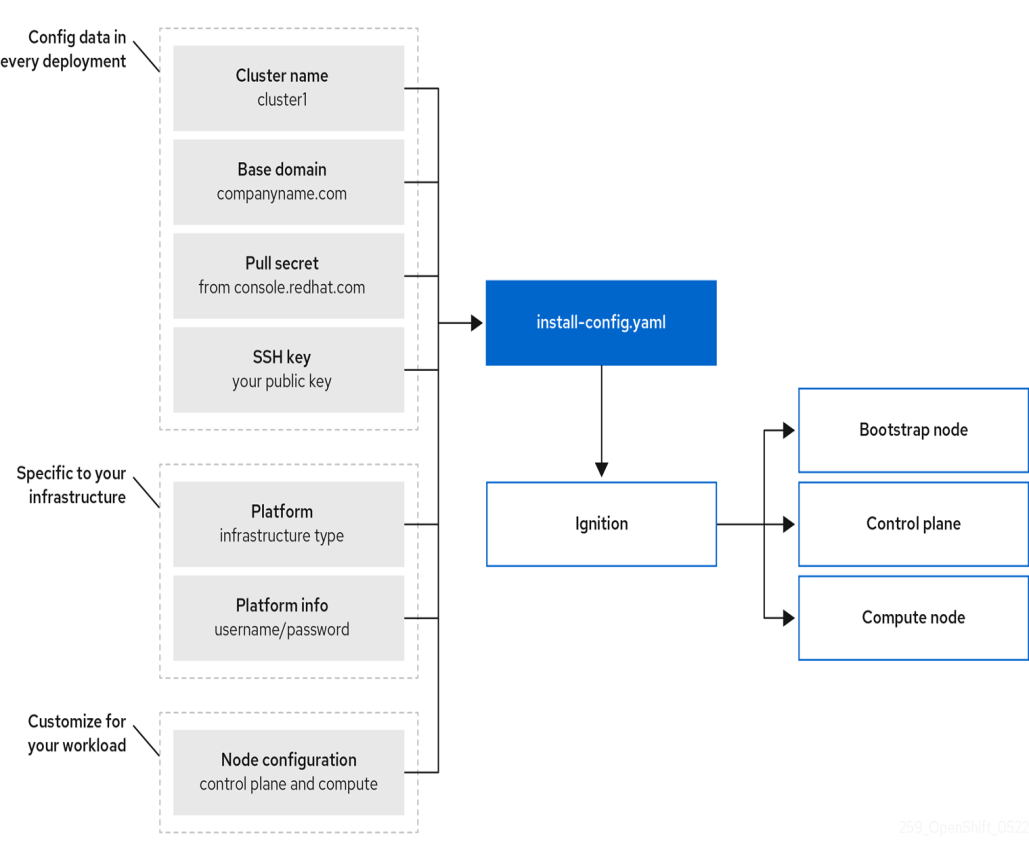
# OpenShift Container Platform 4.12

Installing a user-provisioned cluster on bare meta

* OpenShift Container Platform installation targets and dependencies



#### Installation process details C:\Users\harish\Desktop\download (1).png

**Ignition**

A utility that RHCOS uses to manipulate disks during initial configuration. It completes common disk tasks, including partitioning disks, formatting partitions, writing files, and configuring users.

**installer-provisioned infrastructure**

The installation program deploys and configures the infrastructure that the cluster runs on.

**kubelet**

A primary node agent that runs on each node in the cluster to ensure that containers are running in a pod.

**kubernetes manifest**

Specifications of a Kubernetes API object in a JSON or YAML format. A configuration file can include deployments, config maps, secrets, daemon sets.

**bootstrap**

A temporary machine that runs minimal Kubernetes and deploys the OpenShift Container Platform control plane.

**certificate signing requests (CSRs)**

A resource requests a denoted signer to sign a certificate. This request might get approved or denied.

**Cluster Version Operator (CVO)**

An Operator that checks with the OpenShift Container Platform Update Service to see the valid updates and update paths based on current component versions and information in the graph.

**compute nodes**

Nodes that are responsible for executing workloads for cluster users. Compute nodes are also known as worker nodes.

**control plane**

A container orchestration layer that exposes the API and interfaces to define, deploy, and manage the life cycle of containers. Control planes are also known as control plane machines.

**CRI-O**

A Kubernetes native container runtime implementation that integrates with the operating system to deliver an efficient Kubernetes experience.

**Operator**

The preferred method of packaging, deploying, and managing a Kubernete application in an OpenShift Container Platform cluster.

An Operator takes human operational knowledge and encodes it into software that is packaged and shared with customers.

**OperatorHub**

A platform that contains various OpenShift Container Platform Operators to install.

**Operator Lifecycle Manager (OLM)**

OLM helps you to install, update, and manage the lifecycle of Kubernetes native applications. OLM is an open source toolkit designed to manage Operators in an effective, automated, and scalable way.

**Machine Config Operator (MCO)**

An Operator that applies the new configuration to your cluster machines.

**machine config pools (MCP)**

A group of machines, such as control plane components or user workloads, that are based on the resources that they handle.

### Required machines for cluster installation

The bootstrap and control plane machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system. However, the compute machines can choose between Red Hat Enterprise Linux CoreOS (RHCOS), Red Hat Enterprise Linux (RHEL) 8.4, or RHEL 8.5.

### Minimum resource requirements for cluster installation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Machine | Operating System | CPU | RAM | Storage |
| Bastion | RHEL 8 | 4 | 16 | 100 GB |
| Bootstrap | RHCOS | 4 | 16 | 100 GB |
| Control plane | RHCOS | 4 | 16 | 100 GB |
| Compute | RHCOS, | 2 | 8 GB | 100 GB |

**Network Services**

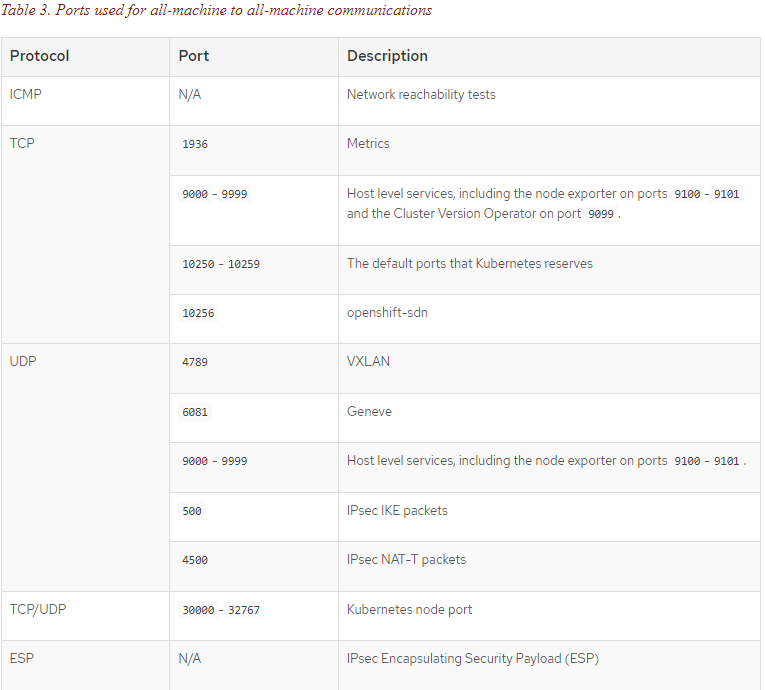
DNS Server: 192.168.0.51/24

HTTP Server: 192.168.0.51/24

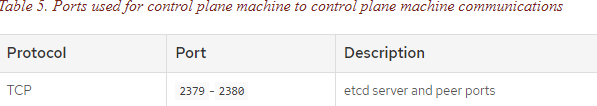
Load Balancer Config Details

**Nodes data**

|  |  |  |  |
| --- | --- | --- | --- |
| Server FQDN | IP | Role | OS |
| Bastion.ocp4-dev.pd.in | 192.168.0.51 | Bastion | RHEL 8 |
| Bootstrap.ocp4-dev.pd.in | 192.168.0.52 | Bootstrap | RHCOS |
| Master0.ocp4-dev.pd.in | 192.168.0.53 | Master | RHCOS |
| Master1.ocp4-dev.pd.in | 192.168.0.54 | Master | RHCOS |
| Master2.ocp4-dev.pd.in | 192.168.0.55 | Master | RHCOS |
| Worker0.ocp4-dev.pd.in | 192.168.0.56 | Worker | RHCOS |
| Worker1.ocp4-dev.pd.in | 192.168.0.57 | Worker | RHCOS |







Creating Bastion Hosts:

CREATING THE DNS CONFIGRATION IN THE BASTION

Before starting the dns server configrations we need to check selinux status ,it should be disable

we need to stop the firewalld service also

We need to set hostname

|  |
| --- |
| **[root@localhost ~]#** hostnamectl set-hostname bastion.ocp4-dev.pd.in  **[root@bastion ~]#** hostname  bastion.ocp4-dev.pd.in  **[root@ocp4-dev ~]#** nmcli connection show  NAME UUID TYPE DEVICE  ens192 d58103bd-718b-4d98-85a0-9debc10eaeaf ethernet ens192  virbr0 7c531db4-eef5-449a-bedc-04e27af44d13 bridge virbr0  **[root@ocp4-dev ~]#** nmcli connection modify "ens192" ipv4.addresses "192.168.0.51/24" ipv4.dns "192.168.0.51" +ipv4.dns "8.8.8.8" +ipv4.dns "4.2.2.2" ipv4.gateway "192.168.0.1" ipv4.dns-search "ocp4-dev.pd.in" +ipv4.dns-search "pd.in" ipv4.method manual  **[root@ocp4-dev ~]#** nmcli connection reload  **[root@ocp4-dev ~]#** nmcli connection show “ens192”  **[root@ocp4-dev ~]#** subscription-manager register  **[root@ocp4-dev ~]#** subscription-manager list  **[root@ocp4-dev ~]#** yum repolist  Updating Subscription Management repositories.  repo id repo name  rhel-8-for-x86\_64-appstream-rpms Red Hat Enterprise Linux 8 for x86\_64 - AppStream (RPMs)  rhel-8-for-x86\_64-baseos-rpms Red Hat Enterprise Linux 8 for x86\_64 - BaseOS (RPMs)  yum list bind bind-utils httpd haproxy  **[root@ocp4-dev ~]#** systemctl status firewalld.service  **[root@ocp4-dev ~]#** systemctl disable firewalld.service  **[root@ocp4-dev ~]#** systemctl stop firewalld.service  **[root@ocp4-dev ~]#** systemctl mask firewalld.service  **[root@ocp4-dev ~]#** vim /etc/selinux/config  SELINUX=disabled  :wq!  **[root@ocp4-dev ~]#** systemctl status firewalld.service  ● firewalld.service  Loaded: masked (Reason: Unit firewalld.service is masked.)  **[root@ocp4-dev ~]#** systemctl enable httpd.service  **[root@ocp4-dev ~]#** systemctl start httpd.service  **[root@ocp4-dev ~]#** systemctl enable named.service  **[root@ocp4-dev ~]#** systemctl start named.service  **[root@ocp4-dev ~]#** systemctl enable haproxy.service  **[root@ocp4-dev ~]#** systemctl start haproxy.service |

#### DNS configuration for user-provisioned clusters

**[root@ocp4-dev ~]#** cat /etc/named.conf

|  |
| --- |
| //  // named.conf  //  // Provided by Red Hat bind package to configure the ISC BIND named(8) DNS  // server as a caching only nameserver (as a localhost DNS resolver only).  //  // See /usr/share/doc/bind\*/sample/ for example named configuration files.  //  options {  listen-on port 53 { 127.0.0.1; any; };  listen-on-v6 port 53 { ::1; };  directory "/var/named";  dump-file "/var/named/data/cache\_dump.db";  statistics-file "/var/named/data/named\_stats.txt";  memstatistics-file "/var/named/data/named\_mem\_stats.txt";  secroots-file "/var/named/data/named.secroots";  recursing-file "/var/named/data/named.recursing";  allow-query { localhost; any;};  forwarders {8.8.8.8; 4.2.2.2;};  /\*  - If you are building an AUTHORITATIVE DNS server, do NOT enable recursion.  - If you are building a RECURSIVE (caching) DNS server, you need to enable  recursion.  - If your recursive DNS server has a public IP address, you MUST enable access  control to limit queries to your legitimate users. Failing to do so will  cause your server to become part of large scale DNS amplification  attacks. Implementing BCP38 within your network would greatly  reduce such attack surface  \*/  recursion yes;  dnssec-enable yes;  dnssec-validation yes;  managed-keys-directory "/var/named/dynamic";  pid-file "/run/named/named.pid";  session-keyfile "/run/named/session.key";  /\* https://fedoraproject.org/wiki/Changes/CryptoPolicy \*/  include "/etc/crypto-policies/back-ends/bind.config";  };  logging {  channel default\_debug {  file "data/named.run";  severity dynamic;  };  };  zone "." IN {  type hint;  file "named.ca";  };  zone "ocp4-dev.pd.in" IN {  type master;  file "forward.zone";  allow-update { none; };  };  zone "0.168.192.in-addr.arpa" IN {  type master;  file "reverse.zone";  allow-update { none; };  };  include "/etc/named.rfc1912.zones";  include "/etc/named.root.key"; |

|  |
| --- |
| **[root@ocp4-dev ~]#** cat /var/named/forward.zone  $TTL 1W  @ IN SOA bastion.ocp4-dev.pd.in. root (  2019070700 ; serial  3H ; refresh (3 hours)  30M ; retry (30 minutes)  2W ; expiry (2 weeks)  1W ) ; minimum (1 week)  IN NS bastion.ocp4-dev.pd.in.  bastion.ocp4-dev.pd.in. IN A 192.168.0.51  api.ocp4-dev.pd.in. IN A 192.168.0.51  api-int.ocp4-dev.pd.in. IN A 192.168.0.51  \*.apps.ocp4-dev.pd.in. IN A 192.168.0.51  bootstrap.ocp4-dev.pd.in. IN A 192.168.0.52  master0.ocp4-dev.pd.in. IN A 192.168.0.53  master1.ocp4-dev.pd.in. IN A 192.168.0.54  master2.ocp4-dev.pd.in. IN A 192.168.0.55  worker0.ocp4-dev.pd.in. IN A 192.168.0.56  worker1.ocp4-dev.pd.in. IN A 192.168.0.57 |

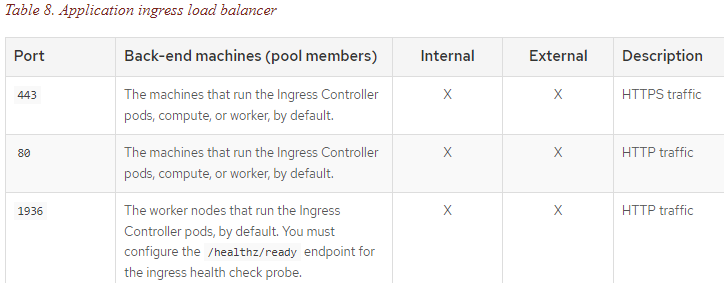
|  |
| --- |
| **[root@ocp4-dev ~]#** cat /var/named/reverse.zone  $TTL 1W  @ IN SOA bastion.ocp4-dev.pd.in. root (  2019070700 ; serial  3H ; refresh (3 hours)  30M ; retry (30 minutes)  2W ; expiry (2 weeks)  1W ) ; minimum (1 week)  IN NS bastion.ocp4-dev.pd.in.  51.0.168.192.in-addr.arpa. IN PTR bastion.ocp4-dev.pd.in.  51.0.168.192.in-addr.arpa. IN PTR api.ocp4-dev.pd.in.  51.0.168.192.in-addr.arpa. IN PTR api-int.ocp4-dev.pd.in.  52.0.168.192.in-addr.arpa. IN PTR bootstrap.ocp4-dev.pd.in.  53.0.168.192.in-addr.arpa. IN PTR master0.ocp4-dev.pd.in.  54.0.168.192.in-addr.arpa. IN PTR master1.ocp4-dev.pd.in.  55.0.168.192.in-addr.arpa. IN PTR master2.ocp4-dev.pd.in.  56.0.168.192.in-addr.arpa. IN PTR worker0.ocp4-dev.pd.in.  57.0.168.192.in-addr.arpa. IN PTR worker1.ocp4-dev.pd.in. |

**[root@bastion ~]#** chown named:named /var/named/forward.zone

**[root@bastion ~]#** chown named:named /var/named/reverse.zone

|  |
| --- |
| **[root@ocp4-dev ~]#** named-checkzone forward.zone /var/named/forward.zone  /var/named/forward.zone:9: ignoring out-of-zone data (bastion.ocp4-dev.pd.in)  /var/named/forward.zone:10: ignoring out-of-zone data (api.ocp4-dev.pd.in)  /var/named/forward.zone:11: ignoring out-of-zone data (api-int.ocp4-dev.pd.in)  /var/named/forward.zone:12: ignoring out-of-zone data (\*.apps.ocp4-dev.pd.in)  /var/named/forward.zone:14: ignoring out-of-zone data (bootstrap.ocp4-dev.pd.in)  /var/named/forward.zone:16: ignoring out-of-zone data (master0.ocp4-dev.pd.in)  /var/named/forward.zone:17: ignoring out-of-zone data (master1.ocp4-dev.pd.in)  /var/named/forward.zone:18: ignoring out-of-zone data (master2.ocp4-dev.pd.in)  /var/named/forward.zone:20: ignoring out-of-zone data (worker0.ocp4-dev.pd.in)  /var/named/forward.zone:21: ignoring out-of-zone data (worker1.ocp4-dev.pd.in)  zone forward.zone/IN: loaded serial 2019070700  OK |

### Load balancing requirements for user-provisioned infrastructure



|  |
| --- |
| **[root@ocp4-dev ~]#** cat /etc/haproxy/haproxy.cfg  #---------------------------------------------------------------------  # Example configuration for a possible web application. See the  # full configuration options online.  #  # https://www.haproxy.org/download/1.8/doc/configuration.txt  #  #---------------------------------------------------------------------  #---------------------------------------------------------------------  # Global settings  #---------------------------------------------------------------------  global  # to have these messages end up in /var/log/haproxy.log you will  # need to:  #  # 1) configure syslog to accept network log events. This is done  # by adding the '-r' option to the SYSLOGD\_OPTIONS in  # /etc/sysconfig/syslog  #  # 2) configure local2 events to go to the /var/log/haproxy.log  # file. A line like the following can be added to  # /etc/sysconfig/syslog  #  # local2.\* /var/log/haproxy.log  #  log 127.0.0.1 local2  chroot /var/lib/haproxy  pidfile /var/run/haproxy.pid  maxconn 4000  user haproxy  group haproxy  daemon  # turn on stats unix socket  stats socket /var/lib/haproxy/stats  # utilize system-wide crypto-policies  ssl-default-bind-ciphers PROFILE=SYSTEM  ssl-default-server-ciphers PROFILE=SYSTEM  #---------------------------------------------------------------------  # common defaults that all the 'listen' and 'backend' sections will  # use if not designated in their block  #---------------------------------------------------------------------  defaults  mode http  log global  option httplog  option dontlognull  option http-server-close  option forwardfor except 127.0.0.0/8  option redispatch  retries 3  timeout http-request 10s  timeout queue 1m  timeout connect 10s  timeout client 1m  timeout server 1m  timeout http-keep-alive 10s  timeout check 10s  maxconn 3000  frontend stats  bind \*:1936  mode http  log global  maxconn 10  stats enable  stats hide-version  stats refresh 30s  stats show-node  stats show-desc Stats for ocp4 cluster  stats auth admin:ocp4-dev  stats uri /stats  listen api-server-6443  bind \*:6443  mode tcp  server bootstrap bootstrap.ocp4-dev.pd.in:6443 check inter 1s backup  server master0 master0.ocp4-dev.pd.in:6443 check inter 1s  server master1 master1.ocp4-dev.pd.in:6443 check inter 1s  server master2 master2.ocp4-dev.pd.in:6443 check inter 1s  listen machine-config-server-22623  bind \*:22623  mode tcp  server bootstrap bootstrap.ocp4-dev.pd.in:22623 check inter 1s backup  server master0 master0.ocp4-dev.pd.in:22623 check inter 1s  server master1 master1.ocp4-dev.pd.in:22623 check inter 1s  server master2 master2.ocp4-dev.pd.in:22623 check inter 1s  listen ingress-router-443  bind \*:443  mode tcp  balance source  server worker0 worker0.ocp4-dev.pd.in:443 check inter 1s  server worker1 worker1.ocp4-dev.pd.in:443 check inter 1s  listen ingress-router-80  bind \*:80  mode tcp  balance source  server worker0 worker0.ocp4-dev.pd.in:80 check inter 1s  server worker1 worker1.ocp4-dev.pd.in:80 check inter 1s |

## Validating DNS resolution for user-provisioned infrastructure

|  |
| --- |
| **[root@ocp4-dev ~]#** dig +noall +answer @192.168.0.51 api.ocp4-dev.pd.in  api.ocp4-dev.pd.in. 604800 IN A 192.168.0.51  **[root@ocp4-dev ~]#** dig +noall +answer @192.168.0.51 api-int.ocp4-dev.pd.in  api-int.ocp4-dev.pd.in. 604800 IN A 192.168.0.51  **[root@ocp4-dev ~]#** dig +noall +answer @192.168.0.51 random.apps.ocp4-dev.pd.in  random.apps.ocp4-dev.pd.in. 604800 IN A 192.168.0.51  **[root@ocp4-dev ~]#** dig +noall +answer @192.168.0.51 console-openshift-console.apps.ocp4-dev.pd.in  console-openshift-console.apps.ocp4-dev.pd.in. 604800 IN A 192.168.0.51  **[root@ocp4-dev ~]#** dig +noall +answer @192.168.0.51 bootstrap.ocp4-dev.pd.in  bootstrap.ocp4-dev.pd.in. 604800 IN A 192.168.0.52  run reverse DNS lookups against the IP addresses  **[root@ocp4-dev ~]#** dig +noall +answer @192.168.0.51 -x 192.168.0.51  51.0.168.192.in-addr.arpa. 604800 IN PTR bastion.ocp4-dev.pd.in.  51.0.168.192.in-addr.arpa. 604800 IN PTR api.ocp4-dev.pd.in.  51.0.168.192.in-addr.arpa. 604800 IN PTR api-int.ocp4-dev.pd.in.  **[root@ocp4-dev ~]#**  dig +noall +answer @192.168.0.51 -x 192.168.0.52  52.0.168.192.in-addr.arpa. 604800 IN PTR bootstrap.ocp4-dev.pd.in. |

## Generating a key pair for cluster node SSH access

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

|  |
| --- |
| **[root@ocp4-dev ~]#**  ssh-keygen -t ed25519 -N '' -f ~/.ssh/id\_ed25519  Generating public/private ed25519 key pair.  Created directory '/root/.ssh'.  Your identification has been saved in /root/.ssh/id\_ed25519.  Your public key has been saved in /root/.ssh/id\_ed25519.pub.  The key fingerprint is:  SHA256:CbIyOvRHxq7kQryMbvS2GuMaQFtlQ/Yvt18xM7ipUbs root@bastion.ocp4-dev.pd.in  The key's randomart image is:  +--[ED25519 256]--+  | .\* |  | + o |  | . .. .. |  |. o .o ... . |  |ooo .+ .So o = |  |o=.o+ o o + = |  |Oooo o o + . |  |oB=oo + o |  |=+++. . E |  +----[SHA256]-----+  **[root@ocp4-dev ~]#**  cd .ssh  **[root@ocp4-dev ~]#**  ls  id\_ed25519 id\_ed25519.pub  **[root@ocp4-dev ~]#**  cat id\_ed25519.pub  ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGOW9GaCi+sjY34U5ed5tj+uJIFGrzk5cLRV7GD13Pcn [root@bastion.ocp4-dev.pd.in](mailto:root@bastion.ocp4-dev.pd.in)  **[root@ocp4-dev ~]#**  eval "$(ssh-agent -s)"  Agent pid 20993  **[root@ocp4-dev ~]#**  ssh-add ~/.ssh/id\_ed25519  Identity added: /root/.ssh/id\_ed25519 ([root@bastion.ocp4-dev.pd.in](mailto:root@bastion.ocp4-dev.pd.in))  OR  We can generate the sshkey normally we can us the .pub key |

## Obtaining the installation program

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

You have a computer that runs Linux with 500 MB of local disk space.

|  |
| --- |
| **Download and extract OpenShift Installer**  <https://mirror.openshift.com/pub/openshift-v4/clients/ocp/4.12.0/>  Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command  **[root@ocp4-dev ~]#** tar -xvf oc-4.12.0-linux.tar.gz  README.md  kubectl  oc  **Installing the OpenShift CLI on Linux**  You can install the OpenShift CLI (oc) binary on Linux  <https://access.redhat.com/downloads/content/290/ver=4.12/rhel---8/4.12.0/x86_64/product-software>  **[root@ocp4-dev ~]#** tar -xvf openshift-install-linux.tar.gz  README.md  openshift-install   * We need to move the oc and kubelet command to /usr/local/bin/ directory to execute the oc commands   **We need to download the pull secret from the link**  <https://console.redhat.com/openshift/install/pull-secret>  Download your installation [pull secret from the Red Hat OpenShift Cluster Manager](https://console.redhat.com/openshift/install/pull-secret). This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components |

## Manually creating the installation configuration file

We need to download the pull secret

You have an SSH public key on your local machine to provide to the installation program. The key will be used for SSH authentication onto your cluster nodes for debugging and disaster recovery.

You have obtained the OpenShift Container Platform installation program and the pull secret for your cluster.

|  |
| --- |
| Create an installation directory to store your required installation assets in:  **[root@ocp4-dev ~]#** mkdir openshift4.8  **install-config.yaml** file template that is provided and save it in the the created directory  **install-config.yaml file for bare meta**  **[root@ocp4-dev ~]#** cat /tmp/install-config.yaml  apiVersion: v1  baseDomain: pd.in  compute:  - hyperthreading: Enabled  name: worker  replicas: 0  controlPlane:  hyperthreading: Enabled  name: master  replicas: 3  metadata:  name: ocp4-dev  networking:  clusterNetwork:  - cidr: 10.128.0.0/14  hostPrefix: 23  networkType: OpenShiftSDN  serviceNetwork:  - 172.30.0.0/16  platform:  none: {}  fips: false  pullSecret: '{"auths": ...}'  sshKey: 'ssh-ed25519 AAAA...' |

## Creating the Kubernetes manifest

**[root@ocp4-dev ~]#** ./openshift-install create manifests --dir openshift4.8/

INFO Consuming Install Config from target directory

WARNING Making control-plane schedulable by setting MastersSchedulable to true f or Scheduler cluster settings

INFO Manifests created in: openshift4.8/manifests and openshift4.8/openshift

**[root@ocp4-dev openshift4.8]#** ls

manifests openshift

**[root@ocp4-dev openshift4.8]# cd** manifests

Inside the manifests directory there is file called cluster scheduler open the file cluster scheduler will be in the **true** state make it **false**

## Creating Ignition config files

**[root@ocp4-dev ~]#** ./openshift-install create ignition-configs --dir openshift4.8/

Openshift4.8 /

INFO Consuming Openshift Manifests from target directory

INFO Consuming OpenShift Install (Manifests) from target directory

INFO Consuming Worker Machines from target directory

INFO Consuming Common Manifests from target directory

INFO Consuming Master Machines from target directory

INFO Ignition-Configs created in: openshift4.8 and openshift4.8/auth

**[root@ocp4-dev ~]#** tree openshift4.8/

openshift4.8/

├── auth

│   ├── kubeadmin-password

│   └── kubeconfig

├── bootstrap.ign

├── master.ign

├── metadata.json

└── worker.ign

**[root@ocp4-dev openshift4.8]#** ls -ltr

total 288

drwxr-x--- 2 root root 50 Apr 20 11:32 auth

-rw-r----- 1 root root 1716 Apr 20 11:32 master.ign

-rw-r----- 1 root root 1716 Apr 20 11:32 worker.ign

-rw-r----- 1 root root 281508 Apr 20 11:32 bootstrap.ign

-rw-r----- 1 root root 104 Apr 20 11:32 metadata.json

**Install httpd**

**[root@ocp4-dev ~]#** yum install httpd –y

**[root@ocp4-dev ~]#** vim /etc/httpd/conf/httpd.conf

Change the port number because haproxy and httpd are running on the same port 80

In this case we need to change the port for httpd service

**[root@ocp4-dev ~]#** systemctl enable httpd.service

**[root@ocp4-dev ~]#** systemctl status httpd.service

**[root@ocp4-dev ~]#** systemctl restart httpd.service

**Copy the ignition files to web server:**

[root@ocp4-dev ~]# cd /var/www/html

[root@ocp4-dev html]# ls -ltr

total 288

drwxrwxrwx 2 root root 50 Apr 20 11:36 auth

-rwxrwxrwx 1 root root 104 Apr 20 11:36 metadata.json

-rwxrwxrwx 1 root root 1716 Apr 20 11:36 master.ign

-rwxrwxrwx 1 root root 281508 Apr 20 11:36 bootstrap.ign

-rwxrwxrwx 1 root root 1716 Apr 20 11:36 worker.ign

**[root@ocp4-dev ~]#** cp \*.ign /var/www/html/

## Installing RHCOS and starting the OpenShift Container Platform bootstrap process

To install OpenShift Container Platform on bare metal infrastructure that you provision, you must install Red Hat Enterprise Linux CoreOS (RHCOS) on the machines. When you install RHCOS, you must provide the Ignition config file that was generated by the OpenShift Container Platform installation program for the type of machine you are installing. If you have configured suitable networking, DNS, and load balancing infrastructure, the OpenShift Container Platform bootstrap process begins automatically after the RHCOS machines have rebooted.

Deploying Bootstrap, masters & workers node

Download the RHCOS live iso image and copy it to Vmware Exiserver datastore to create bootstrap, master and worker

Create a vm for bootstrap by providing the recommended compute resources and boot from the live iso.

Run “sudo nmtui” and configure the network with Static IP.

Run the coreos install command to load the bootstrap ignition config on the bootstrap node

**[root@ocp4-dev ~]#** curl http://192.168.0.51:82/bootstrap.ign

{"ignition":{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGOW9GaCi+sjY34U5ed5tj+uJIFGrzk5cLRV7GD13Pcn root@bastion.ocp4-dev.pd.in","ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC7z4QjMeMKGr1LGYdGeuPuqq4P6fWg9ey9pN0/HSxG+m4qalSkkEBodTWZ6CJRCVPMbAytuQHuzANs0f73iguMSUENr+j69ROO72RydolOr6ZEeWiBg9x89nP5cR8d+Mk0NA1kI6nIdmkIzQb8iE0HyVgLt80swJHdUdZ7rqEbqbpKY4x6TrM7MKqwnlSbEKFl4qZNbw0StebPD5YCyfaDRfcpVPrvXpCvhTZb7i1cluvODw6oUwavBcVIny3Hg2KP+XtCW0hTnl8il5VfdGs6judbP4Ts4hkeDTL/qw/tT3GlZtBbw0r30mpq6zu3jSzrFK4SFedAju40i4gm9BPF\n"]}]},"storage":{"files":[{"overwrite":true,"path":"/etc/containers/registries.conf","user":{"name":"root"},"contents":{"source":"data:text/plain;charset=utf-8;base64,"},"mode":384},{"overwrite":true,"path":"/etc/ignition-machine-config-encapsulated.json","user":{"name":"root"},"contents":{"source":"data:text/plain;charset=utf-8;base64,ewogICJtZXRhZGF0YSI6IHsKICAgICJuYW1lIjogImJvb3RzdHJhcC1maXBzIgogIH0sCiAgInNwZWMiOiB7CiAgICAiY29uZmlnIjogewogICAgICAiaWduaXRpb24iOiB7CiAgICAgICAgInZlcnNpb24iOiAiMy4xLjAiCiAgICAgIH0KICAgIH0sCiAgICAia2VybmVsQXJndW1lbnRzIjogW10sCiAgICAiZmlwcyI6IGZhbHNlCiAgfQp9Cg=="},"mode":384},

Create the Bootstrap Vm

We need to set the network

|  |
| --- |
| For bootstrap nodes:  **[core@bootstrap ~]$** sudo coreos-installer install /dev/sda --ignition-url=<http://192.168.0.51:82/bootstrap.ign> --insecure-ignition --copy-network |

|  |
| --- |
| **[root@bootstrap ~]#** crictl ps  CONTAINER IMAGE CREATED STATE NAME ATTEMPT POD ID  f65d53bf2bc3a quay.io/openshift-release-dev/ocp-release@sha256:31c7741fc7bb73ff752ba43f5acf014b8fadd69196fc522241302de918066cb1 25 seconds ago Running cluster-version-operator 0 2155c73dffbc1  058d4663687bc 2713b51902e9971d2c02112419d8cc7e0c83e3b28a96b9932075b26500d4a81b 27 seconds ago Running cloud-credential-operator 0 33f9d1b85e458  4db41edf527f0 32123307c03238150f230d4e895a31582f9c1c4ef4cc2623372fb1cb3fd17072 About a minute ago Running machine-config-server 0 4026cecd65418  c6d42edddbeb7 7f193ce191e414cf78fcdb0d7a163aaf079049a02aa800f8305b255f687c7547 2 minutes ago Running etcd 0 e4763a3ae2ac2  ed6a18b1648fc quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:2d1371d52c5233f6daf04aa0b0c12f29799155c15b49031bd9581d78529742b2 2 minutes ago Running etcdctl 0 e4763a3ae2ac2 |

Create the 3 Master Node Vms

|  |
| --- |
| **[core@master0 ~]$** sudo coreos-installer install /dev/sda --ignition-url=<http://192.168.0.51:82/master.ign> --insecure-ignition --copy-network |

|  |
| --- |
| **[core@master1 ~]$** sudo coreos-installer install /dev/sda --ignition-url=<http://192.168.0.51:82/master.ign> --insecure-ignition --copy-network |

|  |
| --- |
| **[core@master2 ~]$** sudo coreos-installer install /dev/sda --ignition-url=<http://192.168.0.51:82/master.ign> --insecure-ignition --copy-network |

Create the 2 Worker Node Vms

|  |
| --- |
| **[core@worker0 ~]$** sudo coreos-installer install /dev/sda --ignition-url=<http://192.168.0.51:82/worker.ign> --insecure-ignition --copy-network |

|  |
| --- |
| **[core@worker1 ~]$** sudo coreos-installer install /dev/sda --ignition-url=<http://192.168.0.51:82/worker.ign> --insecure-ignition --copy-network |

## Waiting for the bootstrap process to complete

The OpenShift Container Platform bootstrap process begins after the cluster nodes first boot into the persistent RHCOS environment that has been installed to disk. The configuration information provided through the Ignition config files is used to initialize the bootstrap process and install OpenShift Container Platform on the machines. You must wait for the bootstrap process to complete

|  |
| --- |
| **[root@ocp4-dev ~]#** ./openshift-install --dir <installation\_directory> wait-for bootstrap-complete --log-level=info  Output  INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...  INFO API v1.25.0 up  INFO Waiting up to 30m0s for bootstrapping to complete...  INFO It is now safe to remove the bootstrap resources |

The command succeeds when the Kubernetes API server signals that it has been bootstrapped on the control plane machines.

After the bootstrap process is complete, remove the bootstrap machine from the load balancer.

## Logging in to the cluster by using the CLI

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

Export the kubeadmin credentials:

|  |
| --- |
| **[root@ocp4-dev ~]#** export KUBECONFIG=<installation\_directory>/auth/kubeconfig |

## 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.

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:

|  |
| --- |
| **[root@ocp4-dev ~]#** oc get nodes  **[root@ocp4-dev ~]#** oc get csr |

|  |
| --- |
| To approve all pending CSRs, run the following command:  **[root@ocp4-dev ~]#** 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 |

Some Operators might not become available until some CSRs are approved.

|  |
| --- |
| **[root@ocp4-dev ~]#** $ oc get csr  To approve them individually, run the following command for each valid CSR:  **[root@ocp4-dev ~]#** oc adm certificate approve <csr\_name> |

After all client and server CSRs have been approved, the machines have the Ready status. Verify this by running the following command:

|  |
| --- |
| **[root@ocp4-dev ~]#** oc get nodes  NAME STATUS ROLES AGE VERSION  master0.ocp4-dev.pd.in Ready control-plane,master 28h v1.25.4+77bec7a  master1.ocp4-dev.pd.in Ready control-plane,master 28h v1.25.4+77bec7a  master2.ocp4-dev.pd.in Ready control-plane,master 26h v1.25.4+77bec7a  worker0.ocp4-dev.pd.in Ready worker 26h v1.25.4+77bec7a  worker1.ocp4-dev.pd.in Ready worker 25h v1.25.4+77bec7a |

Initial Operator configuration

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

|  |
| --- |
| Watch the cluster components come online:  **[root@ocp4-dev ~]#** watch -n5 oc get clusteroperators  NAME VERSION AVAILABLE PROGRESSING DEGRADED SINCE MESSAGE  authentication 4.12.0 True False False 26h  baremetal 4.12.0 True False False 28h  cloud-controller-manager 4.12.0 True False False 28h  cloud-credential 4.12.0 True False False 29h  cluster-autoscaler 4.12.0 True False False 28h  config-operator 4.12.0 True False False 28h  console 4.12.0 True False False 26h  control-plane-machine-set 4.12.0 True False False 28h  csi-snapshot-controller 4.12.0 True False False 28h  dns 4.12.0 True False False 28h  etcd 4.12.0 True False False 28h  image-registry 4.12.0 True False False 28h  ingress 4.12.0 True False False 26h  insights 4.12.0 True False False 15s  kube-apiserver 4.12.0 True False False 28h  kube-controller-manager 4.12.0 True False False 28h  kube-scheduler 4.12.0 True False False 28h  kube-storage-version-migrator 4.12.0 True False False 28h  machine-api 4.12.0 True False False 28h  machine-approver 4.12.0 True False False 28h  machine-config 4.12.0 True False False 28h  marketplace 4.12.0 True False False 28h  monitoring 4.12.0 True False False 25h  network 4.12.0 True False False 28h  node-tuning 4.12.0 True False False 28h  openshift-apiserver 4.12.0 True False False 28h  openshift-controller-manager 4.12.0 True False False 28h  openshift-samples 4.12.0 True False False 28h  operator-lifecycle-manager 4.12.0 True False False 28h  operator-lifecycle-manager-catalog 4.12.0 True False False 28h  operator-lifecycle-manager-packageserver 4.12.0 True False False 28h  service-ca 4.12.0 True False False 28h  storage 4.12.0 True False False 28h |

|  |
| --- |
| Completing installation on user-provisioned infrastructure After you complete the Operator configuration, you can finish installing the cluster on infrastructure that you provide.  **[root@ocp4-dev ~]#** ./openshift-install --dir openshift4.8 wait-for install-complete  Output  INFO Waiting up to 40m0s (until 4:45PM) for the cluster at https://api.ocp4-dev.pd.in:6443 to initialize...  INFO Checking to see if there is a route at openshift-console/console...  INFO Install complete!  INFO To access the cluster as the system:admin user when using 'oc', run 'export KUBECONFIG=/root/openshift4.8/auth/kubeconfig'  INFO Access the OpenShift web-console here: https://console-openshift-console.apps.ocp4-dev.pd.in  INFO Login to the console with user: "kubeadmin", and password: "##################"  INFO Time elapsed: 0s |

The command succeeds when the Cluster Version Operator finishes deploying the OpenShift Container Platform cluster from Kubernetes API server.

Confirm that the Kubernetes API server is communicating with the pods:

