.

1.7.3.1. Cluster machines

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

- A bootstrap machine. This machine is required during installation, but you can remove it after your cluster deploys.
- Three control plane machines. The control plane machines are not governed by a machine set.
- Compute machines. You must create at least two compute machines, which are also known as worker machines, during installation. These machines are not governed by a machine set.

You can use the following instance types for the cluster machines with the provided Cloud Formation templates.



IMPORTANT

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

Table 1.14. Instance types for machines

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

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

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

1.7.3.2. 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.

1.7.3.3. Other infrastructure components

- A VPC
- DNS entries
- Load balancers (classic or network) and listeners
- A public and a private Route 53 zone
- Security groups
- IAM roles
- S3 buckets

If you are working in a disconnected environment, you are unable to reach the public IP addresses for EC2 and ELB endpoints. To resolve this, you must create a VPC endpoint and attach it to the subnet that the clusters are using. The endpoints should be named as follows:

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

Required VPC components

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

| Component | AWS type | Description |
|----------------|---|--|
| VPC | <ul style="list-style-type: none"> • AWS::EC2::VPC • AWS::EC2::VPCEndpoint | You must provide a public VPC for the cluster to use. The VPC uses an endpoint that references the route tables for each subnet to improve communication with the registry that is hosted in S3. |
| Public subnets | <ul style="list-style-type: none"> • AWS::EC2::Subnet • AWS::EC2::SubnetNetworkACLAssociation | Your VPC must have public subnets for between 1 and 3 availability zones and associate them with appropriate Ingress rules. |

| Component | AWS type | Description | |
|------------------------|---|--|----------------------------|
| Internet gateway | <ul style="list-style-type: none">● AWS::EC2::InternetGateway● AWS::EC2::VPCGatewayAttachment● AWS::EC2::RouteTable● AWS::EC2::Route● AWS::EC2::SubnetRouteTableAssociation● AWS::EC2::NatGateway● AWS::EC2::EIP | You must have a public Internet gateway, with public routes, attached to the VPC. In the provided templates, each public subnet has a NAT gateway with an EIP address. These NAT gateways allow cluster resources, like private subnet instances, to reach the Internet and are not required for some restricted network or proxy scenarios. | |
| Network access control | <ul style="list-style-type: none">● AWS::EC2::NetworkAcl● AWS::EC2::NetworkAclEntry | You must allow the VPC to access the following ports: | |
| | | Port | Reason |
| | | 80 | Inbound HTTP traffic |
| | | 443 | Inbound HTTPS traffic |
| | | 22 | Inbound SSH traffic |
| | | 1024 - 65535 | Inbound ephemeral traffic |
| | | 0 - 65535 | Outbound ephemeral traffic |
| Private subnets | <ul style="list-style-type: none">● AWS::EC2::Subnet● AWS::EC2::RouteTable● AWS::EC2::SubnetRouteTableAssociation | Your VPC can have private subnets. The provided CloudFormation templates can create private subnets for between 1 and 3 availability zones. If you use private subnets, you must provide appropriate routes and tables for them. | |

Required DNS and load balancing components

Your DNS and load balancer configuration needs to use a public hosted zone and can use a private hosted zone similar to the one that the installation program uses if it provisions the cluster's infrastructure. You must create a DNS entry that resolves to your load balancer. An entry for **api**.

<cluster_name>.<domain> must point to the external load balancer, and an entry for **api-int.<cluster_name>.<domain>** must point to the internal load balancer.

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

| Component | AWS type | Description |
|----------------------------|--|--|
| DNS | AWS::Route53::HostedZone | The hosted zone for your internal DNS. |
| etcd record sets | AWS::Route53::RecordSet | The registration records for etcd for your control plane machines. |
| Public load balancer | AWS::ElasticLoadBalancingV2::LoadBalancer | The load balancer for your public subnets. |
| External API server record | AWS::Route53::RecordSetGroup | Alias records for the external API server. |
| External listener | AWS::ElasticLoadBalancingV2::Listener | A listener on port 6443 for the external load balancer. |
| External target group | AWS::ElasticLoadBalancingV2::TargetGroup | The target group for the external load balancer. |
| Private load balancer | AWS::ElasticLoadBalancingV2::LoadBalancer | The load balancer for your private subnets. |
| Internal API server record | AWS::Route53::RecordSetGroup | Alias records for the internal API server. |
| Internal listener | AWS::ElasticLoadBalancingV2::Listener | A listener on port 22623 for the internal load balancer. |

| Component | AWS type | Description |
|-----------------------|---|---|
| Internal target group | AWS::ElasticLoadBalancingV2::TargetGroup | The target group for the internal load balancer. |
| Internal listener | AWS::ElasticLoadBalancingV2::Listener | A listener on port 6443 for the internal load balancer. |
| Internal target group | AWS::ElasticLoadBalancingV2::TargetGroup | The target group for the internal load balancer. |

Security groups

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

| Group | Type | IP Protocol | Port range |
|-------------------------------|--------------------------------|-------------|--------------|
| MasterSecurityGroup | AWS::EC2::SecurityGroup | icmp | 0 |
| | | tcp | 22 |
| | | tcp | 6443 |
| | | tcp | 22623 |
| WorkerSecurityGroup | AWS::EC2::SecurityGroup | icmp | 0 |
| | | tcp | 22 |
| BootstrapSecurityGroup | AWS::EC2::SecurityGroup | tcp | 22 |
| | | tcp | 19531 |

Control plane Ingress

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

| Ingress group | Description | IP protocol | Port range |
|--------------------------|-------------|-------------|-------------------|
| MasterIngressEtcd | etcd | tcp | 2379- 2380 |

| Ingress group | Description | IP protocol | Port range |
|---|---|-------------|----------------------|
| MasterIngressVxlan | Vxlan packets | udp | 4789 |
| MasterIngressWorkerVxlan | Vxlan packets | udp | 4789 |
| MasterIngressInternal | Internal cluster communication and Kubernetes proxy metrics | tcp | 9000 - 9999 |
| MasterIngressWorkerInternal | Internal cluster communication | tcp | 9000 - 9999 |
| MasterIngressKube | Kubernetes kubelet, scheduler and controller manager | tcp | 10250 - 10259 |
| MasterIngressWorkerKube | Kubernetes kubelet, scheduler and controller manager | tcp | 10250 - 10259 |
| MasterIngressIngressServices | Kubernetes Ingress services | tcp | 30000 - 32767 |
| MasterIngressWorkerIngressServices | Kubernetes Ingress services | tcp | 30000 - 32767 |

Worker Ingress

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

| Ingress group | Description | IP protocol | Port range |
|------------------------------------|--------------------------------|-------------|--------------------|
| WorkerIngressVxlan | Vxlan packets | udp | 4789 |
| WorkerIngressWorkerVxlan | Vxlan packets | udp | 4789 |
| WorkerIngressInternal | Internal cluster communication | tcp | 9000 - 9999 |
| WorkerIngressWorkerInternal | Internal cluster communication | tcp | 9000 - 9999 |

| Ingress group | Description | IP protocol | Port range |
|---|---|-------------|----------------------|
| WorkerIngress Kube | Kubernetes kubelet, scheduler, and controller manager | tcp | 10250 |
| WorkerIngress WorkerKube | Kubernetes kubelet, scheduler, and controller manager | tcp | 10250 |
| WorkerIngress IngressServices | Kubernetes Ingress services | tcp | 30000 - 32767 |
| WorkerIngress WorkerIngress Services | Kubernetes Ingress services | tcp | 30000 - 32767 |

Roles and instance profiles

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

| Role | Effect | Action | Resource |
|-----------|--------------|-------------------------------|----------|
| Master | Allow | ec2:* | * |
| | Allow | elasticloadbalancing:* | * |
| | Allow | iam:PassRole | * |
| | Allow | s3:GetObject | * |
| Worker | Allow | ec2:Describe* | * |
| Bootstrap | Allow | ec2:Describe* | * |
| | Allow | ec2:AttachVolume | * |
| | Allow | ec2:DetachVolume | * |

1.7.3.4. Required AWS permissions

When you attach the **AdministratorAccess** policy to the IAM user that you create in Amazon Web Services (AWS), you grant that user all of the required permissions. To deploy all components of an OpenShift Container Platform cluster, the IAM user requires the following permissions:

Example 1.11. Required EC2 permissions for installation

- **tag:TagResources**
- **tag:UntagResources**
- **ec2:AllocateAddress**
- **ec2:AssociateAddress**
- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2>CreateNetworkInterface**
- **ec2:AttachNetworkInterface**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2>DeleteSecurityGroup**
- **ec2>DeleteSnapshot**
- **ec2>DeleteTags**
- **ec2:DeregisterImage**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInternetGateways**
- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**
- **ec2:DescribeNetworkAcls**
- **ec2:DescribeNetworkInterfaces**

- **ec2:DescribePrefixLists**
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcAttribute**
- **ec2:DescribeVpcClassicLink**
- **ec2:DescribeVpcClassicLinkDnsSupport**
- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**
- **ec2:GetEbsDefaultKmsKeyId**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyNetworkInterfaceAttribute**
- **ec2:ReleaseAddress**
- **ec2:RevokeSecurityGroupEgress**
- **ec2:RevokeSecurityGroupIngress**
- **ec2:RunInstances**
- **ec2:TerminateInstances**

Example 1.12. Required permissions for creating network resources during installation

- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**
- **ec2:CreateDhcpOptions**
- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**

- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**



NOTE

If you use an existing VPC, your account does not require these permissions for creating network resources.

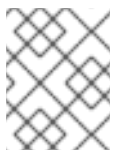
Example 1.13. Required Elastic Load Balancing permissions for installation

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

- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:RegisterTargets**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**

Example 1.14. Required IAM permissions for installation

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



NOTE

If you have not created an elastic load balancer (ELB) in your AWS account, the IAM user also requires the **iam:CreateServiceLinkedRole** permission.

Example 1.15. Required Route 53 permissions for installation

- **route53:ChangeResourceRecordSets**

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

Example 1.16. Required S3 permissions for installation

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

- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

Example 1.17. S3 permissions that cluster Operators require

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

Example 1.18. Required permissions to delete base cluster resources

- **autoscaling:DescribeAutoScalingGroups**
- **ec2:DeleteNetworkInterface**
- **ec2:DeleteVolume**
- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**
- **s3:DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

Example 1.19. Required permissions to delete network resources

- **ec2:DeleteDhcpOptions**

- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteRoute**
- **ec2:DeleteRouteTable**
- **ec2:DeleteSubnet**
- **ec2:DeleteVpc**
- **ec2:DeleteVpcEndpoints**
- **ec2:DetachInternetGateway**
- **ec2:DisassociateRouteTable**
- **ec2:ReplaceRouteTableAssociation**



NOTE

If you use an existing VPC, your account does not require these permissions to delete network resources.

Example 1.20. Additional IAM and S3 permissions that are required to create manifests

- **iam:CreateAccessKey**
- **iam:CreateUser**
- **iam>DeleteAccessKey**
- **iam>DeleteUser**
- **iam>DeleteUserPolicy**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **s3:PutBucketPublicAccessBlock**
- **s3:GetBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**
- **s3:HeadBucket**

- **s3:ListBucketMultipartUploads**
- **s3:AbortMultipartUpload**

1.7.4. Obtaining the installation program

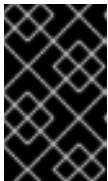
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

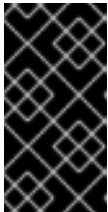
Procedure

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



IMPORTANT

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



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

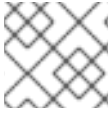
3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

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

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

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

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.

**NOTE**

In a production environment, you require disaster recovery and debugging.

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

**NOTE**

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

Procedure

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

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

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

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

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

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

Example output

```
Agent pid 31874
```

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

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

Example output

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

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

Next steps

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

1.7.6. Creating the installation files for AWS

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

1.7.6.1. Creating the installation configuration file

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

Prerequisites

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

Procedure

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

- a. Run the following command:

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

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



IMPORTANT

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

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



NOTE

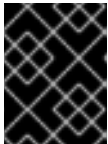
For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **aws** as the platform to target.
- iii. If you do not have an AWS profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.

- iv. Select the AWS region to deploy the cluster to.
 - v. Select the base domain for the Route 53 service that you configured for your cluster.
 - vi. Enter a descriptive name for your cluster.
 - vii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Edit the **install-config.yaml** file to set the number of compute replicas, which are also known as worker replicas, to **0**, as shown in the following **compute** stanza:

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

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



IMPORTANT

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

1.7.6.2. Configuring the cluster-wide proxy during installation

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

Prerequisites

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



NOTE

The **Proxy** object **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: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
...

```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass 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 **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.

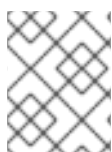


NOTE

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

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

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

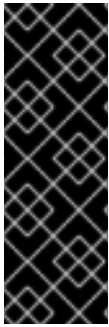


NOTE

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

1.7.6.3. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, 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.

Prerequisites

- Obtain the OpenShift Container Platform installation program.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

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

Example output

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

- 1** For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

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

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

```
$ rm -f <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 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.

4. Modify the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes manifest file to prevent 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 set its value to **False**.
 - c. Save and exit the file.
5. Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the **privateZone** and **publicZone** sections from the `<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: ❶
    id: mycluster-100419-private-zone
  publicZone: ❷
    id: example.openshift.com
status: {}
```

- ❶ ❷ Remove this section completely.

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

6. Obtain the Ignition config files:

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

- ❶ For `<installation_directory>`, specify the same installation directory.

The following files are generated in the directory:

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

1.7.7. Extracting the infrastructure name

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

Prerequisites

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

Procedure

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

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

- 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.

1.7.8. Creating a VPC in AWS

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



NOTE

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

Prerequisites

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

Procedure

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

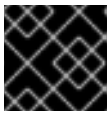
```
[
  {
    "ParameterKey": "VpcCidr", 1
    "ParameterValue": "10.0.0.0/16" 2
  },
  {
```

```

    "ParameterKey": "AvailabilityZoneCount", 3
    "ParameterValue": "1" 4
  },
  {
    "ParameterKey": "SubnetBits", 5
    "ParameterValue": "12" 6
  }
]

```

- 1 The CIDR block for the VPC.
 - 2 Specify a CIDR block in the format **x.x.x.x/16-24**.
 - 3 The number of availability zones to deploy the VPC in.
 - 4 Specify an integer between **1** and **3**.
 - 5 The size of each subnet in each availability zone.
 - 6 Specify an integer between **5** and **13**, where **5** is **/27** and **13** is **/19**.
2. Copy the template from the **CloudFormation template for the VPC** section of this topic and save it as a YAML file on your computer. This template describes the VPC that your cluster requires.
 3. Launch the template:



IMPORTANT

You must enter the command on a single line.

```

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

```

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

```

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

```

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

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

1.7.8.1. CloudFormation template for the VPC

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

Example 1.21. CloudFormation template for the VPC

```

AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice VPC with 1-3 AZs

Parameters:
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\.(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  AvailabilityZoneCount:
    ConstraintDescription: "The number of availability zones. (Min: 1, Max: 3)"
    MinValue: 1
    MaxValue: 3
    Default: 1
    Description: "How many AZs to create VPC subnets for. (Min: 1, Max: 3)"
    Type: Number
  SubnetBits:
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/19-27.
    MinValue: 5
    MaxValue: 13
    Default: 12
    Description: "Size of each subnet to create within the availability zones. (Min: 5 = /27, Max: 13 = /19)"
    Type: Number

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      - Label:
          default: "Network Configuration"
        Parameters:
          - VpcCidr
          - SubnetBits
      - Label:
          default: "Availability Zones"
        Parameters:

```

```

- AvailabilityZoneCount
ParameterLabels:
  AvailabilityZoneCount:
    default: "Availability Zone Count"
  VpcCidr:
    default: "VPC CIDR"
  SubnetBits:
    default: "Bits Per Subnet"

```

Conditions:

```

DoAz3: !Equals [3, !Ref AvailabilityZoneCount]
DoAz2: !Or [!Equals [2, !Ref AvailabilityZoneCount], Condition: DoAz3]

```

Resources:

```

VPC:
  Type: "AWS::EC2::VPC"
  Properties:
    EnableDnsSupport: "true"
    EnableDnsHostnames: "true"
    CidrBlock: !Ref VpcCidr
PublicSubnet:
  Type: "AWS::EC2::Subnet"
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [0, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
PublicSubnet2:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
PublicSubnet3:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
InternetGateway:
  Type: "AWS::EC2::InternetGateway"
GatewayToInternet:
  Type: "AWS::EC2::VPCGatewayAttachment"
  Properties:
    VpcId: !Ref VPC
    InternetGatewayId: !Ref InternetGateway
PublicRouteTable:
  Type: "AWS::EC2::RouteTable"
  Properties:

```

```

    VpcId: !Ref VPC
PublicRoute:
  Type: "AWS::EC2::Route"
  DependsOn: GatewayToInternet
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway
PublicSubnetRouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PublicSubnet
    RouteTableId: !Ref PublicRouteTable
PublicSubnetRouteTableAssociation2:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz2
  Properties:
    SubnetId: !Ref PublicSubnet2
    RouteTableId: !Ref PublicRouteTable
PublicSubnetRouteTableAssociation3:
  Condition: DoAz3
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PublicSubnet3
    RouteTableId: !Ref PublicRouteTable
PrivateSubnet:
  Type: "AWS::EC2::Subnet"
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [3, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable:
  Type: "AWS::EC2::RouteTable"
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PrivateSubnet
    RouteTableId: !Ref PrivateRouteTable
NAT:
  DependsOn:
    - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Properties:
    AllocationId:
      "Fn::GetAtt":
        - EIP
        - AllocationId
    SubnetId: !Ref PublicSubnet
EIP:
  Type: "AWS::EC2::EIP"
  Properties:
    Domain: vpc

```

```
Route:
  Type: "AWS::EC2::Route"
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT
PrivateSubnet2:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [4, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable2:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation2:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz2
  Properties:
    SubnetId: !Ref PrivateSubnet2
    RouteTableId: !Ref PrivateRouteTable2
NAT2:
  DependsOn:
    - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz2
  Properties:
    AllocationId:
      "Fn::GetAtt":
        - EIP2
        - AllocationId
    SubnetId: !Ref PublicSubnet2
EIP2:
  Type: "AWS::EC2::EIP"
  Condition: DoAz2
  Properties:
    Domain: vpc
Route2:
  Type: "AWS::EC2::Route"
  Condition: DoAz2
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT2
PrivateSubnet3:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz3
```

```

Properties:
  VpcId: !Ref VPC
  CidrBlock: !Select [5, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
  AvailabilityZone: !Select
    - 2
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable3:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation3:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz3
  Properties:
    SubnetId: !Ref PrivateSubnet3
    RouteTableId: !Ref PrivateRouteTable3
NAT3:
  DependsOn:
    - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz3
  Properties:
    AllocationId:
      "Fn::GetAtt":
        - EIP3
        - AllocationId
    SubnetId: !Ref PublicSubnet3
EIP3:
  Type: "AWS::EC2::EIP"
  Condition: DoAz3
  Properties:
    Domain: vpc
Route3:
  Type: "AWS::EC2::Route"
  Condition: DoAz3
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable3
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT3
S3Endpoint:
  Type: AWS::EC2::VPCEndpoint
  Properties:
    PolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Principal: "*"
          Action:
            - "*"
          Resource:
            - "*"
    RouteTableIds:
      - !Ref PublicRouteTable

```

```

- !Ref PrivateRouteTable
- !If [DoAz2, !Ref PrivateRouteTable2, !Ref "AWS::NoValue"]
- !If [DoAz3, !Ref PrivateRouteTable3, !Ref "AWS::NoValue"]
ServiceName: !Join
- "
- - com.amazonaws.
- !Ref 'AWS::Region'
- .s3
VpcId: !Ref VPC

Outputs:
VpcId:
  Description: ID of the new VPC.
  Value: !Ref VPC
PublicSubnetIds:
  Description: Subnet IDs of the public subnets.
  Value:
    !Join [
      ",",
      [!Ref PublicSubnet, !If [DoAz2, !Ref PublicSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PublicSubnet3, !Ref "AWS::NoValue"]]
    ]
PrivateSubnetIds:
  Description: Subnet IDs of the private subnets.
  Value:
    !Join [
      ",",
      [!Ref PrivateSubnet, !If [DoAz2, !Ref PrivateSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PrivateSubnet3, !Ref "AWS::NoValue"]]
    ]

```

1.7.9. Creating networking and load balancing components in AWS

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

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



NOTE

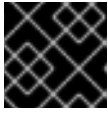
If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

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

Procedure

1. Obtain the Hosted Zone ID for the Route 53 zone that you specified in the **install-config.yaml** file for your cluster. You can obtain this ID from the AWS console or by running the following command:



IMPORTANT

You must enter the command on a single line.

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

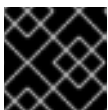
- 1 For the **<route53_domain>**, specify the Route 53 base domain that you used when you generated the **install-config.yaml** file for the cluster.

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

```
[
  {
    "ParameterKey": "ClusterName", 1
    "ParameterValue": "mycluster" 2
  },
  {
    "ParameterKey": "InfrastructureName", 3
    "ParameterValue": "mycluster-<random_string>" 4
  },
  {
    "ParameterKey": "HostedZoneId", 5
    "ParameterValue": "<random_string>" 6
  },
  {
    "ParameterKey": "HostedZoneName", 7
    "ParameterValue": "example.com" 8
  },
  {
    "ParameterKey": "PublicSubnets", 9
    "ParameterValue": "subnet-<random_string>" 10
  },
  {
    "ParameterKey": "PrivateSubnets", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
    "ParameterKey": "VpcId", 13
    "ParameterValue": "vpc-<random_string>" 14
  }
]
```

- 1 A short, representative cluster name to use for host names, etc.

- 2 Specify the cluster name that you used when you generated the **install-config.yaml** file for the cluster.
 - 3 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
 - 4 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
 - 5 The Route 53 public zone ID to register the targets with.
 - 6 Specify the Route 53 public zone ID, which has a format similar to **Z21IXYZABCZ2A4**. You can obtain this value from the AWS console.
 - 7 The Route 53 zone to register the targets with.
 - 8 Specify the Route 53 base domain that you used when you generated the **install-config.yaml** file for the cluster. Do not include the trailing period (.) that is displayed in the AWS console.
 - 9 The public subnets that you created for your VPC.
 - 10 Specify the **PublicSubnetIds** value from the output of the CloudFormation template for the VPC.
 - 11 The private subnets that you created for your VPC.
 - 12 Specify the **PrivateSubnetIds** value from the output of the CloudFormation template for the VPC.
 - 13 The VPC that you created for the cluster.
 - 14 Specify the **VpcId** value from the output of the CloudFormation template for the VPC.
3. Copy the template from the **CloudFormation template for the network and load balancers** section of this topic and save it as a YAML file on your computer. This template describes the networking and load balancing objects that your cluster requires.
 4. Launch the template:



IMPORTANT

You must enter the command on a single line.

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

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

- 3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

5. Confirm that the template components exist:

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

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

| | |
|--------------------------------------|--|
| PrivateHostedZoneId | Hosted zone ID for the private DNS. |
| ExternalApiLoadBalancerName | Full name of the external API load balancer. |
| InternalApiLoadBalancerName | Full name of the internal API load balancer. |
| ApiServerDnsName | Full host name of the API server. |
| RegisterNlbIpTargetsLambda | Lambda ARN useful to help register/deregister IP targets for these load balancers. |
| ExternalApiTargetGroupArn | ARN of external API target group. |
| InternalApiTargetGroupArn | ARN of internal API target group. |
| InternalServiceTargetGroupArn | ARN of internal service target group. |

1.7.9.1. CloudFormation template for the network and load balancers

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

Example 1.22. CloudFormation template for the network and load balancers

AWSTemplateFormatVersion: 2010-09-09

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

Parameters:

ClusterName:

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

MaxLength: 27

MinLength: 1

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

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

Type: String

InfrastructureName:

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

MaxLength: 27

MinLength: 1

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

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

Type: String

HostedZoneId:

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

Type: String

HostedZoneName:

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

Type: String

Default: "example.com"

PublicSubnets:

Description: The internet-facing subnets.

Type: List<AWS::EC2::Subnet::Id>

PrivateSubnets:

Description: The internal subnets.

Type: List<AWS::EC2::Subnet::Id>

VpcId:

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

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

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:

default: "Cluster Information"

Parameters:

- ClusterName

- InfrastructureName

- Label:

default: "Network Configuration"

Parameters:

- VpcId

- PublicSubnets

- PrivateSubnets

- Label:

```

    default: "DNS"
Parameters:
- HostedZoneName
- HostedZoneId
ParameterLabels:
ClusterName:
    default: "Cluster Name"
InfrastructureName:
    default: "Infrastructure Name"
VpcId:
    default: "VPC ID"
PublicSubnets:
    default: "Public Subnets"
PrivateSubnets:
    default: "Private Subnets"
HostedZoneName:
    default: "Public Hosted Zone Name"
HostedZoneId:
    default: "Public Hosted Zone ID"

```

Resources:

ExtApiElb:

```

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

```

IntApiElb:

```

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

```

IntDns:

```

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

```

ExternalApiServerRecord:

```

Type: AWS::Route53::RecordSetGroup
Properties:

```

```

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

```

```

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

```

```

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

```

```

ExternalApiTargetGroup:
Type: AWS::ElasticLoadBalancingV2::TargetGroup
Properties:
  HealthCheckIntervalSeconds: 10
  HealthCheckPath: "/readyz"
  HealthCheckPort: 6443

```

HealthCheckProtocol: HTTPS
 HealthyThresholdCount: 2
 UnhealthyThresholdCount: 2
 Port: 6443
 Protocol: TCP
 TargetType: ip
 VpcId:
 Ref: VpcId
 TargetGroupAttributes:
 - Key: deregistration_delay.timeout_seconds
 Value: 60

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

InternalApiTargetGroup:
 Type: AWS::ElasticLoadBalancingV2::TargetGroup
 Properties:
 HealthCheckIntervalSeconds: 10
 HealthCheckPath: "/readyz"
 HealthCheckPort: 6443
 HealthCheckProtocol: HTTPS
 HealthyThresholdCount: 2
 UnhealthyThresholdCount: 2
 Port: 6443
 Protocol: TCP
 TargetType: ip
 VpcId:
 Ref: VpcId
 TargetGroupAttributes:
 - Key: deregistration_delay.timeout_seconds
 Value: 60

InternalServiceInternalListener:
 Type: AWS::ElasticLoadBalancingV2::Listener
 Properties:
 DefaultActions:
 - Type: forward
 TargetGroupArn:
 Ref: InternalServiceTargetGroup
 LoadBalancerArn:
 Ref: IntApiElb
 Port: 22623
 Protocol: TCP

InternalServiceTargetGroup:
 Type: AWS::ElasticLoadBalancingV2::TargetGroup

Properties:

HealthCheckIntervalSeconds: 10
 HealthCheckPath: "/healthz"
 HealthCheckPort: 22623
 HealthCheckProtocol: HTTPS
 HealthyThresholdCount: 2
 UnhealthyThresholdCount: 2
 Port: 22623
 Protocol: TCP
 TargetType: ip
 VpcId:
 Ref: VpcId
 TargetGroupAttributes:
 - Key: deregistration_delay.timeout_seconds
 Value: 60

RegisterTargetLambdalamRole:

Type: AWS::IAM::Role

Properties:

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

AssumeRolePolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Principal:

Service:

- "lambda.amazonaws.com"

Action:

- "sts:AssumeRole"

Path: "/"

Policies:

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

PolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Action:

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

Resource: !Ref InternalApiTargetGroup

- Effect: "Allow"

Action:

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

Resource: !Ref InternalServiceTargetGroup

- Effect: "Allow"

Action:

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

Resource: !Ref ExternalApiTargetGroup


```

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

```

```

RegisterSubnetTagsLambdalaRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: !Join ["-", [!Ref InfrastructureName, "subnet-tags-lambda-role"]]
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: "Allow"
          Principal:
            Service:
              - "lambda.amazonaws.com"
          Action:
            - "sts:AssumeRole"
    Path: "/"
    Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "subnet-tagging-policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
            - Effect: "Allow"
              Action:
                [
                  "ec2:DeleteTags",
                  "ec2:CreateTags"
                ]
              Resource: "arn:aws:ec2:*:*:subnet/*"
            - Effect: "Allow"
              Action:

```

```

    [
      "ec2:DescribeSubnets",
      "ec2:DescribeTags"
    ]
    Resource: "*"

RegisterSubnetTags:
  Type: "AWS::Lambda::Function"
  Properties:
    Handler: "index.handler"
    Role:
      Fn::GetAtt:
        - "RegisterSubnetTagsLambdaRole"
        - "Arn"
    Code:
      ZipFile: |
        import json
        import boto3
        import cfnresponse
        def handler(event, context):
            ec2_client = boto3.client('ec2')
            if event['RequestType'] == 'Delete':
                for subnet_id in event['ResourceProperties']['Subnets']:
                    ec2_client.delete_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName']}]);
            elif event['RequestType'] == 'Create':
                for subnet_id in event['ResourceProperties']['Subnets']:
                    ec2_client.create_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName'], 'Value': 'shared'}]);
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['InfrastructureName']+event['ResourceProperties']['Subnets'][0])
    Runtime: "python3.7"
    Timeout: 120

RegisterPublicSubnetTags:
  Type: Custom::SubnetRegister
  Properties:
    ServiceToken: !GetAtt RegisterSubnetTags.Arn
    InfrastructureName: !Ref InfrastructureName
    Subnets: !Ref PublicSubnets

RegisterPrivateSubnetTags:
  Type: Custom::SubnetRegister
  Properties:
    ServiceToken: !GetAtt RegisterSubnetTags.Arn
    InfrastructureName: !Ref InfrastructureName
    Subnets: !Ref PrivateSubnets

Outputs:
  PrivateHostedZoneId:
    Description: Hosted zone ID for the private DNS, which is required for private records.
    Value: !Ref IntDns
  ExternalApiLoadBalancerName:
    Description: Full name of the external API load balancer.
    Value: !GetAtt ExtApiElb.LoadBalancerFullName

```

InternalApiLoadBalancerName:
 Description: Full name of the internal API load balancer.
 Value: !GetAtt IntApiElb.LoadBalancerFullName

ApiServerDnsName:
 Description: Full hostname of the API server, which is required for the Ignition config files.
 Value: !Join [".", ["api-int", !Ref ClusterName, !Ref HostedZoneName]]

RegisterNlbTargetsLambda:
 Description: Lambda ARN useful to help register or deregister IP targets for these load balancers.
 Value: !GetAtt RegisterNlbTargets.Arn

ExternalApiTargetGroupArn:
 Description: ARN of the external API target group.
 Value: !Ref ExternalApiTargetGroup

InternalApiTargetGroupArn:
 Description: ARN of the internal API target group.
 Value: !Ref InternalApiTargetGroup

InternalServiceTargetGroupArn:
 Description: ARN of the internal service target group.
 Value: !Ref InternalServiceTargetGroup

1.7.10. Creating security group and roles in AWS

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



NOTE

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

Prerequisites

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

Procedure

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

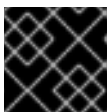
```
[
  {
    "ParameterKey": "InfrastructureName", ❶
    "ParameterValue": "mycluster-<random_string>" ❷
  },
  {
    "ParameterKey": "VpcCidr", ❸
    "ParameterValue": "10.0.0.0/16" ❹
  }
]
```

```

    },
    {
      "ParameterKey": "PrivateSubnets", 5
      "ParameterValue": "subnet-<random_string>" 6
    },
    {
      "ParameterKey": "VpcId", 7
      "ParameterValue": "vpc-<random_string>" 8
    }
  ]
}

```

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



IMPORTANT

You must enter the command on a single line.

```

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

```

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

4. Confirm that the template components exist:

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

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

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

1.7.10.1. CloudFormation template for security objects

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

Example 1.23. CloudFormation template for security objects

AWSTemplateFormatVersion: 2010-09-09

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

Parameters:

InfrastructureName:

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

MaxLength: 27

MinLength: 1

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

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

Type: String

VpcCidr:

AllowedPattern: ^((([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])|([16-9]|2[0-4]))\$

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

Default: 10.0.0.0/16

Description: CIDR block for VPC.

Type: String

VpcId:

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

Type: AWS::EC2::VPC::Id
 PrivateSubnets:
 Description: The internal subnets.
 Type: List<AWS::EC2::Subnet::Id>

Metadata:

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

Resources:

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

WorkerSecurityGroup:
 Type: AWS::EC2::SecurityGroup
 Properties:
 GroupDescription: Cluster Worker Security Group

SecurityGroupIngress:

- IpProtocol: icmp
 - FromPort: 0
 - ToPort: 0
 - CidrIp: !Ref VpcCidr
 - IpProtocol: tcp
 - FromPort: 22
 - ToPort: 22
 - CidrIp: !Ref VpcCidr
- VpcId: !Ref VpcId

MasterIngressEtcd:

Type: AWS::EC2::SecurityGroupIngress

Properties:

- GroupId: !GetAtt MasterSecurityGroup.GroupId
- SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
- Description: etcd
- FromPort: 2379
- ToPort: 2380
- IpProtocol: tcp

MasterIngressVxlan:

Type: AWS::EC2::SecurityGroupIngress

Properties:

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

MasterIngressWorkerVxlan:

Type: AWS::EC2::SecurityGroupIngress

Properties:

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

MasterIngressGeneve:

Type: AWS::EC2::SecurityGroupIngress

Properties:

- GroupId: !GetAtt MasterSecurityGroup.GroupId
- SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
- Description: Geneve packets
- FromPort: 6081
- ToPort: 6081
- IpProtocol: udp

MasterIngressWorkerGeneve:

Type: AWS::EC2::SecurityGroupIngress

Properties:

- GroupId: !GetAtt MasterSecurityGroup.GroupId
- SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Geneve packets

FromPort: 6081

ToPort: 6081

IpProtocol: udp

MasterIngressInternal:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Internal cluster communication

FromPort: 9000

ToPort: 9999

IpProtocol: tcp

MasterIngressWorkerInternal:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Internal cluster communication

FromPort: 9000

ToPort: 9999

IpProtocol: tcp

MasterIngressInternalUDP:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Internal cluster communication

FromPort: 9000

ToPort: 9999

IpProtocol: udp

MasterIngressWorkerInternalUDP:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Internal cluster communication

FromPort: 9000

ToPort: 9999

IpProtocol: udp

MasterIngressKube:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Kubernetes kubelet, scheduler and controller manager

FromPort: 10250

ToPort: 10259

IpProtocol: tcp

MasterIngressWorkerKube:

Type: AWS::EC2::SecurityGroupIngress
 Properties:
 GroupId: !GetAtt MasterSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
 Description: Kubernetes kubelet, scheduler and controller manager
 FromPort: 10250
 ToPort: 10259
 IpProtocol: tcp

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

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

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

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

WorkerIngressVxlan:
 Type: AWS::EC2::SecurityGroupIngress
 Properties:
 GroupId: !GetAtt WorkerSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
 Description: Vxlan packets
 FromPort: 4789

ToPort: 4789
IpProtocol: udp

WorkerIngressMasterVxlan:

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

WorkerIngressGeneve:

Type: AWS::EC2::SecurityGroupIngress
Properties:
 GroupId: !GetAtt WorkerSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
 Description: Geneve packets
 FromPort: 6081
 ToPort: 6081
 IpProtocol: udp

WorkerIngressMasterGeneve:

Type: AWS::EC2::SecurityGroupIngress
Properties:
 GroupId: !GetAtt WorkerSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
 Description: Geneve packets
 FromPort: 6081
 ToPort: 6081
 IpProtocol: udp

WorkerIngressInternal:

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

WorkerIngressMasterInternal:

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

WorkerIngressInternalUDP:

Type: AWS::EC2::SecurityGroupIngress
Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
 Description: Internal cluster communication
 FromPort: 9000
 ToPort: 9999
 IpProtocol: udp

WorkerIngressMasterInternalUDP:

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

WorkerIngressKube:

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

WorkerIngressWorkerKube:

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

WorkerIngressIngressServices:

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

WorkerIngressMasterIngressServices:

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

WorkerIngressIngressServicesUDP:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Kubernetes ingress services

FromPort: 30000

ToPort: 32767

IpProtocol: udp

WorkerIngressMasterIngressServicesUDP:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Kubernetes ingress services

FromPort: 30000

ToPort: 32767

IpProtocol: udp

MasterIamRole:

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Principal:

Service:

- "ec2.amazonaws.com"

Action:

- "sts:AssumeRole"

Policies:

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

PolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Action:

- "ec2:AttachVolume"

- "ec2:AuthorizeSecurityGroupIngress"

- "ec2:CreateSecurityGroup"

- "ec2:CreateTags"

- "ec2:CreateVolume"

- "ec2>DeleteSecurityGroup"

- "ec2>DeleteVolume"

- "ec2:Describe*"

- "ec2:DetachVolume"

- "ec2:ModifyInstanceAttribute"

- "ec2:ModifyVolume"

- "ec2:RevokeSecurityGroupIngress"

- "elasticloadbalancing:AddTags"

- "elasticloadbalancing:AttachLoadBalancerToSubnets"

- "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer"

- "elasticloadbalancing:CreateListener"

- "elasticloadbalancing:CreateLoadBalancer"
- "elasticloadbalancing:CreateLoadBalancerPolicy"
- "elasticloadbalancing:CreateLoadBalancerListeners"
- "elasticloadbalancing:CreateTargetGroup"
- "elasticloadbalancing:ConfigureHealthCheck"
- "elasticloadbalancing>DeleteListener"
- "elasticloadbalancing>DeleteLoadBalancer"
- "elasticloadbalancing>DeleteLoadBalancerListeners"
- "elasticloadbalancing>DeleteTargetGroup"
- "elasticloadbalancing:DeregisterInstancesFromLoadBalancer"
- "elasticloadbalancing:DeregisterTargets"
- "elasticloadbalancing:Describe*"
- "elasticloadbalancing:DetachLoadBalancerFromSubnets"
- "elasticloadbalancing:ModifyListener"
- "elasticloadbalancing:ModifyLoadBalancerAttributes"
- "elasticloadbalancing:ModifyTargetGroup"
- "elasticloadbalancing:ModifyTargetGroupAttributes"
- "elasticloadbalancing:RegisterInstancesWithLoadBalancer"
- "elasticloadbalancing:RegisterTargets"
- "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer"
- "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"
- "kms:DescribeKey"

Resource: ""

MasterInstanceProfile:

Type: "AWS::IAM::InstanceProfile"

Properties:

Roles:

- Ref: "MasterIamRole"

WorkerIamRole:

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Principal:

Service:

- "ec2.amazonaws.com"

Action:

- "sts:AssumeRole"

Policies:

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

PolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Action:

- "ec2:DescribeInstances"

- "ec2:DescribeRegions"

Resource: ""

WorkerInstanceProfile:

Type: "AWS::IAM::InstanceProfile"

Properties:

Roles:
- Ref: "WorkerIamRole"

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

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

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

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

1.7.11. RHCOS AMIs for the AWS infrastructure

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

Table 1.15. RHCOS AMIs

| AWS zone | AWS AMI |
|----------------|-----------------------|
| ap-northeast-1 | ami-0530d04240177f118 |
| ap-northeast-2 | ami-09e4cd700276785d2 |
| ap-south-1 | ami-0754b15d212830477 |
| ap-southeast-1 | ami-03b46cc4b1518c5a8 |
| ap-southeast-2 | ami-0a5b99ab2234a4e6a |
| ca-central-1 | ami-012bc4ee3b6c673bc |
| eu-central-1 | ami-02e08df1201f1c2f8 |
| eu-north-1 | ami-0309c9d2fadcb2d5a |
| eu-west-1 | ami-0bdd69d8e7cd18188 |
| eu-west-2 | ami-0e610e967a62dbdfa |
| eu-west-3 | ami-0e817e26f638a71ac |

| AWS zone | AWS AMI |
|------------|-----------------------|
| me-south-1 | ami-024117d7c87b7ff08 |
| sa-east-1 | ami-08e62f746b94950c1 |
| us-east-1 | ami-077ede5bed2e431ea |
| us-east-2 | ami-0f4ecf819275850dd |
| us-west-1 | ami-0c4990e435bc6c5fe |
| us-west-2 | ami-000d6e92357ac605c |

1.7.12. Creating the bootstrap node in AWS

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



NOTE

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

Prerequisites

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

Procedure

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



IMPORTANT

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



NOTE

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

- a. Create the bucket:

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

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

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

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

- c. Verify that the file uploaded:

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

Example output

```
2019-04-03 16:15:16 314878 bootstrap.ign
```

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

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcOsAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AllowedBootstrapSshCidr", 5
    "ParameterValue": "0.0.0.0/0" 6
  },
  {
    "ParameterKey": "PublicSubnet", 7
    "ParameterValue": "subnet-<random_string>" 8
  }
]
```



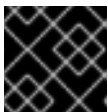
```

    },
    {
      "ParameterKey": "MasterSecurityGroup", 9
      "ParameterValue": "sg-<random_string>" 10
    },
    {
      "ParameterKey": "VpcId", 11
      "ParameterValue": "vpc-<random_string>" 12
    },
    {
      "ParameterKey": "BootstrapIgnitionLocation", 13
      "ParameterValue": "s3://<bucket_name>/bootstrap.ign" 14
    },
    {
      "ParameterKey": "AutoRegisterELB", 15
      "ParameterValue": "yes" 16
    },
    {
      "ParameterKey": "RegisterNlbTargetsLambdaArn", 17
      "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbTargets-<random_string>" 18
    },
    {
      "ParameterKey": "ExternalApiTargetGroupArn", 19
      "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 20
    },
    {
      "ParameterKey": "InternalApiTargetGroupArn", 21
      "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 22
    },
    {
      "ParameterKey": "InternalServiceTargetGroupArn", 23
      "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 24
    }
  ]
}

```

- 1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- 2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
- 3 Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the bootstrap node.
- 4 Specify a valid **AWS::EC2::Image::Id** value.
- 5 CIDR block to allow SSH access to the bootstrap node.
- 6 Specify a CIDR block in the format **x.x.x.x/16-24**.
- 7 The public subnet that is associated with your VPC to launch the bootstrap node into.

- 8 Specify the **PublicSubnetIds** value from the output of the CloudFormation template for the VPC.
 - 9 The master security group ID (for registering temporary rules)
 - 10 Specify the **MasterSecurityGroupId** value from the output of the CloudFormation template for the security group and roles.
 - 11 The VPC created resources will belong to.
 - 12 Specify the **VpcId** value from the output of the CloudFormation template for the VPC.
 - 13 Location to fetch bootstrap Ignition config file from.
 - 14 Specify the S3 bucket and file name in the form **s3://<bucket_name>/bootstrap.ign**.
 - 15 Whether or not to register a network load balancer (NLB).
 - 16 Specify **yes** or **no**. If you specify **yes**, you must provide a Lambda Amazon Resource Name (ARN) value.
 - 17 The ARN for NLB IP target registration lambda group.
 - 18 Specify the **RegisterNlbTargetsLambda** value from the output of the CloudFormation template for DNS and load balancing.
 - 19 The ARN for external API load balancer target group.
 - 20 Specify the **ExternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
 - 21 The ARN for internal API load balancer target group.
 - 22 Specify the **InternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
 - 23 The ARN for internal service load balancer target group.
 - 24 Specify the **InternalServiceTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
3. Copy the template from the **CloudFormation template for the bootstrap machines** section of this topic and save it as a YAML file on your computer. This template describes the bootstrap machine that your cluster requires.
 4. Launch the template:



IMPORTANT

You must enter the command on a single line.

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

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

5. Confirm that the template components exist:

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

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

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

1.7.12.1. CloudFormation template for the bootstrap machine

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

Example 1.24. CloudFormation template for the bootstrap machine

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Bootstrap (EC2 Instance, Security Groups and IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-\_]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
    maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
    used by the cluster.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AllowedBootstrapSshCidr:
    AllowedPattern: ^((([0-9]{1-9}[0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]{1-9}[0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])|(([0-9]{1,3})\.){3}([0-9]{1,3}))$
```

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

Default: 0.0.0.0/0

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

Type: String

PublicSubnet:

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

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

MasterSecurityGroupId:

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

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

VpcId:

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

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

BootstrapIgnitionLocation:

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

Description: Ignition config file location.

Type: String

AutoRegisterELB:

Default: "yes"

AllowedValues:

- "yes"

- "no"

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

Type: String

RegisterNlbTargetsLambdaArn:

Description: ARN for NLB IP target registration lambda.

Type: String

ExternalApiTargetGroupArn:

Description: ARN for external API load balancer target group.

Type: String

InternalApiTargetGroupArn:

Description: ARN for internal API load balancer target group.

Type: String

InternalServiceTargetGroupArn:

Description: ARN for internal service load balancer target group.

Type: String

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:

default: "Cluster Information"

Parameters:

- InfrastructureName

- Label:

default: "Host Information"

Parameters:

- RhcosAmi

- BootstrapIgnitionLocation

- MasterSecurityGroupId

- Label:

default: "Network Configuration"

Parameters:

- VpcId

- AllowedBootstrapSshCidr

- PublicSubnet

```

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

```

```

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

```

```

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

```

```

    Resource: ""
  - Effect: "Allow"
    Action: "s3:GetObject"
    Resource: ""

```

```

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

```

```

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

```

```

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

```

```

RegisterBootstrapApiTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNLBTargetsLambdaArn
    TargetArn: !Ref ExternalApiTargetGroupArn
    TargetIp: !GetAtt BootstrapInstance.PrivateIp

```

```

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

```

```

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

```

```

Outputs:
  BootstrapInstanceId:
    Description: Bootstrap Instance ID.
    Value: !Ref BootstrapInstance

```

```

BootstrapPublicIp:
  Description: The bootstrap node public IP address.
  Value: !GetAtt BootstrapInstance.PublicIp

```

```

BootstrapPrivateIp:
  Description: The bootstrap node private IP address.
  Value: !GetAtt BootstrapInstance.PrivateIp

```

1.7.13. Creating the control plane machines in AWS

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



NOTE

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

Prerequisites

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

Procedure

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

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AutoRegisterDNS", 5
    "ParameterValue": "yes" 6
  },
  {
    "ParameterKey": "PrivateHostedZoneId", 7
    "ParameterValue": "<random_string>" 8
  },
  {
    "ParameterKey": "PrivateHostedZoneName", 9
    "ParameterValue": "mycluster.example.com" 10
  },
  {
    "ParameterKey": "Master0Subnet", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
    "ParameterKey": "Master1Subnet", 13
    "ParameterValue": "subnet-<random_string>" 14
  },
  {
    "ParameterKey": "Master2Subnet", 15
    "ParameterValue": "subnet-<random_string>" 16
  },
  {
    "ParameterKey": "MasterSecurityGroupId", 17
    "ParameterValue": "sg-<random_string>" 18
  },
  {
    "ParameterKey": "IgnitionLocation", 19
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/master"
    20
  },
  {
    "ParameterKey": "CertificateAuthorities", 21
    "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz==" 22
  },
  {
    "ParameterKey": "MasterInstanceProfileName", 23
    "ParameterValue": "<roles_stack>-MasterInstanceProfile-<random_string>" 24
  }
]
```



```

    },
    {
      "ParameterKey": "MasterInstanceType", 25
      "ParameterValue": "m4.xlarge" 26
    },
    {
      "ParameterKey": "AutoRegisterELB", 27
      "ParameterValue": "yes" 28
    },
    {
      "ParameterKey": "RegisterNlbTargetsLambdaArn", 29
      "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbTargets-<random_string>" 30
    },
    {
      "ParameterKey": "ExternalApiTargetGroupArn", 31
      "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 32
    },
    {
      "ParameterKey": "InternalApiTargetGroupArn", 33
      "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 34
    },
    {
      "ParameterKey": "InternalServiceTargetGroupArn", 35
      "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 36
    }
  ]
}

```

- 1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- 2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
- 3 Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the control plane machines.
- 4 Specify an **AWS::EC2::Image::Id** value.
- 5 Whether or not to perform DNS etcd registration.
- 6 Specify **yes** or **no**. If you specify **yes**, you must provide hosted zone information.
- 7 The Route 53 private zone ID to register the etcd targets with.
- 8 Specify the **PrivateHostedZoneId** value from the output of the CloudFormation template for DNS and load balancing.
- 9 The Route 53 zone to register the targets with.
- 10 Specify **<cluster_name>.<domain_name>** where **<domain_name>** is the Route 53 base domain that you used when you generated **install-config.yaml** file for the cluster. Do not include the trailing period (.) that is displayed in the AWS console.

include the training period (.) that is displayed in the AWS console.

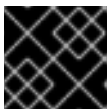
- 11 13 15** A subnet, preferably private, to launch the control plane machines on.
- 12 14 16** Specify a subnet from the **PrivateSubnets** value from the output of the CloudFormation template for DNS and load balancing.
- 17** The master security group ID to associate with master nodes.
- 18** Specify the **MasterSecurityGroupID** value from the output of the CloudFormation template for the security group and roles.
- 19** The location to fetch control plane Ignition config file from.
- 20** Specify the generated Ignition config file location, https://api-int.<cluster_name>.<domain_name>:22623/config/master.
- 21** The base64 encoded certificate authority string to use.
- 22** Specify the value from the **master.ign** file that is in the installation directory. This value is the long string with the format **data:text/plain;charset=utf-8;base64,ABC...xYz==**.
- 23** The IAM profile to associate with master nodes.
- 24** Specify the **MasterInstanceProfile** parameter value from the output of the CloudFormation template for the security group and roles.
- 25** The type of AWS instance to use for the control plane machines.
- 26** Allowed values:
 - **m4.xlarge**
 - **m4.2xlarge**
 - **m4.4xlarge**
 - **m4.8xlarge**
 - **m4.10xlarge**
 - **m4.16xlarge**
 - **c4.2xlarge**
 - **c4.4xlarge**
 - **c4.8xlarge**
 - **r4.xlarge**
 - **r4.2xlarge**
 - **r4.4xlarge**
 - **r4.8xlarge**
 - **r4.16xlarge**



IMPORTANT

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

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



IMPORTANT

You must enter the command on a single line.

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

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

<parameters> is the relative path to and name of the CloudFormation parameters JSON file.

5. Confirm that the template components exist:

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

1.7.13.1. CloudFormation template for control plane machines

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

Example 1.25. CloudFormation template for control plane machines

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 master instances)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-\_]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AutoRegisterDNS:
    Default: "yes"
    AllowedValues:
      - "yes"
      - "no"
    Description: Do you want to invoke DNS etcd registration, which requires Hosted Zone
information?
    Type: String
  PrivateHostedZoneId:
    Description: The Route53 private zone ID to register the etcd targets with, such as
Z21IXYZABCZ2A4.
    Type: String
  PrivateHostedZoneName:
    Description: The Route53 zone to register the targets with, such as cluster.example.com. Omit
the trailing period.
    Type: String
  Master0Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master1Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master2Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
```

MasterSecurityGroupId:

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

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

IgnitionLocation:

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

Description: Ignition config file location.

Type: String

CertificateAuthorities:

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

Description: Base64 encoded certificate authority string to use.

Type: String

MasterInstanceProfileName:

Description: IAM profile to associate with master nodes.

Type: String

MasterInstanceType:

Default: `m4.xlarge`

Type: String

AllowedValues:

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

AutoRegisterELB:

Default: `"yes"`

AllowedValues:

- `"yes"`
- `"no"`

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

Type: String

RegisterNlbTargetsLambdaArn:

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

Type: String

ExternalApiTargetGroupArn:

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

Type: String

InternalApiTargetGroupArn:

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

Type: String

InternalServiceTargetGroupArn:

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

Type: String

```
Metadata:
AWS::CloudFormation::Interface:
  ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
        - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
        - MasterInstanceType
        - RhcosAmi
        - IgnitionLocation
        - CertificateAuthorities
        - MasterSecurityGroupId
        - MasterInstanceProfileName
    - Label:
        default: "Network Configuration"
      Parameters:
        - VpcId
        - AllowedBootstrapSshCidr
        - Master0Subnet
        - Master1Subnet
        - Master2Subnet
    - Label:
        default: "DNS"
      Parameters:
        - AutoRegisterDNS
        - PrivateHostedZoneName
        - PrivateHostedZoneId
    - Label:
        default: "Load Balancer Automation"
      Parameters:
        - AutoRegisterELB
        - RegisterNlbTargetsLambdaArn
        - ExternalApiTargetGroupArn
        - InternalApiTargetGroupArn
        - InternalServiceTargetGroupArn
  ParameterLabels:
    InfrastructureName:
      default: "Infrastructure Name"
    VpcId:
      default: "VPC ID"
    Master0Subnet:
      default: "Master-0 Subnet"
    Master1Subnet:
      default: "Master-1 Subnet"
    Master2Subnet:
      default: "Master-2 Subnet"
    MasterInstanceType:
      default: "Master Instance Type"
    MasterInstanceProfileName:
      default: "Master Instance Profile Name"
    RhcosAmi:
      default: "Red Hat Enterprise Linux CoreOS AMI ID"
    BootstrapIgnitionLocation:
```

```

    default: "Master Ignition Source"
CertificateAuthorities:
    default: "Ignition CA String"
MasterSecurityGroupId:
    default: "Master Security Group ID"
AutoRegisterDNS:
    default: "Use Provided DNS Automation"
AutoRegisterELB:
    default: "Use Provided ELB Automation"
PrivateHostedZoneName:
    default: "Private Hosted Zone Name"
PrivateHostedZoneId:
    default: "Private Hosted Zone ID"

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

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

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

```

RegisterMaster0InternalApiTarget:

Condition: DoRegistration

Type: Custom::NLBRegister

Properties:

ServiceToken: !Ref RegisterNlbTargetsLambdaArn

TargetArn: !Ref InternalApiTargetGroupArn

TargetIp: !GetAtt Master0.PrivateIp

RegisterMaster0InternalServiceTarget:

Condition: DoRegistration

Type: Custom::NLBRegister

Properties:

ServiceToken: !Ref RegisterNlbTargetsLambdaArn

TargetArn: !Ref InternalServiceTargetGroupArn

TargetIp: !GetAtt Master0.PrivateIp

Master1:

Type: AWS::EC2::Instance

Properties:

ImageId: !Ref RhcosAmi

BlockDeviceMappings:

- DeviceName: /dev/xvda

Ebs:

VolumeSize: "120"

VolumeType: "gp2"

IamInstanceProfile: !Ref MasterInstanceProfileName

InstanceType: !Ref MasterInstanceType

NetworkInterfaces:

- AssociatePublicIpAddress: "false"

DeviceIndex: "0"

GroupSet:

- !Ref "MasterSecurityGroupId"

SubnetId: !Ref "Master1Subnet"

UserData:

Fn::Base64: !Sub

```
- '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}],"security":{"tls":{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]},"timeouts":{"version":"2.2.0"},"networkd":{"passwd":{},"storage":{},"systemd":{}}'
- {
```

SOURCE: !Ref IgnitionLocation,

CA_BUNDLE: !Ref CertificateAuthorities,

}

Tags:

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

Value: "shared"

RegisterMaster1:

Condition: DoRegistration

Type: Custom::NLBRegister

Properties:

ServiceToken: !Ref RegisterNlbTargetsLambdaArn

TargetArn: !Ref ExternalApiTargetGroupArn

TargetIp: !GetAtt Master1.PrivateIp

RegisterMaster1InternalApiTarget:

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

RegisterMaster1InternalServiceTarget:

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

Master2:

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

RegisterMaster2:

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

RegisterMaster2InternalApiTarget:

Condition: DoRegistration
 Type: Custom::NLBRegister

Properties:

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

RegisterMaster2InternalServiceTarget:

Condition: DoRegistration

Type: Custom::NLBRegister

Properties:

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

EtcdSrvRecords:

Condition: DoDns

Type: AWS::Route53::RecordSet

Properties:

HostedZoneId: !Ref PrivateHostedZoneId

Name: !Join [".", ["_etcd-server-ssl._tcp", !Ref PrivateHostedZoneName]]

ResourceRecords:

- !Join [
 " ",
 ["0 10 2380", !Join [".", ["etcd-0", !Ref PrivateHostedZoneName]]],
]
 - !Join [
 " ",
 ["0 10 2380", !Join [".", ["etcd-1", !Ref PrivateHostedZoneName]]],
]
 - !Join [
 " ",
 ["0 10 2380", !Join [".", ["etcd-2", !Ref PrivateHostedZoneName]]],
]

TTL: 60

Type: SRV

Etcd0Record:

Condition: DoDns

Type: AWS::Route53::RecordSet

Properties:

HostedZoneId: !Ref PrivateHostedZoneId

Name: !Join [".", ["etcd-0", !Ref PrivateHostedZoneName]]

ResourceRecords:

- !GetAtt Master0.PrivateIp

TTL: 60

Type: A

Etcd1Record:

Condition: DoDns

Type: AWS::Route53::RecordSet

Properties:

HostedZoneId: !Ref PrivateHostedZoneId

Name: !Join [".", ["etcd-1", !Ref PrivateHostedZoneName]]

ResourceRecords:

- !GetAtt Master1.PrivateIp

TTL: 60

Type: A

```

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

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

```

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

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

Prerequisites

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

Procedure

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

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

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

- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

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

1.7.14.1. Creating the worker nodes in AWS

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



IMPORTANT

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



NOTE

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

Prerequisites

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

Procedure

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

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "Subnet", 5
```

```

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

```

- 1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- 2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
- 3 Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the worker nodes.
- 4 Specify an **AWS::EC2::Image::Id** value.
- 5 A subnet, preferably private, to launch the worker nodes on.
- 6 Specify a subnet from the **PrivateSubnets** value from the output of the CloudFormation template for DNS and load balancing.
- 7 The worker security group ID to associate with worker nodes.
- 8 Specify the **WorkerSecurityGroupID** value from the output of the CloudFormation template for the security group and roles.
- 9 The location to fetch bootstrap Ignition config file from.
- 10 Specify the generated Ignition config location, https://api-int.<cluster_name>.<domain_name>:22623/config/worker.
- 11 Base64 encoded certificate authority string to use.
- 12 Specify the value from the **worker.ign** file that is in the installation directory. This value is the long string with the format **data:text/plain;charset=utf-8;base64,ABC...xYz==**.

- 13 The IAM profile to associate with worker nodes.
- 14 Specify the **WorkerInstanceProfile** parameter value from the output of the CloudFormation template for the security group and roles.
- 15 The type of AWS instance to use for the control plane machines.
- 16 Allowed values:
 - **m4.large**
 - **m4.xlarge**
 - **m4.2xlarge**
 - **m4.4xlarge**
 - **m4.8xlarge**
 - **m4.10xlarge**
 - **m4.16xlarge**
 - **c4.large**
 - **c4.xlarge**
 - **c4.2xlarge**
 - **c4.4xlarge**
 - **c4.8xlarge**
 - **r4.large**
 - **r4.xlarge**
 - **r4.2xlarge**
 - **r4.4xlarge**
 - **r4.8xlarge**
 - **r4.16xlarge**



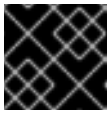
IMPORTANT

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

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

4. Create a worker stack.

a. Launch the template:

**IMPORTANT**

You must enter the command on a single line.

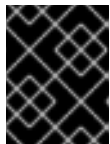
```
$ aws cloudformation create-stack --stack-name <name> 1
  --template-body file://<template>.yaml \ 2
  --parameters file://<parameters>.json 3
```

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

b. Confirm that the template components exist:

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

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

**IMPORTANT**

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

1.7.14.1.1. CloudFormation template for worker machines

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

Example 1.26. CloudFormation template for worker machines

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 worker instance)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
```

RhcosAmi:

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

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

Subnet:

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

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

WorkerSecurityGroupId:

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

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

IgnitionLocation:

Default: [https://api-int.\\$CLUSTER_NAME.\\$DOMAIN:22623/config/worker](https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/worker)

Description: Ignition config file location.

Type: String

CertificateAuthorities:

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

Description: Base64 encoded certificate authority string to use.

Type: String

WorkerInstanceProfileName:

Description: IAM profile to associate with master nodes.

Type: String

WorkerInstanceType:

Default: m4.large

Type: String

AllowedValues:

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

Metadata:**AWS::CloudFormation::Interface:**

ParameterGroups:

- Label:

default: "Cluster Information"

Parameters:

- InfrastructureName

- Label:

default: "Host Information"

Parameters:

- WorkerInstanceType

- RhcosAmi

- IgnitionLocation


```

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

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

```

```

Outputs:
PrivateIP:
  Description: The compute node private IP address.
  Value: !GetAtt Worker0.PrivateIp

```

1.7.15. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order 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 4.5. Download and install the new version of **oc**.

1.7.15.1. Installing the CLI on Linux

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

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command-line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

1.7.15.2. Installing the CLI on Windows

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

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.

3. In the **Command-line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
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
```

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

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

1.7.15.3. Installing the CLI on macOS

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

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command-line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
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
```

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

```
$ oc <command>
```

1.7.16. Logging in to the cluster

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

Prerequisites

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

Procedure

1. Export the **kubeadmin** credentials:

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

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

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

```
$ oc whoami
```

Example output

```
system:admin
```

1.7.17. 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   v1.18.3
master-1  Ready    master   63m   v1.18.3
master-2  Ready    master   64m   v1.18.3
worker-0  NotReady worker   76s   v1.18.3
worker-1  NotReady worker   70s   v1.18.3
```

The output lists all of the machines that you created.

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 you approve the initial CSRs, the subsequent node client CSRs are automatically approved by the cluster **kube-controller-manager**. You must implement a method of automatically approving the kubelet serving certificate requests.

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

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

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

- 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
```

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
```

- 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   v1.20.0
master-1  Ready    master   73m   v1.20.0
master-2  Ready    master   74m   v1.20.0
worker-0  Ready    worker   11m   v1.20.0
worker-1  Ready    worker   11m   v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

1.7.18. Initial Operator configuration

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

Prerequisites

- Your control plane has initialized.

Procedure

- Watch the cluster components come online:

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

Example output

| NAME | VERSION | AVAILABLE | PROGRESSING | DEGRADED |
|--------------------|---------|-----------|-------------|----------|
| authentication | 4.5.4 | True | False | 69s |
| cloud-credential | 4.5.4 | True | False | 12m |
| cluster-autoscaler | 4.5.4 | True | False | 11m |

| | | | | | |
|------------------------------------|-------|------|-------|-------|--------|
| console | 4.5.4 | True | False | False | 46s |
| dns | 4.5.4 | True | False | False | 11m |
| image-registry | 4.5.4 | True | False | False | 5m26s |
| ingress | 4.5.4 | True | False | False | 5m36s |
| kube-apiserver | 4.5.4 | True | False | False | 8m53s |
| kube-controller-manager | 4.5.4 | True | False | False | 7m24s |
| kube-scheduler | 4.5.4 | True | False | False | 12m |
| machine-api | 4.5.4 | True | False | False | 12m |
| machine-config | 4.5.4 | True | False | False | 7m36s |
| marketplace | 4.5.4 | True | False | False | 7m54m |
| monitoring | 4.5.4 | True | False | False | 7h54s |
| network | 4.5.4 | True | False | False | 5m9s |
| node-tuning | 4.5.4 | True | False | False | 11m |
| openshift-apiserver | 4.5.4 | True | False | False | 11m |
| openshift-controller-manager | 4.5.4 | True | False | False | 5m943s |
| openshift-samples | 4.5.4 | True | False | False | 3m55s |
| operator-lifecycle-manager | 4.5.4 | True | False | False | 11m |
| operator-lifecycle-manager-catalog | 4.5.4 | True | False | False | 11m |
| service-ca | 4.5.4 | True | False | False | 11m |
| service-catalog-apiserver | 4.5.4 | True | False | False | 5m26s |
| service-catalog-controller-manager | 4.5.4 | True | False | False | 5m25s |
| storage | 4.5.4 | True | False | False | 5m30s |

2. Configure the Operators that are not available.

1.7.18.1. Image registry storage configuration

Amazon Web Services provides default storage, which means the Image Registry Operator is available after installation. However, if the Registry Operator cannot create an S3 bucket and automatically configure storage, you must manually configure registry storage.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

You can configure registry storage for user-provisioned infrastructure in AWS to deploy OpenShift Container Platform to hidden regions. See [Configuring the registry for AWS user-provisioned infrastructure](#) for more information.

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

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

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

Prerequisites

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

- **REGISTRY_STORAGE_S3_ACCESSKEY**
- **REGISTRY_STORAGE_S3_SECRETKEY**

Procedure

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

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

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

Example configuration

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



WARNING

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

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

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

Procedure

1. To set the image registry storage to an empty directory:

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



WARNING

Configure this option for only non-production clusters.

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

```
-
```



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

Wait a few minutes and run the command again.

2. Ensure that your registry is set to managed to enable building and pushing of images.

- Run:

```
$ oc edit configs.imageregistry/cluster
```

Then, change the line

```
managementState: Removed
```

to

```
managementState: Managed
```

1.7.19. Deleting the bootstrap resources

After you complete the initial Operator configuration for the cluster, remove the bootstrap resources from Amazon Web Services (AWS).

Prerequisites

- You completed the initial Operator configuration for your cluster.

Procedure

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

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

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

1.7.20. Creating the Ingress DNS Records

If you removed the DNS Zone configuration, manually create DNS records that point to the Ingress load balancer. You can create either a wildcard record or specific records. While the following procedure uses A records, you can use other record types that you require, such as CNAME or alias.

Prerequisites

- You deployed an OpenShift Container Platform cluster on Amazon Web Services (AWS) that uses infrastructure that you provisioned.
- Install the OpenShift CLI (**oc**).
- Install the **jq** package.
- Download the AWS CLI and install it on your computer. See [Install the AWS CLI Using the Bundled Installer \(Linux, macOS, or Unix\)](#).

Procedure

1. Determine the routes to create.

- To create a wildcard record, use ***.apps.<cluster_name>.<domain_name>**, where **<cluster_name>** is your cluster name, and **<domain_name>** is the Route 53 base domain for your OpenShift Container Platform cluster.
- To create specific records, you must create a record for each route that your cluster uses, as shown in the output of the following command:

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

Example output

```
oauth-openshift.apps.<cluster_name>.<domain_name>
console-openshift-console.apps.<cluster_name>.<domain_name>
downloads-openshift-console.apps.<cluster_name>.<domain_name>
alertmanager-main-openshift-monitoring.apps.<cluster_name>.<domain_name>
grafana-openshift-monitoring.apps.<cluster_name>.<domain_name>
prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<domain_name>
```

2. Retrieve the Ingress Operator load balancer status and note the value of the external IP address that it uses, which is shown in the **EXTERNAL-IP** column:

```
$ oc -n openshift-ingress get service router-default
```

Example output

| NAME | TYPE | CLUSTER-IP | EXTERNAL-IP | PORT(S) |
|----------------|--------------|---------------|--------------------------------------|----------------------------|
| router-default | LoadBalancer | 172.30.62.215 | ab3...28.us-east-2.elb.amazonaws.com | 80:31499/TCP,443:30693/TCP |
| | | | | 5m |

3. Locate the hosted zone ID for the load balancer:

```
$ aws elb describe-load-balancers | jq -r '.LoadBalancerDescriptions[] | select(.DNSName ==
"<external_ip>").CanonicalHostedZoneNameID' 1
```

- 1** For **<external_ip>**, specify the value of the external IP address of the Ingress Operator load balancer that you obtained.

Example output

```
Z3AADJGX6KTTL2
```

The output of this command is the load balancer hosted zone ID.

4. Obtain the public hosted zone ID for your cluster's domain:

```
$ aws route53 list-hosted-zones-by-name \
```

```
--dns-name "<domain_name>" \ ❶
--query 'HostedZones[? Config.PrivateZone != `true` && Name ==
`<domain_name>.`].Id' ❷
--output text
```

- ❶ ❷ For **<domain_name>**, specify the Route 53 base domain for your OpenShift Container Platform cluster.

Example output

```
/hostedzone/Z3URY6TWQ91KVV
```

The public hosted zone ID for your domain is shown in the command output. In this example, it is **Z3URY6TWQ91KVV**.

5. Add the alias records to your private zone:

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<private_hosted_zone_id>" --
change-batch '{ ❶
> "Changes": [
> {
>   "Action": "CREATE",
>   "ResourceRecordSet": {
>     "Name": "\\052.apps.<cluster_domain>", ❷
>     "Type": "A",
>     "AliasTarget":{
>       "HostedZoneId": "<hosted_zone_id>", ❸
>       "DNSName": "<external_ip>.", ❹
>       "EvaluateTargetHealth": false
>     }
>   }
> }
> ]
> }'
```

- ❶ For **<private_hosted_zone_id>**, specify the value from the output of the CloudFormation template for DNS and load balancing.
- ❷ For **<cluster_domain>**, specify the domain or subdomain that you use with your OpenShift Container Platform cluster.
- ❸ For **<hosted_zone_id>**, specify the public hosted zone ID for the load balancer that you obtained.
- ❹ For **<external_ip>**, specify the value of the external IP address of the Ingress Operator load balancer. Ensure that you include the trailing period (.) in this parameter value.

6. Add the records to your public zone:

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<public_hosted_zone_id>" --
change-batch '{ ❶
> "Changes": [
> {
```

```

>   "Action": "CREATE",
>   "ResourceRecordSet": {
>     "Name": "\\052.apps.<cluster_domain>", 2
>     "Type": "A",
>     "AliasTarget": {
>       "HostedZoneId": "<hosted_zone_id>", 3
>       "DNSName": "<external_ip>.", 4
>       "EvaluateTargetHealth": false
>     }
>   }
> }
> ]
> }'

```

- 1 For **<public_hosted_zone_id>**, specify the public hosted zone for your domain.
- 2 For **<cluster_domain>**, specify the domain or subdomain that you use with your OpenShift Container Platform cluster.
- 3 For **<hosted_zone_id>**, specify the public hosted zone ID for the load balancer that you obtained.
- 4 For **<external_ip>**, specify the value of the external IP address of the Ingress Operator load balancer. Ensure that you include the trailing period (.) in this parameter value.

1.7.21. Completing an AWS installation on user-provisioned infrastructure

After you start the OpenShift Container Platform installation on Amazon Web Service (AWS) user-provisioned infrastructure, monitor the deployment to completion.

Prerequisites

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

Procedure

- Complete the cluster installation:

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

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

Example output

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



IMPORTANT

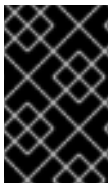
The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically 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.

1.7.22. Next steps

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

1.8. INSTALLING A CLUSTER ON AWS IN A RESTRICTED NETWORK WITH USER-PROVISIONED INFRASTRUCTURE

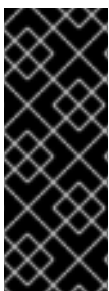
In OpenShift Container Platform version 4.5, you can install a cluster on Amazon Web Services (AWS) using infrastructure that you provide and an internal mirror of the installation release content.



IMPORTANT

While you can install an OpenShift Container Platform cluster by using mirrored installation release content, your cluster still requires Internet access to use the AWS APIs.

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

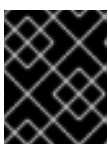


IMPORTANT

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 CloudFormation 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.

1.8.1. Prerequisites

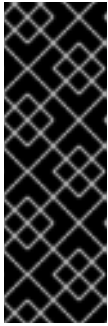
- You [created a mirror registry on your mirror host](#) and obtained the **imageContentSources** data for your version of OpenShift Container Platform.



IMPORTANT

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

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



IMPORTANT

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

- Download the AWS CLI and install it on your computer. See [Install the AWS CLI Using the Bundled Installer \(Linux, macOS, or Unix\)](#) in the AWS documentation.
- If you use a firewall and plan to use telemetry, you must [configure the firewall to allow the sites](#) that your cluster requires access to.



NOTE

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

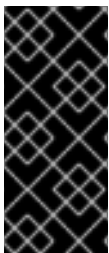
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

1.8.2. About installations in restricted networks

In OpenShift Container Platform 4.5, 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 IAM service, require Internet access, so you might still require Internet access. Depending on your network, you might require less Internet access for an installation on bare metal hardware or on VMware vSphere.

To complete a restricted network installation, you must create a registry that mirrors the contents of the OpenShift Container Platform registry and contains the installation media. You can create this 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.

1.8.2.1. 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.

1.8.3. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to obtain the images that are necessary to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

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

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page 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 content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

1.8.4. Required AWS infrastructure components

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

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

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

1.8.4.1. Cluster machines

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

- A bootstrap machine. This machine is required during installation, but you can remove it after your cluster deploys.
- Three control plane machines. The control plane machines are not governed by a machine set.
- Compute machines. You must create at least two compute machines, which are also known as worker machines, during installation. These machines are not governed by a machine set.

You can use the following instance types for the cluster machines with the provided Cloud Formation templates.



IMPORTANT

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

Table 1.16. Instance types for machines

| Instance type | Bootstrap | Control plane | Compute |
|--------------------------------------|-----------|---------------|---------|
| i3.large | x | | |
| m4.large or m5.large | | | x |
| m4.xlarge or m5.xlarge | | x | x |
| m4.2xlarge | | x | x |
| m4.4xlarge | | x | x |
| m4.8xlarge | | x | x |
| m4.10xlarge | | x | x |
| m4.16xlarge | | x | x |
| c4.large | | | x |
| c4.xlarge | | | x |
| c4.2xlarge | | x | x |
| c4.4xlarge | | x | x |
| c4.8xlarge | | x | x |
| r4.large | | | x |

| Instance type | Bootstrap | Control plane | Compute |
|--------------------|-----------|---------------|---------|
| r4.xlarge | | x | x |
| r4.2xlarge | | x | x |
| r4.4xlarge | | x | x |
| r4.8xlarge | | x | x |
| r4.16xlarge | | x | x |

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

1.8.4.2. 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.

1.8.4.3. Other infrastructure components

- A VPC
- DNS entries
- Load balancers (classic or network) and listeners
- A public and a private Route 53 zone
- Security groups
- IAM roles
- S3 buckets

If you are working in a disconnected environment, you are unable to reach the public IP addresses for EC2 and ELB endpoints. To resolve this, you must create a VPC endpoint and attach it to the subnet that the clusters are using. The endpoints should be named as follows:

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

Required VPC components

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

| Component | AWS type | Description | |
|------------------------|---|--|----------------------------|
| VPC | <ul style="list-style-type: none">• AWS::EC2::VPC• AWS::EC2::VPCEndpoint | You must provide a public VPC for the cluster to use. The VPC uses an endpoint that references the route tables for each subnet to improve communication with the registry that is hosted in S3. | |
| Public subnets | <ul style="list-style-type: none">• AWS::EC2::Subnet• AWS::EC2::SubnetNetworkAclAssociation | Your VPC must have public subnets for between 1 and 3 availability zones and associate them with appropriate Ingress rules. | |
| Internet gateway | <ul style="list-style-type: none">• AWS::EC2::InternetGateway• AWS::EC2::VPCGatewayAttachment• AWS::EC2::RouteTable• AWS::EC2::Route• AWS::EC2::SubnetRouteTableAssociation• AWS::EC2::NatGateway• AWS::EC2::EIP | You must have a public Internet gateway, with public routes, attached to the VPC. In the provided templates, each public subnet has a NAT gateway with an EIP address. These NAT gateways allow cluster resources, like private subnet instances, to reach the Internet and are not required for some restricted network or proxy scenarios. | |
| Network access control | <ul style="list-style-type: none">• AWS::EC2::NetworkAcl• AWS::EC2::NetworkAclEntry | You must allow the VPC to access the following ports: | |
| | | Port | Reason |
| | | 80 | Inbound HTTP traffic |
| | | 443 | Inbound HTTPS traffic |
| | | 22 | Inbound SSH traffic |
| | | 1024 - 65535 | Inbound ephemeral traffic |
| | | 0 - 65535 | Outbound ephemeral traffic |

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

Required DNS and load balancing components

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

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

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

| Component | AWS type | Description |
|----------------------------|--|--|
| External target group | AWS::ElasticLoadBalancingV2::TargetGroup | The target group for the external load balancer. |
| Private load balancer | AWS::ElasticLoadBalancingV2::LoadBalancer | The load balancer for your private subnets. |
| Internal API server record | AWS::Route53::RecordSetGroup | Alias records for the internal API server. |
| Internal listener | AWS::ElasticLoadBalancingV2::Listener | A listener on port 22623 for the internal load balancer. |
| Internal target group | AWS::ElasticLoadBalancingV2::TargetGroup | The target group for the internal load balancer. |
| Internal listener | AWS::ElasticLoadBalancingV2::Listener | A listener on port 6443 for the internal load balancer. |
| Internal target group | AWS::ElasticLoadBalancingV2::TargetGroup | The target group for the internal load balancer. |

Security groups

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

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

| Group | Type | IP Protocol | Port range |
|-------------------------------|--------------------------------|-------------|--------------|
| | | tcp | 22623 |
| WorkerSecurityGroup | AWS::EC2::SecurityGroup | icmp | 0 |
| | | tcp | 22 |
| BootstrapSecurityGroup | AWS::EC2::SecurityGroup | tcp | 22 |
| | | tcp | 19531 |

Control plane Ingress

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

| Ingress group | Description | IP protocol | Port range |
|---|---|-------------|----------------------|
| MasterIngressEtcd | etcd | tcp | 2379- 2380 |
| MasterIngressVxlan | Vxlan packets | udp | 4789 |
| MasterIngressWorkerVxlan | Vxlan packets | udp | 4789 |
| MasterIngressInternal | Internal cluster communication and Kubernetes proxy metrics | tcp | 9000 - 9999 |
| MasterIngressWorkerInternal | Internal cluster communication | tcp | 9000 - 9999 |
| MasterIngressKube | Kubernetes kubelet, scheduler and controller manager | tcp | 10250 - 10259 |
| MasterIngressWorkerKube | Kubernetes kubelet, scheduler and controller manager | tcp | 10250 - 10259 |
| MasterIngressIngressServices | Kubernetes Ingress services | tcp | 30000 - 32767 |
| MasterIngressWorkerIngressServices | Kubernetes Ingress services | tcp | 30000 - 32767 |

Worker Ingress

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

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

Roles and instance profiles

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

| Role | Effect | Action | Resource |
|--------|--------------|-------------------------------|----------|
| Master | Allow | ec2:* | * |
| | Allow | elasticloadbalancing:* | * |
| | Allow | iam:PassRole | * |

| Role | Effect | Action | Resource |
|-----------|--------------|-------------------------|----------|
| | Allow | s3:GetObject | * |
| Worker | Allow | ec2:Describe* | * |
| Bootstrap | Allow | ec2:Describe* | * |
| | Allow | ec2:AttachVolume | * |
| | Allow | ec2:DetachVolume | * |

1.8.4.4. Required AWS permissions

When you attach the **AdministratorAccess** policy to the IAM user that you create in Amazon Web Services (AWS), you grant that user all of the required permissions. To deploy all components of an OpenShift Container Platform cluster, the IAM user requires the following permissions:

Example 1.27. Required EC2 permissions for installation

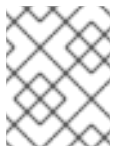
- **tag:TagResources**
- **tag:UntagResources**
- **ec2:AllocateAddress**
- **ec2:AssociateAddress**
- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2:CreateNetworkInterface**
- **ec2:AttachNetworkInterface**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2>DeleteSecurityGroup**
- **ec2>DeleteSnapshot**
- **ec2>DeleteTags**
- **ec2:DeregisterImage**
- **ec2:DescribeAccountAttributes**

- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInternetGateways**
- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**
- **ec2:DescribeNetworkAcls**
- **ec2:DescribeNetworkInterfaces**
- **ec2:DescribePrefixLists**
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcAttribute**
- **ec2:DescribeVpcClassicLink**
- **ec2:DescribeVpcClassicLinkDnsSupport**
- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**
- **ec2:GetEbsDefaultKmsKeyId**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyNetworkInterfaceAttribute**
- **ec2:ReleaseAddress**
- **ec2:RevokeSecurityGroupEgress**

- **ec2:RevokeSecurityGroupIngress**
- **ec2:RunInstances**
- **ec2:TerminateInstances**

Example 1.28. Required permissions for creating network resources during installation

- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**
- **ec2:CreateDhcpOptions**
- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**
- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**



NOTE

If you use an existing VPC, your account does not require these permissions for creating network resources.

Example 1.29. Required Elastic Load Balancing permissions for installation

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

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

Example 1.30. Required IAM permissions for installation

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

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



NOTE

If you have not created an elastic load balancer (ELB) in your AWS account, the IAM user also requires the **iam:CreateServiceLinkedRole** permission.

Example 1.31. Required Route 53 permissions for installation

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

Example 1.32. Required S3 permissions for installation

- **s3:CreateBucket**
- **s3>DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketAcl**
- **s3:GetBucketCors**

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

Example 1.33. S3 permissions that cluster Operators require

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

Example 1.34. Required permissions to delete base cluster resources

- **autoscaling:DescribeAutoScalingGroups**
- **ec2:DeleteNetworkInterface**
- **ec2:DeleteVolume**

- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**
- **s3:DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

Example 1.35. Required permissions to delete network resources

- **ec2:DeleteDhcpOptions**
- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteRoute**
- **ec2:DeleteRouteTable**
- **ec2:DeleteSubnet**
- **ec2:DeleteVpc**
- **ec2:DeleteVpcEndpoints**
- **ec2:DetachInternetGateway**
- **ec2:DisassociateRouteTable**
- **ec2:ReplaceRouteTableAssociation**



NOTE

If you use an existing VPC, your account does not require these permissions to delete network resources.

Example 1.36. Additional IAM and S3 permissions that are required to create manifests

- **iam:CreateAccessKey**
- **iam:CreateUser**

- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:DeleteUserPolicy**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **s3:PutBucketPublicAccessBlock**
- **s3:GetBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**
- **s3:HeadBucket**
- **s3:ListBucketMultipartUploads**
- **s3:AbortMultipartUpload**

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

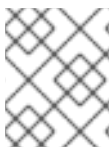
If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

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



NOTE

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

Procedure

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

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

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

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

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

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

Example output

```
Agent pid 31874
```

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

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

Example output

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

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

Next steps

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

1.8.6. Creating the installation files for AWS

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

1.8.6.1. Creating the installation configuration file

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

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster. For a restricted network installation, these files are on your mirror host.

Procedure

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

- a. Run the following command:

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

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



IMPORTANT

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

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



NOTE

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

- ii. Select **aws** as the platform to target.
 - iii. If you do not have an AWS profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.
 - iv. Select the AWS region to deploy the cluster to.
 - v. Select the base domain for the Route 53 service that you configured for your cluster.
 - vi. Enter a descriptive name for your cluster.
 - vii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Edit the **install-config.yaml** file to set the number of compute replicas, which are also known as worker replicas, to **0**, as shown in the following **compute** stanza:

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


3. Edit the **install-config.yaml** file to provide the additional information that is required for an installation in a restricted network.

- a. Update the **pullSecret** value to contain the authentication information for your registry:

```
pullSecret: '{"auths":{"<local_registry>":{"auth":"<credentials>","email":  
"you@example.com"}}}'
```

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.

- b. Add the **additionalTrustBundle** parameter and value. The value must be the contents of the certificate file that you used for your mirror registry, which can be an exiting, trusted certificate authority or the self-signed certificate that you generated for the mirror registry.

[illegible]

- c. Add the image content resources:

```
imageContentSources:
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: registry.svc.ci.openshift.org/ocp/release
```

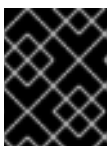
Use the **imageContentSources** section from the output of the command to mirror the repository or the values that you used when you mirrored the content from the media that you brought into your restricted network.

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

publish: Internal

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

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



IMPORTANT

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

1.8.6.2. Configuring the cluster-wide proxy during installation

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

Prerequisites

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



NOTE

The **Proxy** object **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: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
  additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
  ...
```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass 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

Proxy object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.



NOTE

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

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

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

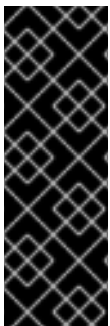


NOTE

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

1.8.6.3. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, 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.

Prerequisites

- Obtain the OpenShift Container Platform installation program. For a restricted network installation, these files are on your mirror host.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

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

Example output

INFO Consuming Install Config from target directory
 WARNING Making control-plane schedulable by setting MastersSchedulable to true for Scheduler cluster settings

- 1 For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

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

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

```
$ rm -f <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 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.

4. Modify the **<installation_directory>/manifests/cluster-scheduler-02-config.yaml** Kubernetes manifest file to prevent pods from being scheduled on the control plane machines:
 - a. Open the **<installation_directory>/manifests/cluster-scheduler-02-config.yaml** file.
 - b. Locate the **mastersSchedulable** parameter and set its value to **False**.
 - c. Save and exit the file.
5. Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the **privateZone** and **publicZone** sections from the **<installation_directory>/manifests/cluster-dns-02-config.yaml** 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.

6. Obtain the Ignition config files:

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

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

The following files are generated in the directory:

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

1.8.7. Extracting the infrastructure name

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

Prerequisites

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

Procedure

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

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

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.

1.8.8. Creating a VPC in AWS

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



NOTE

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

Prerequisites

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

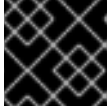
Procedure

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

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

- ¹ The CIDR block for the VPC.
- ² Specify a CIDR block in the format **x.x.x.x/16-24**.
- ³ The number of availability zones to deploy the VPC in.
- ⁴ Specify an integer between **1** and **3**.
- ⁵ The size of each subnet in each availability zone.
- ⁶ Specify an integer between **5** and **13**, where **5** is **/27** and **13** is **/19**.

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



IMPORTANT

You must enter the command on a single line.

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

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

4. Confirm that the template components exist:

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

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

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

1.8.8.1. CloudFormation template for the VPC

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

Example 1.37. CloudFormation template for the VPC

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice VPC with 1-3 AZs

Parameters:
  VpcCidr:
    AllowedPattern: ^((([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])|(V(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
```

AvailabilityZoneCount:

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

MinValue: 1

MaxValue: 3

Default: 1

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

Type: Number

SubnetBits:

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

MinValue: 5

MaxValue: 13

Default: 12

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

Type: Number

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:

default: "Network Configuration"

Parameters:

- VpcCidr

- SubnetBits

- Label:

default: "Availability Zones"

Parameters:

- AvailabilityZoneCount

ParameterLabels:

AvailabilityZoneCount:

default: "Availability Zone Count"

VpcCidr:

default: "VPC CIDR"

SubnetBits:

default: "Bits Per Subnet"

Conditions:

DoAz3: !Equals [3, !Ref AvailabilityZoneCount]

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

Resources:

VPC:

Type: "AWS::EC2::VPC"

Properties:

EnableDnsSupport: "true"

EnableDnsHostnames: "true"

CidrBlock: !Ref VpcCidr

PublicSubnet:

Type: "AWS::EC2::Subnet"

Properties:

VpcId: !Ref VPC

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

AvailabilityZone: !Select

- 0

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

PublicSubnet2:


```

Type: "AWS::EC2::Subnet"
Condition: DoAz2
Properties:
  VpcId: !Ref VPC
  CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
  AvailabilityZone: !Select
    - 1
    - Fn::GetAZs: !Ref "AWS::Region"
PublicSubnet3:
Type: "AWS::EC2::Subnet"
Condition: DoAz3
Properties:
  VpcId: !Ref VPC
  CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
  AvailabilityZone: !Select
    - 2
    - Fn::GetAZs: !Ref "AWS::Region"
InternetGateway:
Type: "AWS::EC2::InternetGateway"
GatewayToInternet:
Type: "AWS::EC2::VPCGatewayAttachment"
Properties:
  VpcId: !Ref VPC
  InternetGatewayId: !Ref InternetGateway
PublicRouteTable:
Type: "AWS::EC2::RouteTable"
Properties:
  VpcId: !Ref VPC
PublicRoute:
Type: "AWS::EC2::Route"
DependsOn: GatewayToInternet
Properties:
  RouteTableId: !Ref PublicRouteTable
  DestinationCidrBlock: 0.0.0.0/0
  GatewayId: !Ref InternetGateway
PublicSubnetRouteTableAssociation:
Type: "AWS::EC2::SubnetRouteTableAssociation"
Properties:
  SubnetId: !Ref PublicSubnet
  RouteTableId: !Ref PublicRouteTable
PublicSubnetRouteTableAssociation2:
Type: "AWS::EC2::SubnetRouteTableAssociation"
Condition: DoAz2
Properties:
  SubnetId: !Ref PublicSubnet2
  RouteTableId: !Ref PublicRouteTable
PublicSubnetRouteTableAssociation3:
Condition: DoAz3
Type: "AWS::EC2::SubnetRouteTableAssociation"
Properties:
  SubnetId: !Ref PublicSubnet3
  RouteTableId: !Ref PublicRouteTable
PrivateSubnet:
Type: "AWS::EC2::Subnet"
Properties:
  VpcId: !Ref VPC

```

```

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

```

```

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

```

```

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

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

```

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

1.8.9. Creating networking and load balancing components in AWS

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

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



NOTE

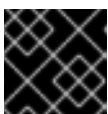
If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

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

Procedure

1. Obtain the Hosted Zone ID for the Route 53 zone that you specified in the **install-config.yaml** file for your cluster. You can obtain this ID from the AWS console or by running the following command:



IMPORTANT

You must enter the command on a single line.

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

- 1** For the **<route53_domain>**, specify the Route 53 base domain that you used when you generated the **install-config.yaml** file for the cluster.

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

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

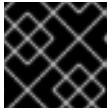
```

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

```

- 1 A short, representative cluster name to use for host names, etc.
- 2 Specify the cluster name that you used when you generated the **install-config.yaml** file for the cluster.
- 3 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- 4 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
- 5 The Route 53 public zone ID to register the targets with.
- 6 Specify the Route 53 public zone ID, which as a format similar to **Z21XYZABCZ2A4**. You can obtain this value from the AWS console.
- 7 The Route 53 zone to register the targets with.
- 8 Specify the Route 53 base domain that you used when you generated the **install-config.yaml** file for the cluster. Do not include the trailing period (.) that is displayed in the AWS console.
- 9 The public subnets that you created for your VPC.
- 10 Specify the **PublicSubnetIds** value from the output of the CloudFormation template for the VPC.
- 11 The private subnets that you created for your VPC.

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



IMPORTANT

You must enter the command on a single line.

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

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

5. Confirm that the template components exist:

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

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

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

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

1.8.9.1. CloudFormation template for the network and load balancers

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

Example 1.38. CloudFormation template for the network and load balancers

```

AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Network Elements (Route53 & LBs)

Parameters:
  ClusterName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-\_]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Cluster name must be alphanumeric, start with a letter, and have a
    maximum of 27 characters.
    Description: A short, representative cluster name to use for host names and other identifying
    names.
    Type: String
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-\_]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
    maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
    used by the cluster.
    Type: String
  HostedZoneId:
    Description: The Route53 public zone ID to register the targets with, such as
    Z21IXYZABCZ2A4.
    Type: String

```


HostedZoneName:

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

Type: String

Default: "example.com"

PublicSubnets:

Description: The internet-facing subnets.

Type: List<AWS::EC2::Subnet::Id>

PrivateSubnets:

Description: The internal subnets.

Type: List<AWS::EC2::Subnet::Id>

VpcId:

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

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

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:

default: "Cluster Information"

Parameters:

- ClusterName

- InfrastructureName

- Label:

default: "Network Configuration"

Parameters:

- VpcId

- PublicSubnets

- PrivateSubnets

- Label:

default: "DNS"

Parameters:

- HostedZoneName

- HostedZoneId

ParameterLabels:

ClusterName:

default: "Cluster Name"

InfrastructureName:

default: "Infrastructure Name"

VpcId:

default: "VPC ID"

PublicSubnets:

default: "Public Subnets"

PrivateSubnets:

default: "Private Subnets"

HostedZoneName:

default: "Public Hosted Zone Name"

HostedZoneId:

default: "Public Hosted Zone ID"

Resources:

ExtApiElb:

Type: AWS::ElasticLoadBalancingV2::LoadBalancer

Properties:

Name: !Join ["-", [!Ref InfrastructureName, "ext"]]

IpAddressType: ipv4

Subnets: !Ref PublicSubnets

Type: network

IntApiElb:

Type: AWS::ElasticLoadBalancingV2::LoadBalancer

Properties:

Name: !Join ["-", [!Ref InfrastructureName, "int"]]

Scheme: internal

IpAddressType: ipv4

Subnets: !Ref PrivateSubnets

Type: network

IntDns:

Type: "AWS::Route53::HostedZone"

Properties:

HostedZoneConfig:

Comment: "Managed by CloudFormation"

Name: !Join [".", [!Ref ClusterName, !Ref HostedZoneName]]

HostedZoneTags:

- Key: Name

Value: !Join ["-", [!Ref InfrastructureName, "int"]]

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

Value: "owned"

VPCs:

- VPCId: !Ref VpcId

VPCRegion: !Ref "AWS::Region"

ExternalApiServerRecord:

Type: AWS::Route53::RecordSetGroup

Properties:

Comment: Alias record for the API server

HostedZoneId: !Ref HostedZoneId

RecordSets:

- Name:

!Join [

".",

["api", !Ref ClusterName, !Join ["/", [!Ref HostedZoneName, "."]]],

]

Type: A

AliasTarget:

HostedZoneId: !GetAtt ExtApiElb.CanonicalHostedZoneID

DNSName: !GetAtt ExtApiElb.DNSName

InternalApiServerRecord:

Type: AWS::Route53::RecordSetGroup

Properties:

Comment: Alias record for the API server

HostedZoneId: !Ref IntDns

RecordSets:

- Name:

!Join [

".",

["api", !Ref ClusterName, !Join ["/", [!Ref HostedZoneName, "."]]],

]

Type: A

AliasTarget:

```

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

```

```

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

```

```

ExternalApiTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 10
    HealthCheckPath: "/readyz"
    HealthCheckPort: 6443
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
    Port: 6443
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
      - Key: deregistration_delay.timeout_seconds
        Value: 60

```

```

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

```

```

InternalApiTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup

```

Properties:

HealthCheckIntervalSeconds: 10
HealthCheckPath: `"/readyz"`
HealthCheckPort: 6443
HealthCheckProtocol: HTTPS
HealthyThresholdCount: 2
UnhealthyThresholdCount: 2
Port: 6443
Protocol: TCP
TargetType: ip
VpcId:
 Ref: VpcId
TargetGroupAttributes:
 - Key: deregistration_delay.timeout_seconds
 Value: 60

InternalServiceInternalListener:

Type: AWS::ElasticLoadBalancingV2::Listener
Properties:
 DefaultActions:
 - Type: forward
 TargetGroupArn:
 Ref: InternalServiceTargetGroup
 LoadBalancerArn:
 Ref: IntApiElb
 Port: 22623
 Protocol: TCP

InternalServiceTargetGroup:

Type: AWS::ElasticLoadBalancingV2::TargetGroup
Properties:
 HealthCheckIntervalSeconds: 10
 HealthCheckPath: `"/healthz"`
 HealthCheckPort: 22623
 HealthCheckProtocol: HTTPS
 HealthyThresholdCount: 2
 UnhealthyThresholdCount: 2
 Port: 22623
 Protocol: TCP
 TargetType: ip
 VpcId:
 Ref: VpcId
 TargetGroupAttributes:
 - Key: deregistration_delay.timeout_seconds
 Value: 60

RegisterTargetLambdalaRole:

Type: AWS::IAM::Role
Properties:
 RoleName: !Join ["-", [!Ref InfrastructureName, "nlb", "lambda", "role"]]
 AssumeRolePolicyDocument:
 Version: "2012-10-17"
 Statement:
 - Effect: "Allow"
 Principal:
 Service:

```

    - "lambda.amazonaws.com"
  Action:
    - "sts:AssumeRole"
  Path: "/"
  Policies:
    - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
    PolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: "Allow"
          Action:
            [
              "elasticloadbalancing:RegisterTargets",
              "elasticloadbalancing:DeregisterTargets",
            ]
          Resource: !Ref InternalApiTargetGroup
        - Effect: "Allow"
          Action:
            [
              "elasticloadbalancing:RegisterTargets",
              "elasticloadbalancing:DeregisterTargets",
            ]
          Resource: !Ref InternalServiceTargetGroup
        - Effect: "Allow"
          Action:
            [
              "elasticloadbalancing:RegisterTargets",
              "elasticloadbalancing:DeregisterTargets",
            ]
          Resource: !Ref ExternalApiTargetGroup

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

```

Timeout: 120

RegisterSubnetTagsLambdalamRole:

Type: AWS::IAM::Role

Properties:

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

AssumeRolePolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Principal:

Service:

- "lambda.amazonaws.com"

Action:

- "sts:AssumeRole"

Path: "/"

Policies:

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

PolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Action:

```
[
  "ec2:DeleteTags",
  "ec2:CreateTags"
]
```

Resource: "arn:aws:ec2:*:*:subnet/*"

- Effect: "Allow"

Action:

```
[
  "ec2:DescribeSubnets",
  "ec2:DescribeTags"
]
```

Resource: ""

RegisterSubnetTags:

Type: "AWS::Lambda::Function"

Properties:

Handler: "index.handler"

Role:

Fn::GetAtt:

- "RegisterSubnetTagsLambdalamRole"

- "Arn"

Code:

ZipFile: |

```
import json
```

```
import boto3
```

```
import cfnresponse
```

```
def handler(event, context):
```

```
    ec2_client = boto3.client('ec2')
```

```
    if event['RequestType'] == 'Delete':
```

```
        for subnet_id in event['ResourceProperties']['Subnets']:
```

```
            ec2_client.delete_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
```

```
event['ResourceProperties']['InfrastructureName']});
```

```
        elif event['RequestType'] == 'Create':
```

```

        for subnet_id in event['ResourceProperties']['Subnets']:
            ec2_client.create_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName'], 'Value': 'shared'}]);
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['InfrastructureName']+event['ResourceProperties']['Subnets'][0])
    Runtime: "python3.7"
    Timeout: 120

```

```

RegisterPublicSubnetTags:
  Type: Custom::SubnetRegister
  Properties:
    ServiceToken: !GetAtt RegisterSubnetTags.Arn
    InfrastructureName: !Ref InfrastructureName
    Subnets: !Ref PublicSubnets

```

```

RegisterPrivateSubnetTags:
  Type: Custom::SubnetRegister
  Properties:
    ServiceToken: !GetAtt RegisterSubnetTags.Arn
    InfrastructureName: !Ref InfrastructureName
    Subnets: !Ref PrivateSubnets

```

```

Outputs:
PrivateHostedZoneId:
  Description: Hosted zone ID for the private DNS, which is required for private records.
  Value: !Ref IntDns
ExternalApiLoadBalancerName:
  Description: Full name of the external API load balancer.
  Value: !GetAtt ExtApiElb.LoadBalancerFullName
InternalApiLoadBalancerName:
  Description: Full name of the internal API load balancer.
  Value: !GetAtt IntApiElb.LoadBalancerFullName
ApiServerDnsName:
  Description: Full hostname of the API server, which is required for the Ignition config files.
  Value: !Join [".", ["api-int", !Ref ClusterName, !Ref HostedZoneName]]
RegisterNlbPTargetsLambda:
  Description: Lambda ARN useful to help register or deregister IP targets for these load
balancers.
  Value: !GetAtt RegisterNlbPTargets.Arn
ExternalApiTargetGroupArn:
  Description: ARN of the external API target group.
  Value: !Ref ExternalApiTargetGroup
InternalApiTargetGroupArn:
  Description: ARN of the internal API target group.
  Value: !Ref InternalApiTargetGroup
InternalServiceTargetGroupArn:
  Description: ARN of the internal service target group.
  Value: !Ref InternalServiceTargetGroup

```

1.8.10. Creating security group and roles in AWS

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



NOTE

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

Prerequisites

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

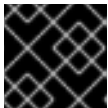
Procedure

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

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

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

- 6 Specify the **PrivateSubnetIds** value from the output of the CloudFormation template for the VPC.
 - 7 The VPC that you created for the cluster.
 - 8 Specify the **VpcId** value from the output of the CloudFormation template for the VPC.
2. Copy the template from the **CloudFormation template for security objects** section of this topic and save it as a YAML file on your computer. This template describes the security groups and roles that your cluster requires.
 3. Launch the template:



IMPORTANT

You must enter the command on a single line.

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

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

4. Confirm that the template components exist:

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

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

| | |
|---|-----------------------------|
| MasterSecurityGroup Id | Master Security Group ID |
| WorkerSecurityGroup Id | Worker Security Group ID |
| MasterInstanceProfile Id | Master IAM Instance Profile |

WorkerInstanceProfile

Worker IAM Instance Profile

1.8.10.1. CloudFormation template for security objects

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

Example 1.39. CloudFormation template for security objects

```

AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Security Elements (Security Groups & IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-\_]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
    maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
    used by the cluster.
    Type: String
  VpcCidr:
    AllowedPattern: ^((([0-9]{1-9}[0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]{1-9}[0-9]|1[0-9]{2}|2[0-
    4][0-9]|25[0-5])|(V(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>

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

```

VpcId:
 default: "VPC ID"
 VpcCidr:
 default: "VPC CIDR"
 PrivateSubnets:
 default: "Private Subnets"

Resources:

MasterSecurityGroup:

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

WorkerSecurityGroup:

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

MasterIngressEtcd:

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

MasterIngressVxlan:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Vxlan packets

FromPort: 4789

ToPort: 4789

IpProtocol: udp

MasterIngressWorkerVxlan:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Vxlan packets

FromPort: 4789

ToPort: 4789

IpProtocol: udp

MasterIngressGeneve:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Geneve packets

FromPort: 6081

ToPort: 6081

IpProtocol: udp

MasterIngressWorkerGeneve:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Geneve packets

FromPort: 6081

ToPort: 6081

IpProtocol: udp

MasterIngressInternal:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Internal cluster communication

FromPort: 9000

ToPort: 9999

IpProtocol: tcp

MasterIngressWorkerInternal:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Internal cluster communication

FromPort: 9000

ToPort: 9999

IpProtocol: tcp

MasterIngressInternalUDP:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Internal cluster communication

FromPort: 9000

ToPort: 9999

IpProtocol: udp

MasterIngressWorkerInternalUDP:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Internal cluster communication

FromPort: 9000

ToPort: 9999

IpProtocol: udp

MasterIngressKube:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Kubernetes kubelet, scheduler and controller manager

FromPort: 10250

ToPort: 10259

IpProtocol: tcp

MasterIngressWorkerKube:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Kubernetes kubelet, scheduler and controller manager

FromPort: 10250

ToPort: 10259

IpProtocol: tcp

MasterIngressIngressServices:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Kubernetes ingress services

FromPort: 30000

ToPort: 32767

IpProtocol: tcp

MasterIngressWorkerIngressServices:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
Description: Kubernetes ingress services
FromPort: 30000
ToPort: 32767
IpProtocol: tcp

MasterIngressIngressServicesUDP:

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

MasterIngressWorkerIngressServicesUDP:

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

WorkerIngressVxlan:

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

WorkerIngressMasterVxlan:

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

WorkerIngressGeneve:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt WorkerSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
Description: Geneve packets
FromPort: 6081
ToPort: 6081

IpProtocol: udp

WorkerIngressMasterGeneve:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Geneve packets

FromPort: 6081

ToPort: 6081

IpProtocol: udp

WorkerIngressInternal:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Internal cluster communication

FromPort: 9000

ToPort: 9999

IpProtocol: tcp

WorkerIngressMasterInternal:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Internal cluster communication

FromPort: 9000

ToPort: 9999

IpProtocol: tcp

WorkerIngressInternalUDP:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Internal cluster communication

FromPort: 9000

ToPort: 9999

IpProtocol: udp

WorkerIngressMasterInternalUDP:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Internal cluster communication

FromPort: 9000

ToPort: 9999

IpProtocol: udp

WorkerIngressKube:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
Description: Kubernetes secure kubelet port
FromPort: 10250
ToPort: 10250
IpProtocol: tcp

WorkerIngressWorkerKube:

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

WorkerIngressIngressServices:

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

WorkerIngressMasterIngressServices:

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

WorkerIngressIngressServicesUDP:

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

WorkerIngressMasterIngressServicesUDP:

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


```

MasterIamRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: "Allow"
          Principal:
            Service:
              - "ec2.amazonaws.com"
          Action:
            - "sts:AssumeRole"
    Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
            - Effect: "Allow"
              Action:
                - "ec2:AttachVolume"
                - "ec2:AuthorizeSecurityGroupIngress"
                - "ec2:CreateSecurityGroup"
                - "ec2:CreateTags"
                - "ec2:CreateVolume"
                - "ec2>DeleteSecurityGroup"
                - "ec2>DeleteVolume"
                - "ec2:Describe*"
                - "ec2:DetachVolume"
                - "ec2:ModifyInstanceAttribute"
                - "ec2:ModifyVolume"
                - "ec2:RevokeSecurityGroupIngress"
                - "elasticloadbalancing:AddTags"
                - "elasticloadbalancing:AttachLoadBalancerToSubnets"
                - "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer"
                - "elasticloadbalancing:CreateListener"
                - "elasticloadbalancing:CreateLoadBalancer"
                - "elasticloadbalancing:CreateLoadBalancerPolicy"
                - "elasticloadbalancing:CreateLoadBalancerListeners"
                - "elasticloadbalancing:CreateTargetGroup"
                - "elasticloadbalancing:ConfigureHealthCheck"
                - "elasticloadbalancing>DeleteListener"
                - "elasticloadbalancing>DeleteLoadBalancer"
                - "elasticloadbalancing>DeleteLoadBalancerListeners"
                - "elasticloadbalancing>DeleteTargetGroup"
                - "elasticloadbalancing:DeregisterInstancesFromLoadBalancer"
                - "elasticloadbalancing:DeregisterTargets"
                - "elasticloadbalancing:Describe*"
                - "elasticloadbalancing:DetachLoadBalancerFromSubnets"
                - "elasticloadbalancing:ModifyListener"
                - "elasticloadbalancing:ModifyLoadBalancerAttributes"
                - "elasticloadbalancing:ModifyTargetGroup"
                - "elasticloadbalancing:ModifyTargetGroupAttributes"
                - "elasticloadbalancing:RegisterInstancesWithLoadBalancer"
                - "elasticloadbalancing:RegisterTargets"
                - "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer"
                - "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"

```

- "kms:DescribeKey"

Resource: ""

MasterInstanceProfile:

Type: "AWS::IAM::InstanceProfile"

Properties:

Roles:

- Ref: "MasterIamRole"

WorkerIamRole:

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Principal:

Service:

- "ec2.amazonaws.com"

Action:

- "sts:AssumeRole"

Policies:

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

PolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Action:

- "ec2:DescribeInstances"

- "ec2:DescribeRegions"

Resource: ""

WorkerInstanceProfile:

Type: "AWS::IAM::InstanceProfile"

Properties:

Roles:

- Ref: "WorkerIamRole"

Outputs:

MasterSecurityGroupId:

Description: Master Security Group ID

Value: !GetAtt MasterSecurityGroup.GroupId

WorkerSecurityGroupId:

Description: Worker Security Group ID

Value: !GetAtt WorkerSecurityGroup.GroupId

MasterInstanceProfile:

Description: Master IAM Instance Profile

Value: !Ref MasterInstanceProfile

WorkerInstanceProfile:

Description: Worker IAM Instance Profile

Value: !Ref WorkerInstanceProfile

1.8.11. RHCOS AMIs for the AWS infrastructure

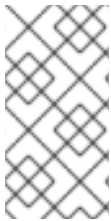
You must use a valid Red Hat Enterprise Linux CoreOS (RHCOS) AMI for your Amazon Web Services (AWS) zone for your OpenShift Container Platform nodes.

Table 1.17. RHCOS AMIs

| AWS zone | AWS AMI |
|----------------|-----------------------|
| ap-northeast-1 | ami-0530d04240177f118 |
| ap-northeast-2 | ami-09e4cd700276785d2 |
| ap-south-1 | ami-0754b15d212830477 |
| ap-southeast-1 | ami-03b46cc4b1518c5a8 |
| ap-southeast-2 | ami-0a5b99ab2234a4e6a |
| ca-central-1 | ami-012bc4ee3b6c673bc |
| eu-central-1 | ami-02e08df1201f1c2f8 |
| eu-north-1 | ami-0309c9d2fadcb2d5a |
| eu-west-1 | ami-0bdd69d8e7cd18188 |
| eu-west-2 | ami-0e610e967a62dbdfa |
| eu-west-3 | ami-0e817e26f638a71ac |
| me-south-1 | ami-024117d7c87b7ff08 |
| sa-east-1 | ami-08e62f746b94950c1 |
| us-east-1 | ami-077ede5bed2e431ea |
| us-east-2 | ami-0f4ecf819275850dd |
| us-west-1 | ami-0c4990e435bc6c5fe |
| us-west-2 | ami-000d6e92357ac605c |

1.8.12. Creating the bootstrap node in AWS

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

**NOTE**

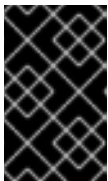
If you do not use the provided CloudFormation template to create your bootstrap node, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.
- Create and configure DNS, load balancers, and listeners in AWS.
- Create control plane and compute roles.

Procedure

1. Provide a location to serve the **bootstrap.ign** Ignition config file to your cluster. This file is located in your installation directory. One way to do this is to create an S3 bucket in your cluster's region and upload the Ignition config file to it.

**IMPORTANT**

The provided CloudFormation Template assumes that the Ignition config files for your cluster are served from an S3 bucket. If you choose to serve the files from another location, you must modify the templates.

**NOTE**

The bootstrap Ignition config file does contain secrets, like X.509 keys. The following steps provide basic security for the S3 bucket. To provide additional security, you can enable an S3 bucket policy to allow only certain users, such as the OpenShift IAM user, to access objects that the bucket contains. You can avoid S3 entirely and serve your bootstrap Ignition config file from any address that the bootstrap machine can reach.

- a. Create the bucket:

```
$ aws s3 mb s3://<cluster-name>-infra 1
```

- 1** **<cluster-name>-infra** is the bucket name.

- b. Upload the **bootstrap.ign** Ignition config file to the bucket:

```
$ aws s3 cp bootstrap.ign s3://<cluster-name>-infra/bootstrap.ign
```

- c. Verify that the file uploaded:

```
$ aws s3 ls s3://<cluster-name>-infra/
```

Example output

2019-04-03 16:15:16 314878 bootstrap.ign

2. Create a JSON file that contains the parameter values that the template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AllowedBootstrapSshCidr", 5
    "ParameterValue": "0.0.0.0/0" 6
  },
  {
    "ParameterKey": "PublicSubnet", 7
    "ParameterValue": "subnet-<random_string>" 8
  },
  {
    "ParameterKey": "MasterSecurityGroupId", 9
    "ParameterValue": "sg-<random_string>" 10
  },
  {
    "ParameterKey": "VpcId", 11
    "ParameterValue": "vpc-<random_string>" 12
  },
  {
    "ParameterKey": "BootstrapIgnitionLocation", 13
    "ParameterValue": "s3://<bucket_name>/bootstrap.ign" 14
  },
  {
    "ParameterKey": "AutoRegisterELB", 15
    "ParameterValue": "yes" 16
  },
  {
    "ParameterKey": "RegisterNlbTargetsLambdaArn", 17
    "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbTargets-<random_string>" 18
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", 19
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 20
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", 21
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:"
```

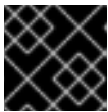
```

<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 22
    },
    {
      "ParameterKey": "InternalServiceTargetGroupArn", 23
      "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 24
    }
  ]

```

- 1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- 2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
- 3 Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the bootstrap node.
- 4 Specify a valid **AWS::EC2::Image::Id** value.
- 5 CIDR block to allow SSH access to the bootstrap node.
- 6 Specify a CIDR block in the format **x.x.x.x/16-24**.
- 7 The public subnet that is associated with your VPC to launch the bootstrap node into.
- 8 Specify the **PublicSubnetIds** value from the output of the CloudFormation template for the VPC.
- 9 The master security group ID (for registering temporary rules)
- 10 Specify the **MasterSecurityGroupId** value from the output of the CloudFormation template for the security group and roles.
- 11 The VPC created resources will belong to.
- 12 Specify the **VpcId** value from the output of the CloudFormation template for the VPC.
- 13 Location to fetch bootstrap Ignition config file from.
- 14 Specify the S3 bucket and file name in the form **s3://<bucket_name>/bootstrap.ign**.
- 15 Whether or not to register a network load balancer (NLB).
- 16 Specify **yes** or **no**. If you specify **yes**, you must provide a Lambda Amazon Resource Name (ARN) value.
- 17 The ARN for NLB IP target registration lambda group.
- 18 Specify the **RegisterNlbTargetsLambda** value from the output of the CloudFormation template for DNS and load balancing.
- 19 The ARN for external API load balancer target group.
- 20 Specify the **ExternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
- 21 The ARN for internal API load balancer target group.

- 22 Specify the **InternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
 - 23 The ARN for internal service load balancer target group.
 - 24 Specify the **InternalServiceTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
3. Copy the template from the **CloudFormation template for the bootstrap machine** section of this topic and save it as a YAML file on your computer. This template describes the bootstrap machine that your cluster requires.
 4. Launch the template:



IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> 1
  --template-body file://<template>.yaml 2
  --parameters file://<parameters>.json 3
  --capabilities CAPABILITY_NAMED_IAM
```

- 1 **<name>** is the name for the CloudFormation stack, such as **cluster-bootstrap**. You need the name of this stack if you remove the cluster.
- 2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- 3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

5. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

| | |
|-----------------------------|--|
| Bootstrap InstanceId | The bootstrap Instance ID. |
| Bootstrap PublicIp | The bootstrap node public IP address. |
| Bootstrap PrivateIp | The bootstrap node private IP address. |

1.8.12.1. CloudFormation template for the bootstrap machine

You can use the following CloudFormation template to deploy the bootstrap machine that you need for your OpenShift Container Platform cluster.

Example 1.40. CloudFormation template for the bootstrap machine

AWSTemplateFormatVersion: 2010-09-09

Description: Template for OpenShift Cluster Bootstrap (EC2 Instance, Security Groups and IAM)

Parameters:

InfrastructureName:

AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-_]{0,26})\$

MaxLength: 27

MinLength: 1

ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of 27 characters.

Description: A short, unique cluster ID used to tag cloud resources and identify items owned or used by the cluster.

Type: String

RhcosAmi:

Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.

Type: AWS::EC2::Image::Id

AllowedBootstrapSshCidr:

AllowedPattern: ^((([0-9]{1-9}|[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]{1-9}|[0-9]{2}|2[0-4][0-9]|25[0-5])|([0-9]{1-9}|[0-9]{2}|3[0-2]))\$

ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/0-32.

Default: 0.0.0.0/0

Description: CIDR block to allow SSH access to the bootstrap node.

Type: String

PublicSubnet:

Description: The public subnet to launch the bootstrap node into.

Type: AWS::EC2::Subnet::Id

MasterSecurityGroupId:

Description: The master security group ID for registering temporary rules.

Type: AWS::EC2::SecurityGroup::Id

VpcId:

Description: The VPC-scoped resources will belong to this VPC.

Type: AWS::EC2::VPC::Id

BootstrapIgnitionLocation:

Default: s3://my-s3-bucket/bootstrap.ign

Description: Ignition config file location.

Type: String

AutoRegisterELB:

Default: "yes"

AllowedValues:

- "yes"

- "no"

Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?

Type: String

RegisterNlbTargetsLambdaArn:

Description: ARN for NLB IP target registration lambda.

Type: String

ExternalApiTargetGroupArn:

Description: ARN for external API load balancer target group.

Type: String

InternalApiTargetGroupArn:

Description: ARN for internal API load balancer target group.

Type: String
 InternalServiceTargetGroupArn:
 Description: ARN for internal service load balancer target group.
 Type: String

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:

default: "Cluster Information"

Parameters:

- InfrastructureName

- Label:

default: "Host Information"

Parameters:

- RhcosAmi

- BootstrapIgnitionLocation

- MasterSecurityGroupId

- Label:

default: "Network Configuration"

Parameters:

- VpcId

- AllowedBootstrapSshCidr

- PublicSubnet

- Label:

default: "Load Balancer Automation"

Parameters:

- AutoRegisterELB

- RegisterNlbTargetsLambdaArn

- ExternalApiTargetGroupArn

- InternalApiTargetGroupArn

- InternalServiceTargetGroupArn

ParameterLabels:

InfrastructureName:

default: "Infrastructure Name"

VpcId:

default: "VPC ID"

AllowedBootstrapSshCidr:

default: "Allowed SSH Source"

PublicSubnet:

default: "Public Subnet"

RhcosAmi:

default: "Red Hat Enterprise Linux CoreOS AMI ID"

BootstrapIgnitionLocation:

default: "Bootstrap Ignition Source"

MasterSecurityGroupId:

default: "Master Security Group ID"

AutoRegisterELB:

default: "Use Provided ELB Automation"

Conditions:

DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]

Resources:

BootstrapIamRole:

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Principal:

Service:

- "ec2.amazonaws.com"

Action:

- "sts:AssumeRole"

Path: "/"

Policies:

- PolicyName: !Join ["-", [!Ref InfrastructureName, "bootstrap", "policy"]]

PolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Action: "ec2:Describe*"

Resource: ""

- Effect: "Allow"

Action: "ec2:AttachVolume"

Resource: ""

- Effect: "Allow"

Action: "ec2:DetachVolume"

Resource: ""

- Effect: "Allow"

Action: "s3:GetObject"

Resource: ""

BootstrapInstanceProfile:

Type: "AWS::IAM::InstanceProfile"

Properties:

Path: "/"

Roles:

- Ref: "BootstrapIamRole"

BootstrapSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: Cluster Bootstrap Security Group

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: 22

ToPort: 22

CidrIp: !Ref AllowedBootstrapSshCidr

- IpProtocol: tcp

ToPort: 19531

FromPort: 19531

CidrIp: 0.0.0.0/0

VpcId: !Ref VpcId

BootstrapInstance:

Type: AWS::EC2::Instance

Properties:

ImageId: !Ref RhcosAmi

IamInstanceProfile: !Ref BootstrapInstanceProfile

```

InstanceType: "i3.large"
NetworkInterfaces:
  - AssociatePublicIp: "true"
    DeviceIndex: "0"
    GroupSet:
      - !Ref "BootstrapSecurityGroup"
      - !Ref "MasterSecurityGroup"
    SubnetId: !Ref "PublicSubnet"
UserData:
  Fn::Base64: !Sub
    - '{"ignition":{"config":{"replace":{"source":"${S3Loc}","verification":{}}},"timeouts":
      {}, "version":"2.1.0"},"networkd":{},"passwd":{},"storage":{},"systemd":{}}'
    - {
        S3Loc: !Ref BootstrapIgnitionLocation
      }

RegisterBootstrapApiTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref ExternalApiTargetGroupArn
    TargetIp: !GetAtt BootstrapInstance.PrivateIp

RegisterBootstrapInternalApiTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref InternalApiTargetGroupArn
    TargetIp: !GetAtt BootstrapInstance.PrivateIp

RegisterBootstrapInternalServiceTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref InternalServiceTargetGroupArn
    TargetIp: !GetAtt BootstrapInstance.PrivateIp

Outputs:
BootstrapInstanceId:
  Description: Bootstrap Instance ID.
  Value: !Ref BootstrapInstance

BootstrapPublicIp:
  Description: The bootstrap node public IP address.
  Value: !GetAtt BootstrapInstance.PublicIp

BootstrapPrivateIp:
  Description: The bootstrap node private IP address.
  Value: !GetAtt BootstrapInstance.PrivateIp

```

1.8.13. Creating the control plane machines in AWS

You must create the control plane machines in Amazon Web Services (AWS) for your cluster to use. The easiest way to create these nodes is to modify the provided CloudFormation template.



NOTE

If you do not use the provided CloudFormation template to create your control plane nodes, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.
- Create and configure DNS, load balancers, and listeners in AWS.
- Create control plane and compute roles.
- Create the bootstrap machine.

Procedure

1. Create a JSON file that contains the parameter values that the template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcoshAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AutoRegisterDNS", 5
    "ParameterValue": "yes" 6
  },
  {
    "ParameterKey": "PrivateHostedZoneId", 7
    "ParameterValue": "<random_string>" 8
  },
  {
    "ParameterKey": "PrivateHostedZoneName", 9
    "ParameterValue": "mycluster.example.com" 10
  },
  {
    "ParameterKey": "Master0Subnet", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
```

```

    "ParameterKey": "Master1Subnet", 13
    "ParameterValue": "subnet-<random_string>" 14
  },
  {
    "ParameterKey": "Master2Subnet", 15
    "ParameterValue": "subnet-<random_string>" 16
  },
  {
    "ParameterKey": "MasterSecurityGroupId", 17
    "ParameterValue": "sg-<random_string>" 18
  },
  {
    "ParameterKey": "IgnitionLocation", 19
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/master"
  }, 20
  {
    "ParameterKey": "CertificateAuthorities", 21
    "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz==" 22
  },
  {
    "ParameterKey": "MasterInstanceProfileName", 23
    "ParameterValue": "<roles_stack>-MasterInstanceProfile-<random_string>" 24
  },
  {
    "ParameterKey": "MasterInstanceType", 25
    "ParameterValue": "m4.xlarge" 26
  },
  {
    "ParameterKey": "AutoRegisterELB", 27
    "ParameterValue": "yes" 28
  },
  {
    "ParameterKey": "RegisterNlbTargetsLambdaArn", 29
    "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbTargets-<random_string>" 30
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", 31
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 32
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", 33
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 34
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", 35
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 36
  }
]

```

-
- 1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- 2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
- 3 Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the control plane machines.
- 4 Specify an **AWS::EC2::Image::Id** value.
- 5 Whether or not to perform DNS etcd registration.
- 6 Specify **yes** or **no**. If you specify **yes**, you must provide hosted zone information.
- 7 The Route 53 private zone ID to register the etcd targets with.
- 8 Specify the **PrivateHostedZoneId** value from the output of the CloudFormation template for DNS and load balancing.
- 9 The Route 53 zone to register the targets with.
- 10 Specify **<cluster_name>.<domain_name>** where **<domain_name>** is the Route 53 base domain that you used when you generated **install-config.yaml** file for the cluster. Do not include the trailing period (.) that is displayed in the AWS console.
- 11 13 15 A subnet, preferably private, to launch the control plane machines on.
- 12 14 16 Specify a subnet from the **PrivateSubnets** value from the output of the CloudFormation template for DNS and load balancing.
- 17 The master security group ID to associate with master nodes.
- 18 Specify the **MasterSecurityGroupID** value from the output of the CloudFormation template for the security group and roles.
- 19 The location to fetch control plane Ignition config file from.
- 20 Specify the generated Ignition config file location, https://api-int.<cluster_name>.<domain_name>:22623/config/master.
- 21 The base64 encoded certificate authority string to use.
- 22 Specify the value from the **master.ign** file that is in the installation directory. This value is the long string with the format **data:text/plain;charset=utf-8;base64,ABC...xYz==**.
- 23 The IAM profile to associate with master nodes.
- 24 Specify the **MasterInstanceProfile** parameter value from the output of the CloudFormation template for the security group and roles.
- 25 The type of AWS instance to use for the control plane machines.
- 26 Allowed values:
 - **m4.xlarge**

- **m4.2xlarge**
- **m4.4xlarge**
- **m4.8xlarge**
- **m4.10xlarge**
- **m4.16xlarge**
- **c4.2xlarge**
- **c4.4xlarge**
- **c4.8xlarge**
- **r4.xlarge**
- **r4.2xlarge**
- **r4.4xlarge**
- **r4.8xlarge**
- **r4.16xlarge**

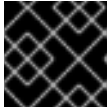


IMPORTANT

If **m4** instance types are not available in your region, such as with **eu-west-3**, specify an **m5** type, such as **m5.xlarge**, instead.

- 27** Whether or not to register a network load balancer (NLB).
- 28** Specify **yes** or **no**. If you specify **yes**, you must provide a Lambda Amazon Resource Name (ARN) value.
- 29** The ARN for NLB IP target registration lambda group.
- 30** Specify the **RegisterNlbTargetsLambda** value from the output of the CloudFormation template for DNS and load balancing.
- 31** The ARN for external API load balancer target group.
- 32** Specify the **ExternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
- 33** The ARN for internal API load balancer target group.
- 34** Specify the **InternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
- 35** The ARN for internal service load balancer target group.
- 36** Specify the **InternalServiceTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.

- Copy the template from the **CloudFormation template for control plane machines** section of this topic and save it as a YAML file on your computer. This template describes the control plane machines that your cluster requires.
- If you specified an **m5** instance type as the value for **MasterInstanceType**, add that instance type to the **MasterInstanceType.AllowedValues** parameter in the CloudFormation template.
- Launch the template:



IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> ❶
  --template-body file://<template>.yaml ❷
  --parameters file://<parameters>.json ❸
```

- ❶ **<name>** is the name for the CloudFormation stack, such as **cluster-control-plane**. You need the name of this stack if you remove the cluster.
- ❷ **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- ❸ **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

- Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

1.8.13.1. CloudFormation template for control plane machines

You can use the following CloudFormation template to deploy the control plane machines that you need for your OpenShift Container Platform cluster.

Example 1.41. CloudFormation template for control plane machines

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 master instances)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-\_]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
    maximum of 27 characters.
  Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
  Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AutoRegisterDNS:
```


Default: "yes"

AllowedValues:

- "yes"
- "no"

Description: Do you want to invoke DNS etcd registration, which requires Hosted Zone information?

Type: String

PrivateHostedZoneId:

Description: The Route53 private zone ID to register the etcd targets with, such as Z21XYZABCZ2A4.

Type: String

PrivateHostedZoneName:

Description: The Route53 zone to register the targets with, such as cluster.example.com. Omit the trailing period.

Type: String

Master0Subnet:

Description: The subnets, recommend private, to launch the master nodes into.

Type: AWS::EC2::Subnet::Id

Master1Subnet:

Description: The subnets, recommend private, to launch the master nodes into.

Type: AWS::EC2::Subnet::Id

Master2Subnet:

Description: The subnets, recommend private, to launch the master nodes into.

Type: AWS::EC2::Subnet::Id

MasterSecurityGroupId:

Description: The master security group ID to associate with master nodes.

Type: AWS::EC2::SecurityGroup::Id

IgnitionLocation:

Default: https://api-int.\$CLUSTER_NAME.\$DOMAIN:22623/config/master

Description: Ignition config file location.

Type: String

CertificateAuthorities:

Default: data:text/plain;charset=utf-8;base64,ABC...xYz==

Description: Base64 encoded certificate authority string to use.

Type: String

MasterInstanceProfileName:

Description: IAM profile to associate with master nodes.

Type: String

MasterInstanceType:

Default: m4.xlarge

Type: String

AllowedValues:

- "m4.xlarge"
- "m4.2xlarge"
- "m4.4xlarge"
- "m4.8xlarge"
- "m4.10xlarge"
- "m4.16xlarge"
- "c4.2xlarge"
- "c4.4xlarge"
- "c4.8xlarge"
- "r4.xlarge"
- "r4.2xlarge"
- "r4.4xlarge"
- "r4.8xlarge"
- "r4.16xlarge"

AutoRegisterELB:

Default: "yes"

AllowedValues:

- "yes"
- "no"

Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?

Type: String

RegisterNlbTargetsLambdaArn:

Description: ARN for NLB IP target registration lambda. Supply the value from the cluster infrastructure or select "no" for AutoRegisterELB.

Type: String

ExternalApiTargetGroupArn:

Description: ARN for external API load balancer target group. Supply the value from the cluster infrastructure or select "no" for AutoRegisterELB.

Type: String

InternalApiTargetGroupArn:

Description: ARN for internal API load balancer target group. Supply the value from the cluster infrastructure or select "no" for AutoRegisterELB.

Type: String

InternalServiceTargetGroupArn:

Description: ARN for internal service load balancer target group. Supply the value from the cluster infrastructure or select "no" for AutoRegisterELB.

Type: String

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:

default: "Cluster Information"

Parameters:

- InfrastructureName

- Label:

default: "Host Information"

Parameters:

- MasterInstanceType

- RhcosAmi

- IgnitionLocation

- CertificateAuthorities

- MasterSecurityGroupId

- MasterInstanceProfileName

- Label:

default: "Network Configuration"

Parameters:

- VpcId

- AllowedBootstrapSshCidr

- Master0Subnet

- Master1Subnet

- Master2Subnet

- Label:

default: "DNS"

Parameters:

- AutoRegisterDNS

- PrivateHostedZoneName

- PrivateHostedZoneId

- Label:

default: "Load Balancer Automation"

Parameters:

- AutoRegisterELB
- RegisterNLbIpTargetsLambdaArn
- ExternalApiTargetGroupArn
- InternalApiTargetGroupArn
- InternalServiceTargetGroupArn

ParameterLabels:

InfrastructureName:

default: "Infrastructure Name"

VpcId:

default: "VPC ID"

Master0Subnet:

default: "Master-0 Subnet"

Master1Subnet:

default: "Master-1 Subnet"

Master2Subnet:

default: "Master-2 Subnet"

MasterInstanceType:

default: "Master Instance Type"

MasterInstanceProfileName:

default: "Master Instance Profile Name"

RhcosAmi:

default: "Red Hat Enterprise Linux CoreOS AMI ID"

BootstrapIgnitionLocation:

default: "Master Ignition Source"

CertificateAuthorities:

default: "Ignition CA String"

MasterSecurityGroupId:

default: "Master Security Group ID"

AutoRegisterDNS:

default: "Use Provided DNS Automation"

AutoRegisterELB:

default: "Use Provided ELB Automation"

PrivateHostedZoneName:

default: "Private Hosted Zone Name"

PrivateHostedZoneId:

default: "Private Hosted Zone ID"

Conditions:

DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]

DoDns: !Equals ["yes", !Ref AutoRegisterDNS]

Resources:

Master0:

Type: AWS::EC2::Instance

Properties:

ImageId: !Ref RhcosAmi

BlockDeviceMappings:

- DeviceName: /dev/xvda

Ebs:

VolumeSize: "120"

VolumeType: "gp2"

IamInstanceProfile: !Ref MasterInstanceProfileName

InstanceType: !Ref MasterInstanceType

NetworkInterfaces:

- AssociatePublicIpAddress: "false"

```

    DeviceIndex: "0"
    GroupSet:
      - !Ref "MasterSecurityGroupId"
    SubnetId: !Ref "Master0Subnet"
    UserData:
      Fn::Base64: !Sub
        - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}],"security":{"tls":{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]},"timeouts":{"version":"2.2.0"},"networkd":{"passwd":{"storage":{"systemd":{"}}'
          - {
            SOURCE: !Ref IgnitionLocation,
            CA_BUNDLE: !Ref CertificateAuthorities,
          }
    Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

RegisterMaster0:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref ExternalApiTargetGroupArn
    TargetIp: !GetAtt Master0.PrivateIp

RegisterMaster0InternalApiTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref InternalApiTargetGroupArn
    TargetIp: !GetAtt Master0.PrivateIp

RegisterMaster0InternalServiceTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref InternalServiceTargetGroupArn
    TargetIp: !GetAtt Master0.PrivateIp

Master1:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref RhcosAmi
    BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
    IamInstanceProfile: !Ref MasterInstanceProfileName
    InstanceType: !Ref MasterInstanceType
    NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:

```

```

- !Ref "MasterSecurityGroupId"
SubnetId: !Ref "Master1Subnet"
UserData:
  Fn::Base64: !Sub
    - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}],"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]},"timeouts":
{"version":"2.2.0"},"networkd":{"},"passwd":{"},"storage":{"},"systemd":{"}}'
    - {
      SOURCE: !Ref IgnitionLocation,
      CA_BUNDLE: !Ref CertificateAuthorities,
    }
Tags:
- Key: !Join ["/"], ["kubernetes.io/cluster/", !Ref InfrastructureName]]
  Value: "shared"

```

```

RegisterMaster1:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref ExternalApiTargetGroupArn
    TargetIp: !GetAtt Master1.PrivateIp

```

```

RegisterMaster1InternalApiTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref InternalApiTargetGroupArn
    TargetIp: !GetAtt Master1.PrivateIp

```

```

RegisterMaster1InternalServiceTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref InternalServiceTargetGroupArn
    TargetIp: !GetAtt Master1.PrivateIp

```

```

Master2:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref RhcosAmi
    BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
    IamInstanceProfile: !Ref MasterInstanceProfileName
    InstanceType: !Ref MasterInstanceType
    NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
          - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master2Subnet"

```

```

    UserData:
      Fn::Base64: !Sub
        - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}],"security":{"tls":{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]},"timeouts":{"version":"2.2.0"},"networkd":{"},"passwd":{"},"storage":{"},"systemd":{"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
        - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
          Value: "shared"

    RegisterMaster2:
      Condition: DoRegistration
      Type: Custom::NLBRegister
      Properties:
        ServiceToken: !Ref RegisterNlbTargetsLambdaArn
        TargetArn: !Ref ExternalApiTargetGroupArn
        TargetIp: !GetAtt Master2.PrivateIp

    RegisterMaster2InternalApiTarget:
      Condition: DoRegistration
      Type: Custom::NLBRegister
      Properties:
        ServiceToken: !Ref RegisterNlbTargetsLambdaArn
        TargetArn: !Ref InternalApiTargetGroupArn
        TargetIp: !GetAtt Master2.PrivateIp

    RegisterMaster2InternalServiceTarget:
      Condition: DoRegistration
      Type: Custom::NLBRegister
      Properties:
        ServiceToken: !Ref RegisterNlbTargetsLambdaArn
        TargetArn: !Ref InternalServiceTargetGroupArn
        TargetIp: !GetAtt Master2.PrivateIp

    EtcdSrvRecords:
      Condition: DoDns
      Type: AWS::Route53::RecordSet
      Properties:
        HostedZoneId: !Ref PrivateHostedZoneId
        Name: !Join [".", ["_etcd-server-ssl._tcp", !Ref PrivateHostedZoneName]]
        ResourceRecords:
          - !Join [
              " ",
              ["0 10 2380", !Join [".", ["etcd-0", !Ref PrivateHostedZoneName]]],
            ]
          - !Join [
              " ",
              ["0 10 2380", !Join [".", ["etcd-1", !Ref PrivateHostedZoneName]]],
            ]
          - !Join [
              " ",
              ["0 10 2380", !Join [".", ["etcd-2", !Ref PrivateHostedZoneName]]],
            ]

```

```
TTL: 60
Type: SRV
```

```
Etcd0Record:
  Condition: DoDns
  Type: AWS::Route53::RecordSet
  Properties:
    HostedZoneId: !Ref PrivateHostedZoneId
    Name: !Join [".", ["etcd-0", !Ref PrivateHostedZoneName]]
    ResourceRecords:
      - !GetAtt Master0.PrivateIp
    TTL: 60
    Type: A
```

```
Etcd1Record:
  Condition: DoDns
  Type: AWS::Route53::RecordSet
  Properties:
    HostedZoneId: !Ref PrivateHostedZoneId
    Name: !Join [".", ["etcd-1", !Ref PrivateHostedZoneName]]
    ResourceRecords:
      - !GetAtt Master1.PrivateIp
    TTL: 60
    Type: A
```

```
Etcd2Record:
  Condition: DoDns
  Type: AWS::Route53::RecordSet
  Properties:
    HostedZoneId: !Ref PrivateHostedZoneId
    Name: !Join [".", ["etcd-2", !Ref PrivateHostedZoneName]]
    ResourceRecords:
      - !GetAtt Master2.PrivateIp
    TTL: 60
    Type: A
```

```
Outputs:
  PrivateIPs:
    Description: The control-plane node private IP addresses.
    Value:
      !Join [
        ",",
        [!GetAtt Master0.PrivateIp, !GetAtt Master1.PrivateIp, !GetAtt Master2.PrivateIp]
      ]
```

1.8.14. Initializing the bootstrap node on AWS with user-provisioned infrastructure

After you create all of the required infrastructure in Amazon Web Services (AWS), you can install the cluster.

Prerequisites

- Configure an AWS account.

- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.
- Create and configure DNS, load balancers, and listeners in AWS.
- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.
- If you plan to manually manage the worker machines, create the worker machines.

Procedure

1. Change to the directory that contains the installation program and run the following command:

```
$ ./openshift-install wait-for bootstrap-complete --dir=<installation_directory> \ 1
--log-level=info 2
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2** To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

If the command exits without a **FATAL** warning, your production control plane has initialized.

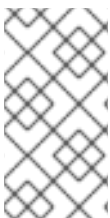
1.8.14.1. Creating the worker nodes in AWS

You can create worker nodes in Amazon Web Services (AWS) for your cluster to use. The easiest way to manually create these nodes is to modify the provided CloudFormation template.



IMPORTANT

The CloudFormation template creates a stack that represents one worker machine. You must create a stack for each worker machine.



NOTE

If you do not use the provided CloudFormation template to create your worker nodes, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.
- Create and configure DNS, load balancers, and listeners in AWS.

- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.

Procedure

1. Create a JSON file that contains the parameter values that the CloudFormation template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "Subnet", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "WorkerSecurityGroup", 7
    "ParameterValue": "sg-<random_string>" 8
  },
  {
    "ParameterKey": "IgnitionLocation", 9
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/worker"
  } 10
  {
    "ParameterKey": "CertificateAuthorities", 11
    "ParameterValue": "" 12
  },
  {
    "ParameterKey": "WorkerInstanceProfileName", 13
    "ParameterValue": "" 14
  },
  {
    "ParameterKey": "WorkerInstanceType", 15
    "ParameterValue": "m4.large" 16
  }
]
```

- 1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- 2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.

- 3 Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the worker nodes.
- 4 Specify an **AWS::EC2::Image::Id** value.
- 5 A subnet, preferably private, to launch the worker nodes on.
- 6 Specify a subnet from the **PrivateSubnets** value from the output of the CloudFormation template for DNS and load balancing.
- 7 The worker security group ID to associate with worker nodes.
- 8 Specify the **WorkerSecurityGroupID** value from the output of the CloudFormation template for the security group and roles.
- 9 The location to fetch bootstrap Ignition config file from.
- 10 Specify the generated Ignition config location, https://api-int.<cluster_name>.<domain_name>:22623/config/worker.
- 11 Base64 encoded certificate authority string to use.
- 12 Specify the value from the **worker.ign** file that is in the installation directory. This value is the long string with the format **data:text/plain;charset=utf-8;base64,ABC...xYz==**.
- 13 The IAM profile to associate with worker nodes.
- 14 Specify the **WorkerInstanceProfile** parameter value from the output of the CloudFormation template for the security group and roles.
- 15 The type of AWS instance to use for the control plane machines.
- 16 Allowed values:
 - **m4.large**
 - **m4.xlarge**
 - **m4.2xlarge**
 - **m4.4xlarge**
 - **m4.8xlarge**
 - **m4.10xlarge**
 - **m4.16xlarge**
 - **c4.large**
 - **c4.xlarge**
 - **c4.2xlarge**
 - **c4.4xlarge**
 - **c4.8xlarge**
 - **r4.large**

- **r4.xlarge**
- **r4.2xlarge**
- **r4.4xlarge**
- **r4.8xlarge**
- **r4.16xlarge**



IMPORTANT

If **m4** instance types are not available in your region, such as with **eu-west-3**, use **m5** types instead.

2. Copy the template from the **CloudFormation template for worker machines** section of this topic and save it as a YAML file on your computer. This template describes the networking objects and load balancers that your cluster requires.
3. If you specified an **m5** instance type as the value for **WorkerInstanceType**, add that instance type to the **WorkerInstanceType.AllowedValues** parameter in the CloudFormation template.
4. Create a worker stack.
 - a. Launch the template:



IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> ❶
  --template-body file://<template>.yaml \ ❷
  --parameters file://<parameters>.json ❸
```

- ❶ **<name>** is the name for the CloudFormation stack, such as **cluster-workers**. You need the name of this stack if you remove the cluster.
- ❷ **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- ❸ **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

- b. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

5. Continue to create worker stacks until you have created enough worker machines for your cluster.

**IMPORTANT**

You must create at least two worker machines, so you must create at least two stacks that use this CloudFormation template.

1.8.14.1.1. CloudFormation template for worker machines

You can use the following CloudFormation template to deploy the worker machines that you need for your OpenShift Container Platform cluster.

Example 1.42. CloudFormation template for worker machines

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 worker instance)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-\_]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
    maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  WorkerSecurityGroupId:
    Description: The master security group ID to associate with master nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/worker
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  WorkerInstanceProfileName:
    Description: IAM profile to associate with master nodes.
    Type: String
  WorkerInstanceType:
    Default: m4.large
    Type: String
    AllowedValues:
      - "m4.large"
      - "m4.xlarge"
      - "m4.2xlarge"
      - "m4.4xlarge"
      - "m4.8xlarge"
      - "m4.10xlarge"
      - "m4.16xlarge"
      - "c4.large"
```

- "c4.xlarge"
- "c4.2xlarge"
- "c4.4xlarge"
- "c4.8xlarge"
- "r4.large"
- "r4.xlarge"
- "r4.2xlarge"
- "r4.4xlarge"
- "r4.8xlarge"
- "r4.16xlarge"

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:

default: "Cluster Information"

Parameters:

- InfrastructureName

- Label:

default: "Host Information"

Parameters:

- WorkerInstanceType

- RhcosAmi

- IgnitionLocation

- CertificateAuthorities

- WorkerSecurityGroupId

- WorkerInstanceProfileName

- Label:

default: "Network Configuration"

Parameters:

- Subnet

ParameterLabels:

Subnet:

default: "Subnet"

InfrastructureName:

default: "Infrastructure Name"

WorkerInstanceType:

default: "Worker Instance Type"

WorkerInstanceProfileName:

default: "Worker Instance Profile Name"

RhcosAmi:

default: "Red Hat Enterprise Linux CoreOS AMI ID"

IgnitionLocation:

default: "Worker Ignition Source"

CertificateAuthorities:

default: "Ignition CA String"

WorkerSecurityGroupId:

default: "Worker Security Group ID"

Resources:

Worker0:

Type: AWS::EC2::Instance

Properties:

ImageId: !Ref RhcosAmi

BlockDeviceMappings:

- DeviceName: /dev/xvda

```

Ebs:
  VolumeSize: "120"
  VolumeType: "gp2"
IamInstanceProfile: !Ref WorkerInstanceProfileName
InstanceType: !Ref WorkerInstanceType
NetworkInterfaces:
- AssociatePublicIpAddress: "false"
  DeviceIndex: "0"
  GroupSet:
  - !Ref "WorkerSecurityGroupId"
  SubnetId: !Ref "Subnet"
UserData:
  Fn::Base64: !Sub
    - {"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}],"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}],"timeouts":
{"version":"2.2.0"},"networkd":{"passwd":{"storage":{"systemd":{"
- {
  SOURCE: !Ref IgnitionLocation,
  CA_BUNDLE: !Ref CertificateAuthorities,
}
}
Tags:
- Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
  Value: "shared"

Outputs:
PrivatelP:
  Description: The compute node private IP address.
  Value: !GetAtt Worker0.PrivateIp

```

1.8.15. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

■

```
$ oc whoami
```

Example output

```
system:admin
```

1.8.16. 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   v1.18.3
master-1  Ready    master   63m   v1.18.3
master-2  Ready    master   64m   v1.18.3
worker-0  NotReady worker   76s   v1.18.3
worker-1  NotReady worker   70s   v1.18.3
```

The output lists all of the machines that you created.

- 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 you approve the initial CSRs, the subsequent node client CSRs are automatically approved by the cluster **kube-controller-manager**. You must implement a method of automatically approving the kubelet serving certificate requests.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** is the name of a CSR from the list of current CSRs.

- 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
```

- 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  v1.20.0
master-1  Ready   master 73m  v1.20.0
master-2  Ready   master 74m  v1.20.0
worker-0  Ready   worker 11m  v1.20.0
worker-1  Ready   worker 11m  v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

1.8.17. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

- Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

Example output

| NAME | VERSION | AVAILABLE | PROGRESSING | DEGRADED |
|-------------------------|---------|-----------|-------------|-------------|
| SINCE | | | | |
| authentication | 4.5.4 | True | False | False 69s |
| cloud-credential | 4.5.4 | True | False | False 12m |
| cluster-autoscaler | 4.5.4 | True | False | False 11m |
| console | 4.5.4 | True | False | False 46s |
| dns | 4.5.4 | True | False | False 11m |
| image-registry | 4.5.4 | True | False | False 5m26s |
| ingress | 4.5.4 | True | False | False 5m36s |
| kube-apiserver | 4.5.4 | True | False | False 8m53s |
| kube-controller-manager | 4.5.4 | True | False | False 7m24s |
| kube-scheduler | 4.5.4 | True | False | False 12m |
| machine-api | 4.5.4 | True | False | False 12m |
| machine-config | 4.5.4 | True | False | False 7m36s |
| marketplace | 4.5.4 | True | False | False 7m54m |
| monitoring | 4.5.4 | True | False | False 7h54s |

| | | | | | |
|------------------------------------|-------|------|-------|-------|--------|
| network | 4.5.4 | True | False | False | 5m9s |
| node-tuning | 4.5.4 | True | False | False | 11m |
| openshift-apiserver | 4.5.4 | True | False | False | 11m |
| openshift-controller-manager | 4.5.4 | True | False | False | 5m943s |
| openshift-samples | 4.5.4 | True | False | False | 3m55s |
| operator-lifecycle-manager | 4.5.4 | True | False | False | 11m |
| operator-lifecycle-manager-catalog | 4.5.4 | True | False | False | 11m |
| service-ca | 4.5.4 | True | False | False | 11m |
| service-catalog-apiserver | 4.5.4 | True | False | False | 5m26s |
| service-catalog-controller-manager | 4.5.4 | True | False | False | 5m25s |
| storage | 4.5.4 | True | False | False | 5m30s |

2. Configure the Operators that are not available.

1.8.17.1. Image registry storage configuration

Amazon Web Services provides default storage, which means the Image Registry Operator is available after installation. However, if the Registry Operator cannot create an S3 bucket and automatically configure storage, you must manually configure registry storage.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

1.8.17.1.1. Configuring registry storage for AWS with user-provisioned infrastructure

During installation, your cloud credentials are sufficient to create an Amazon S3 bucket and the Registry Operator will automatically configure storage.

If the Registry Operator cannot create an S3 bucket and automatically configure storage, you can create an S3 bucket and configure storage with the following procedure.

Prerequisites

- A cluster on AWS with user-provisioned infrastructure.
- For Amazon S3 storage, the secret is expected to contain two keys:
 - **REGISTRY_STORAGE_S3_ACCESSKEY**
 - **REGISTRY_STORAGE_S3_SECRETKEY**

Procedure

Use the following procedure if the Registry Operator cannot create an S3 bucket and automatically configure storage.

1. Set up a [Bucket Lifecycle Policy](#) to abort incomplete multipart uploads that are one day old.
2. Fill in the storage configuration in **configs.imageregistry.operator.openshift.io/cluster**:

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

Example configuration

```
storage:
  s3:
    bucket: <bucket-name>
    region: <region-name>
```



WARNING

To secure your registry images in AWS, [block public access](#) to the S3 bucket.

1.8.17.1.2. Configuring storage for the image registry in non-production clusters

You must configure storage for the Image Registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

Procedure

1. To set the image registry storage to an empty directory:

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec": {"storage":{"emptyDir":{}}}}'
```



WARNING

Configure this option for only non-production clusters.

If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

Wait a few minutes and run the command again.

2. Ensure that your registry is set to managed to enable building and pushing of images.
 - Run:

```
$ oc edit configs.imageregistry/cluster
```

Then, change the line

```
managementState: Removed
```

to

```
managementState: Managed
```

1.8.18. Deleting the bootstrap resources

After you complete the initial Operator configuration for the cluster, remove the bootstrap resources from Amazon Web Services (AWS).

Prerequisites

- You completed the initial Operator configuration for your cluster.

Procedure

- Delete the bootstrap resources. If you used the CloudFormation template, [delete its stack](#):

```
$ aws cloudformation delete-stack --stack-name <name> 1
```

1 **<name>** is the name of your bootstrap stack.

1.8.19. Creating the Ingress DNS Records

If you removed the DNS Zone configuration, manually create DNS records that point to the Ingress load balancer. You can create either a wildcard record or specific records. While the following procedure uses A records, you can use other record types that you require, such as CNAME or alias.

Prerequisites

- You deployed an OpenShift Container Platform cluster on Amazon Web Services (AWS) that uses infrastructure that you provisioned.
- Install the OpenShift CLI (**oc**).
- Install the **jq** package.
- Download the AWS CLI and install it on your computer. See [Install the AWS CLI Using the Bundled Installer \(Linux, macOS, or Unix\)](#).

Procedure

- Determine the routes to create.
 - To create a wildcard record, use ***.apps.<cluster_name>.<domain_name>**, where **<cluster_name>** is your cluster name, and **<domain_name>** is the Route 53 base domain for your OpenShift Container Platform cluster.
 - To create specific records, you must create a record for each route that your cluster uses, as shown in the output of the following command:

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

Example output

```

oauth-openshift.apps.<cluster_name>.<domain_name>
console-openshift-console.apps.<cluster_name>.<domain_name>
downloads-openshift-console.apps.<cluster_name>.<domain_name>
alertmanager-main-openshift-monitoring.apps.<cluster_name>.<domain_name>
grafana-openshift-monitoring.apps.<cluster_name>.<domain_name>
prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<domain_name>

```

2. Retrieve the Ingress Operator load balancer status and note the value of the external IP address that it uses, which is shown in the **EXTERNAL-IP** column:

```
$ oc -n openshift-ingress get service router-default
```

Example output

| NAME | TYPE | CLUSTER-IP | EXTERNAL-IP | PORT(S) |
|----------------|--------------|---------------|--------------------------------------|----------------------------|
| router-default | LoadBalancer | 172.30.62.215 | ab3...28.us-east-2.elb.amazonaws.com | 80:31499/TCP,443:30693/TCP |
| | | 5m | | |

3. Locate the hosted zone ID for the load balancer:

```
$ aws elb describe-load-balancers | jq -r '.LoadBalancerDescriptions[] | select(.DNSName == "<external_ip>").CanonicalHostedZoneNameID' 1
```

- 1** For **<external_ip>**, specify the value of the external IP address of the Ingress Operator load balancer that you obtained.

Example output

```
Z3AADJGX6KTTL2
```

The output of this command is the load balancer hosted zone ID.

4. Obtain the public hosted zone ID for your cluster's domain:

```
$ aws route53 list-hosted-zones-by-name \
  --dns-name "<domain_name>" 1
  --query 'HostedZones[? Config.PrivateZone != `true` && Name ==
`<domain_name>.`].Id' 2
  --output text
```

- 1** **2** For **<domain_name>**, specify the Route 53 base domain for your OpenShift Container Platform cluster.

Example output

```
/hostedzone/Z3URY6TWQ91KVV
```

The public hosted zone ID for your domain is shown in the command output. In this example, it is **Z3URY6TWQ91KVV**.

5. Add the alias records to your private zone:

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<private_hosted_zone_id>" --
change-batch '{
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>",
>         "Type": "A",
>         "AliasTarget": {
>           "HostedZoneId": "<hosted_zone_id>",
>           "DNSName": "<external_ip>.",
>           "EvaluateTargetHealth": false
>         }
>       }
>     ]
>   }
```

- 1** For **<private_hosted_zone_id>**, specify the value from the output of the CloudFormation template for DNS and load balancing.
- 2** For **<cluster_domain>**, specify the domain or subdomain that you use with your OpenShift Container Platform cluster.
- 3** For **<hosted_zone_id>**, specify the public hosted zone ID for the load balancer that you obtained.
- 4** For **<external_ip>**, specify the value of the external IP address of the Ingress Operator load balancer. Ensure that you include the trailing period (.) in this parameter value.

6. Add the records to your public zone:

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<public_hosted_zone_id>" --
change-batch '{
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>",
>         "Type": "A",
>         "AliasTarget": {
>           "HostedZoneId": "<hosted_zone_id>",
>           "DNSName": "<external_ip>.",
>           "EvaluateTargetHealth": false
>         }
>       }
>     ]
>   }
```

- 1 For **<public_hosted_zone_id>**, specify the public hosted zone for your domain.
- 2 For **<cluster_domain>**, specify the domain or subdomain that you use with your OpenShift Container Platform cluster.
- 3 For **<hosted_zone_id>**, specify the public hosted zone ID for the load balancer that you obtained.
- 4 For **<external_ip>**, specify the value of the external IP address of the Ingress Operator load balancer. Ensure that you include the trailing period (.) in this parameter value.

1.8.20. Completing an AWS installation on user-provisioned infrastructure

After you start the OpenShift Container Platform installation on Amazon Web Service (AWS) user-provisioned infrastructure, monitor the deployment to completion.

Prerequisites

- Removed the bootstrap node for an OpenShift Container Platform cluster on user-provisioned AWS infrastructure.
- Install the **oc** CLI and log in.

Procedure

1. Complete the cluster installation:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
INFO Waiting up to 30m0s for the cluster to initialize...
```



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-bootstrap** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

2. Register your cluster on the [Cluster registration](#) page.

1.8.21. Next steps

- [Customize your cluster.](#)

- 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, you can [remove cloud provider credentials](#).

1.9. UNINSTALLING A CLUSTER ON AWS

You can remove a cluster that you deployed to Amazon Web Services (AWS).

1.9.1. 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.

Prerequisites

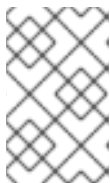
- Have a copy of the installation program that you used to deploy the cluster.
- Have the files that the installation program generated when you created your cluster.

Procedure

1. From the computer that you used to install the cluster, run the following command:

```
$ ./openshift-install destroy cluster \
--dir=<installation_directory> --log-level=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.