DSTG

OpenShift Container Platform

Detailed Design

# Preface

## Confidentiality, Copyright, and Disclaimer

This is a Customer-facing document between Red Hat, Inc. and the Defence Science and Technology Group (DSTG).

Copyright 2023© Red Hat, Inc. All Rights Reserved. No part of the work covered by the copyright herein may be reproduced or used in any form or by any means – graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems without permission in writing from Red Hat except as is required to share this information as provided with the aforementioned confidential parties.

This document is not a quote and does not include any binding commitments by Red Hat. If acceptable, a formal quote can be issued upon request, which will include the scope of work, cost, and any customer requirements as necessary.

## Trademarks

Trademarked names may appear throughout this document. Rather than list the names and entities that own the trademarks or insert a trademark symbol with each mention of the trademarked name, the names are used only for editorial purposes and to the benefit of the trademark owner with no intention of infringing upon that trademark.

## Audience

This document is intended for Client technical staff responsible for the environment.

## Additional Background and Related Documents

This document does not contain step by step details of installation or other tasks, as they are covered in the relevant documentation on <http://access.redhat.com/>. Links to the appropriate documents will be made when required.

## Scripts and Playbooks

Any scripts provided are being provided as-is, without any form of support or warranty. All provided scripts can be modified by the customer at will.

# Review History

| **Version** | **Date** | **Contributor** | **Role** | **Description** |
| --- | --- | --- | --- | --- |
| 0.1 |  | [Craig Scott](mailto:craig.scott@redhat.com) | Senior Consultant | First draft |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Table of Contents

[**Preface 2**](#_3rdcrjn)

[Confidentiality, Copyright, and Disclaimer 2](#_26in1rg)

[Trademarks 2](#_lnxbz9)

[Audience 2](#_35nkun2)

[Additional Background and Related Documents 2](#_1ksv4uv)

[Scripts and Playbooks 3](#_44sinio)

[**Review History 3**](#_2jxsxqh)

[**Introduction 7**](#_1y810tw)

[Purpose 7](#_4i7ojhp)

[Scope 7](#_2xcytpi)

[Staffing 7](#_mwidzinf2yao)

[1.4. Terms and Acronyms 8](#_2bn6wsx)

[**High Level Overview 10**](#_3as4poj)

[Objective 10](#_1pxezwc)

[Requirements 10](#_49x2ik5)

[Solution Overview 12](#_147n2zr)

[**Component Overview 13**](#_ihv636)

[Server hardware 13](#_ewmym67h2zhb)

[Installation machine 13](#_90otqn6dd1ec)

[Networking 14](#_v9vxak2t5yog)

[Operational VLAN 14](#_e72lyua0t35c)

[Software Defined Network 15](#_5bz8go50t4cf)

[iDRAC VLAN 15](#_j9342ctrxcf4)

[iSCSI VLAN 16](#_t1xpguv93knv)

[DNS 17](#_k082sb4x75hf)

[Certificates 17](#_epym5yz23ng6)

[Container registry 18](#_yp8dc8664if8)

[Secrets management 18](#_aidkddefuxvw)

[Storage 18](#_esqqb9ur9ios)

[Logging 18](#_9vyn7ws555cl)

[Source code management 18](#_tsbiqwtvxrr)

[**Cluster Installation 19**](#_4f1mdlm)

[Node configuration 19](#_kufmdfmlmf9y)

[Control plane nodes 19](#_2pta16n)

[Compute nodes 19](#_14ykbeg)

[Content mirroring 19](#_cmi5h7wevng5)

[Mirroring images 20](#_la9plv5suuuy)

[Source registry authentication 21](#_mu1rrgsc96ab)

[Destination registry authentication 21](#_cf6aiepmpv7e)

[Configuration 22](#_jd2djxagmqth)

[OpenShift platform images 22](#_f89wspbil5m)

[Operator catalogues 22](#_lmsfcm2ig7ze)

[Mirroring 24](#_ro1frmsmn4gx)

[Cluster Deployment 25](#_3l18frh)

[Deployment steps 27](#_kayzoifs1w4y)

[Monitoring the installation 28](#_wcd275x7m3ts)

[Install machine 28](#_gtl3lgvikjpf)

[Rendezvous machine console 29](#_r88qzn1tf76n)

[Rendezvous machine CLI 29](#_8ab6kv33vx91)

[Kubernetes API 29](#_4qym0ibt339w)

[Post deployment configuration 30](#_xvoj95k1gn8u)

[Bootstrap OpenShift GitOps 31](#_1egqt2p)

[General Config 33](#_h5g7g5nlezho)

[Operator catalogues 33](#_d6pd5ghcwhep)

[DSTG CA 33](#_oyclu3hoyxat)

[OpenShift ingress certificate 34](#_bw3v9yy12ij6)

[Openshift API Certificate 34](#_3h8fh01n3f39)

[IPSec encryption 34](#_lq1gczuyibqo)

[Authentication 35](#_urmn2qdfem5n)

[LDAP group sync 35](#_4e8787q27d14)

[Network Policies 35](#_hcx0qmsy7f5f)

[OpenShift Logging 35](#_a69ddup48yxf)

[Dell CSM Operator 36](#_2q44mb2wvf6g)

[NVidia GPU support 36](#_efnwtwywhidv)

[Project configuration 36](#_2g2vr92zth9v)

[Network Policy 37](#_sn0w2o75e1ds)

[Quotas 37](#_jeddji2reskk)

[Management 38](#_1vsw3ci)

[Cluster Monitoring 38](#_69qjikkn2v8y)

[Monitoring user workloads 38](#_ufby46akeou)

[Patching 39](#_gtrgavew1akx)

[Platform 39](#_fkw39k9du68h)

[Monitoring the upgrade 40](#_r63j6o9m9ev6)

[Operators 43](#_73y1c93vnucp)

[Service account management 44](#_v1i8ol29q4ss)

[Certificate Management 45](#_p2y5dvc1kxi6)

[Ingress key and certificate 45](#_64a15z9hvbqy)

[API key and certificate 45](#_ou8x3mxc961d)

[Root CA certificate 45](#_v4z9pm32q7yq)

[Power off or reboot a node 46](#_qzr6oouskffr)

[Cluster shutdown and startup 47](#_g0conov9x1k2)

[Adding nodes 47](#_cs2a49t87u2b)

[**3. Security Architecture 48**](#_3tbugp1)

[Network Connectivity 48](#_j8sehv)

[Encryption 49](#_3ygebqi)

[At Rest 49](#_5r606yh39jbu)

[In Transit 49](#_mbha8u6xdlce)

[Authentication 49](#_2dlolyb)

[RBAC 50](#_sqyw64)

[Logging 51](#_2r0uhxc)

[Secrets 51](#_3q5sasy)

[**Appendix A: Mirror configuration 51**](#_itavugrtwqfh)

[**Appendix B: Installation configuration 52**](#_wjgur7oqx2hz)

[install-config.yaml 52](#_alvg7zogdtw1)

[agent-config.yaml 52](#_gle5ktu55qbz)

# Introduction

## Purpose

This document expands upon the previous high level design, and captures the lower level technical information required for the supported deployments of OpenShift Container Platform.

## Scope

This document aims to provide a detailed design to meet requirements identified in the HLD, that the DSTG will be able to deploy and support. This architecture is the result of the previous high level design, and design workshop and should cover all requirements that were expressed during this workshop. Anything not explicitly listed as in-scope as per the above is deemed to be out of the scope of this design document.

## Staffing

*DSTG Project Team*

| Name | Function | E-mail address |
| --- | --- | --- |
| Heath Cooks | Director ICT Support |  |
| Justin Baldock | Discipline Leader Intelligence Program | justin.baldock@defence.gov.au |
| Mark Whitburn | Facilities and network management | mark.whitburn@defence.gov.au |
| Graeme Boskell | Red Hat admin |  |
| Allen Copley | Windows admin | allen.copley1@defence.gov.au |
| Jeff Borkent | Sysadmin | jeff.borkent@defence.gov.au |
| Shane Bennett | Cyber Integration | shane.bennett@defence.gov.au |

*Red Hat Project Team*

| Name | Function | E-mail address |
| --- | --- | --- |
| Dale Brandon | Service Delivery Manager | dbrandon@redhat.com |
| Craig Scott | Senior Consultant | craig.scott@redhat.com |
| Mark Hahl | Senior Consultant | mark.hahl@redhat.com |
| Krishna Giri | Project Manager | kgiri@redhat.com |

## 1.4. Terms and Acronyms

The table below provides a glossary of the terms and acronyms used within this document.

| **Acronym** | **Definition** |  |
| --- | --- | --- |
| AD | Active Directory |  |
| AD CS | Active Directory Certificate Services |  |
| ADFS | Active Directory Federation Services |  |
| ACS | Red Hat Advanced Cluster Security for Kubernetes |  |
| API | Application Programming Interface |  |
| BLOB | Binary Large Object |  |
| CA | Certifying Authority |  |
| CI/CD | Continuous Integration/Continuous Deployment |  |
| CPU | Central Processing Unit |  |
| CRD | Custom Resource Definition |  |
| CRI-O | Container Runtime Interface - Open Container Initiative |  |
| CRL | Certificate Revocation List |  |
| CSI | Container Storage Interface |  |
| eBPF | extended Berkeley Packet Filter |  |
| HPA | Horizontal Pod Autoscalar |  |
| HTTP | Hypertext Transfer Protocol |  |
| HTTPS | Hypertext Transfer Protocol Secure |  |
| IaC | Infrastructure as Code |  |
| ISV | Independent Software Vendors |  |
| JSON | JavaScript Object Notation |  |
| LDAP | Lightweight Directory Access Protocol |  |
| MFA | Multi-Factor Authentication |  |
| OADP | OpenShift API for Data Protection |  |
| OCI | Open Container Initiative |  |
| OS | Operating System |  |
| OCP | OpenShift Container Platform |  |
| PaaS | Platform as a Service |  |
| PoC | Proof of Concept |  |
| PV | Persistent Volume |  |
| PVC | Persistent Volume Claim |  |
| QA | Quality Assurance |  |
| RBAC | Role Based Access Control |  |
| RHCOS | Red Hat CoreOS |  |
| RHEL | Red Hat Enterprise Linux |  |
| S3 | Amazon Simple Storage Service |  |
| SCCM | Source Code and Configuration Management |  |
| SIEM | Security Information and Event Management |  |
| SOAR | Security Orchestration Automated Response |  |
| SSO | Single Sign-On |  |
| SRE | Site Reliability Engineering |  |
| TLS | Transport Layer Security |  |
| UI | User Interface |  |
| VM | Virtual Machine |  |
| YAML | Yet Another Markup Language |  |

# High Level Overview

## Objective

The key objective of this document is to describe how the RedHat Openshift bare metal cluster was installed and configured to support the upcoming "hackathon" and other internal applications.

## Requirements

* 1. The following table captures the solution requirements and are cited as they guided the Design Decisions.
  2. The MoSCoW method of requirement classification is used to steer design decisions, where:
  3. **MUST** describes a fit-for-purpose requirement the team cannot modify or change.  
     **SHOULD** describes a fit-for-purpose requirement the team can modify or change.  
     **COULD** describes best effort requirements the solution will meet if possible, but will still be regarded as fit for purpose if not met; and  
     WON’T describes deliberate-omission requirements the solution will not attempt to meet.

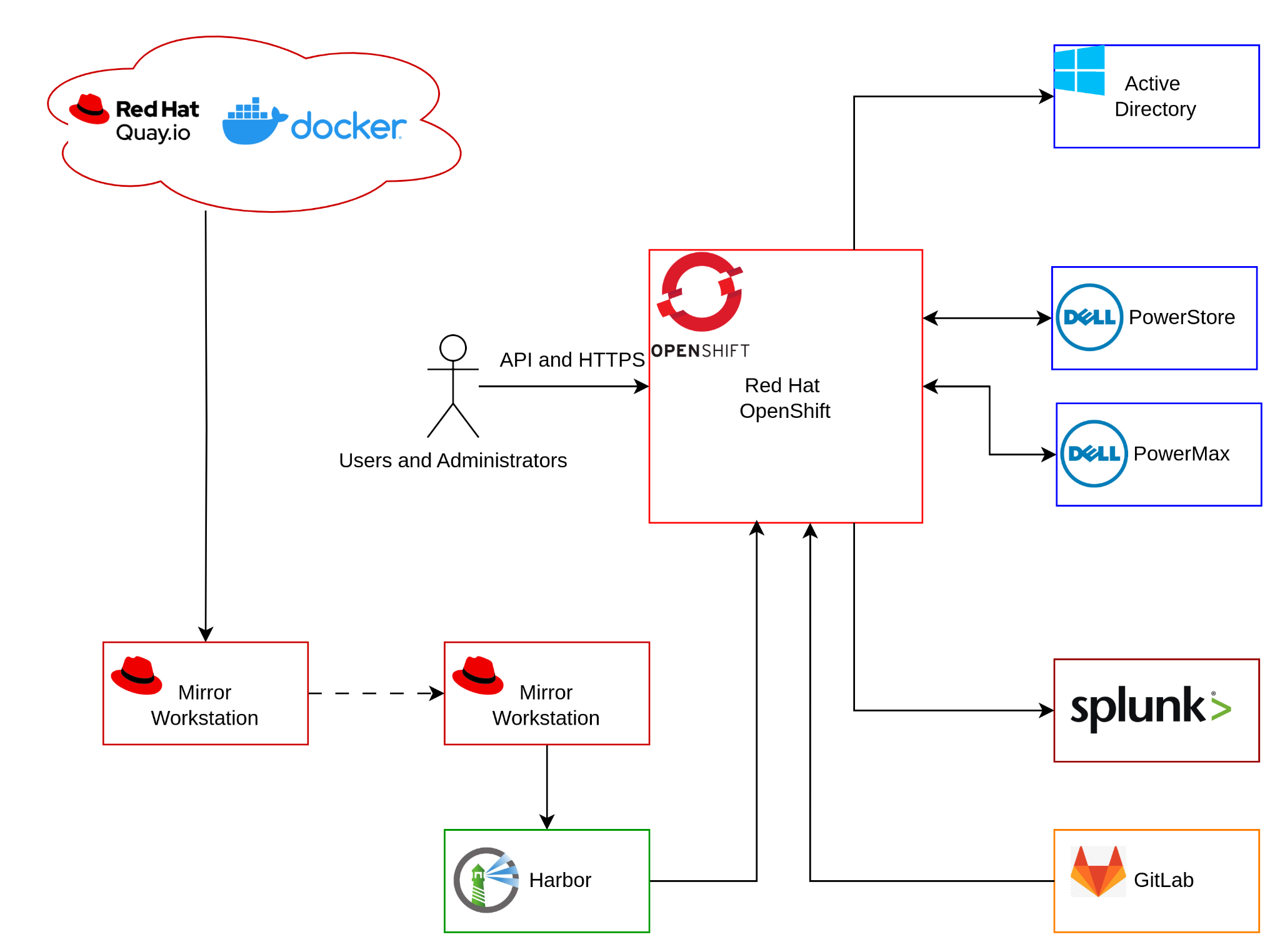
| **ID** | **Type** | **Category** | **Requirement Detail** |
| --- | --- | --- | --- |
| 1 | MUST | Vendor Support | The solution will be supported by Red Hat |
| 2 | SHOULD | General | Where multiple approaches are valid, the solution will favour the option offering the greater operational simplicity. |
| 3 | MUST | Encryption | The cluster installation must be FIPS compliant |
| 4 | SHOULD | Automation | Cluster configuration should be managed through Infrastructure as Code principles |
| 5 | SHOULD | Network Zoning | The solution will follow network zoning guidelines set by DSTG, including placement of specific solution components such as management consoles and databases in appropriate network zones. |
| 7 | SHOULD | High Availability | The solution will allow for further nodes or components to be added at a later date, allowing for high availability. |
| 8 | MUST | Secret Storage | The solution will make use of a sensitive information system fit for storing credentials (e.g. encrypted, masking in execution logs, etc) |
| 9 | MUST | Secret Storage | The solution will use the nominated tool for storage of sensitive information, such as passwords and other credentials |
| 10 | SHOULD | RBAC | The solution will offer isolation between objects used by different teams. |
| 11 | SHOULD | Network traffic encryption | The solution will use encrypted traffic between its components. |
| 12 | SHOULD | IAM | The solution will use the nominated directory applicable to the environment for authentication and authorisation services. |
| 13 | SHOULD | Source Code Management | The solution will use the nominated Source Code Management tool - DSTG enterprise GitLab repository. |
| 14 | SHOULD | Logging | The solution will log to DSTG centralised logging store |
| 15 | SHOULD | Certificates | The solution will use certificates compliant with the internal standard, requiring AES256 or higher, and signed by the nominated Certificate Authority. |
| 16 | SHOULD | Compliance | The solution will comply with Security guidelines for application control or seek an approved exemption. |
| 17 | SHOULD | Patching and Upgrades | The solution will comply with Security guidelines for patching . |
|  |  |  |  |
|  |  |  |  |

## 

## Solution Overview

There will be a single OpenShift instance deployed in the DSTG environment for the initial purpose of hosting a "hackathon" with overseas partners. The OpenShift cluster will be deployed to bare metal Dell servers in the on-premises data centre in RAAF Edinburgh.

The cluster will be installed on DSTG provided bare metal servers using the **Agent** install methodology where an ISO image is generated by the OpenShift installer and that ISO is used to install the CoreOS operating system and configure the machines as control plane and compute nodes. Storage for persistent volumes will be provisioned from the Dell Powerstore and Dell Powermax appliances.



*Figure 1: High level overview*

Once the OpenShift cluster has been deployed, application and operator images required to extend the platform and run the applications are pulled from the DSTG managed container registry.

The post-installation configuration of the clusters will be applied following infrastructure as code (IaC) principles utilising OpenShift Gitops to apply the configuration defined in one or more source code repositories in the DSTG GitLab instance. OpenShift GitOps applications also ensure that the configuration does not drift from that defined in the GitOps applications.

# Component Overview

## Server hardware

DSTG has supplied 12 Dell PowerEdge MX760C Servers to run the OpenShift cluster. The specifications of the cluster are:

* 2 X 32 Core Intel XEON Platinum
* Network interface connectivity at the enclosure
* 64GB RAM
* 960GB Raid 1 NVMe disk
* Trusted Platform Module v2

A Dell PowerStore 3200T and a Dell PowerMax appliance have also been supplied to provide storage to the platform.

Full details of the hardware can be found in the bill of materials provided by Dell.

## Installation machine

A Linux machine, typically Red Hat Enterprise Linux 9 or higher, is required to act as the install host to generate the installation ISO and to monitor the installation process.

The install host needs the configurations and software described in the following table to be installed

| **Component** | **Installed from** | **Notes** |
| --- | --- | --- |
| FIPS mode enabled | n/a | Can be enabled either at install time or following installation. Refer to the RHEL documentation for details. |
| Network Manager API: nmstate | CDN |  |
| OpenShift client | https://mirror.openshift.com |  |
| OpenShift installer (FIPS) | https://mirror.openshift.com | The OppenShift installer for FIPS is bundled in a file of the name: openshift-install-rhel9-amd64.tar.gz |

*Table 1: Install host components*

A specific installer is required to install the OpenShift cluster in FIPS mode This ensures that the RHCOS is configured to generate and use all cryptographic keys only with FIPS-approved algorithms. This installer is named openshift-install-fips.

The version of the installer dictates what version of OpenShift will be installed. To check the version of the installer, run the following after downloading and unpack the installer archive:

| $ ./openshift-install-fips version openshift-install 4.19.6 ... |
| --- |

This installer will install OpenShift 4.19.6.

**Note:** In a disconnected environment, you must ensure you have mirrored the corresponding version of the OpenShift platform containers into the local mirror.

The version of the OpenShift client should also be kept up to date as much as practicable, but there is no direct correlation required between the OpenShift cluster and client versions.

## Networking

**NOTE:** Due to issues with the configuration of the networking stack, the VLAN tagging has been applied at the NIC level rather than at the switch level.

The VLANs virtual NICs are created by the OpenShift configuration and therefore do not come up until the OpenShift cluster is starting.

This could be reverted if the networking can be reconfigured.

### Operational VLAN

One of the NICs presented to each server is dedicated to the Operational network.

The servers are located in a DSTG domain with the FQDN: <Domain name>

The CIDR range for cluster compute and control plane nodes is: 0.0.0.0/0

The operational network VLAN 000 is allocated to these NICs.

IP address allocations are via DHCP, permanent reservations mapping the allocated address to the nodes are created following installation.

| **Purpose** | **Hostname** | **MAC address** | **IP Address** |
| --- | --- | --- | --- |
| Control plane |  |  |  |
| Control plane |  |  |  |
| Control plane |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |

*Table 2: Operational VLAN hostname and IP addresses*

The OpenShift nodes determine their hostname by reverse DNS lookup once they have been allocated their IP addresses at boot. As the DHCP MAC address mappings are reserved in DHCP following server build, the nodes will always get the same hostname.

### Software Defined Network

The OpenShift cluster is being installed with the default **OVNKubernetes** Software Defined network (SDN) plugin. The SDN provides a unique IP address for every pod and every service in the cluster. This IP address is internal to the cluster only, it is not routable outside of the cluster

The CIDR for the pod network is: 0.0.0.0/0

The CIDR for the service network is: 0.0.0.0/0

### iDRAC VLAN

The servers have a dedicated interface for the out of band management iDRAC connection..

The CIDR range for iDRAC interfaces on the Dell server is: 0.0.0.0/0

iDRAC IP address allocations are static and shown in the following table.

| **Purpose** | **Hostname** | **MAC address** | **IP Address** |
| --- | --- | --- | --- |
| Control plane |  |  |  |
| Control plane |  |  |  |
| Control plane |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |

*Table 3: iDRAC hostname and IP addresses*

### iSCSI VLAN

Connectivity to the Dell PowerStore is via another NIC presented to each of the servers.

The CIDR range for iSCSI interfaces on the Dell server is: 0.0.0.0/0.

The operational network VLAN 000 is allocated to these NICs.

iSCSI IP address allocations are static and shown in the following table.

| **Purpose** | **Hostname** | **MAC address** | **IP Address** |
| --- | --- | --- | --- |
| Control plane |  |  |  |
| Control plane |  |  |  |
| Control plane |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |
| Compute |  |  |  |

*Table 4: iSCSI hostname and IP addresses*

**Note:** another NIC and VLAN should be configured to allow multi-path access to the storage array

## DNS

DNS A records for each server host are allocated in the DNS server manually or via D the DHCP server. Additionally, 3 further DNS entries are required mapped to the API and APPs VIP:

| Purpose | DNS name | IP Address |
| --- | --- | --- |
| API VIP | api.<cluster>.<domain> |  |
| Internal API VIP | api-int.<cluster>.<domain> | As above |
| APPs VIP (wildcard) | \*.apps.<cluster>.<domain> |  |

*Table 5: DNS records*

## Certificates

Certificates are issued by DSTG Microsoft Certificate manager service. Two certificates are required and must use the include the *subjectAltName* extension.

The API endpoint needs a certificate for the FQDN and its corresponding private key. Each should be in a separate PEM format file. The private key must be unencrypted. The certificate must include the subjectAltName extension showing the FQDN:

| api.<cluster>.<domain> |
| --- |

The certificate file can contain one or more certificates in a chain. The certificate for the API server FQDN must be the first certificate in the file. It can then be followed with any intermediate certificates, and the file should end with the root CA certificate.

The APPs endpoint needs a wildcard certificate for the fully qualified .apps subdomain and its corresponding private key. Each should be in a separate PEM format file. The private key must be unencrypted. The certificate must include the *subjectAltName* extension showing \*.apps.<clustername>.<domain>

The certificate file can contain one or more certificates in a chain. The file must list the wildcard certificate as the first certificate, followed by other intermediate certificates, and then ending with the root CA certificate.

## Container registry

Container images to run both the OpenShift platform and the operators will be consumed from the internal container registry. The Harbor internal container registry will be populated from approved internet sources via a manual DSTG process.

A project named **openshift** will be created in the Harbor registry to manage the images required by the OpenShift cluster. Other images that are not required for the OpenShift platform operator installation should be managed in separate Harbor projects.

## Secrets management

GitLab project and global variables can be used to manage secrets such as service account passwords and certificates.

The External Secrets Operator (ESO) can be used to access secrets in GitLab variables and create corresponding secret objects in OpenShift.

## Storage

Storage for persistent volumes is provisioned from the Dell PowerStore 3200T appliance. The Dell PowerStore presents LUNs to the nodes over iSCSI. The Dell CSM driver provides the interface between the OpenShift cluster and the Dell PowerStore.

## Logging

Logs from OpenShift will ultimately be stored in Splunk. In the interim, logs will be stored in the central syslog server.

## Source code management

The local GitLab instance will be used to manage the manifest files that define the cluster installation and operator configuration.

# Cluster Installation

The cluster installation is initiated from the install host. Boot ISOs are created based on this host based on the configuration files that describe the cluster. The servers then booted from the agent ISOs and the cluster is built on the target machines.

## Node configuration

#### Control plane nodes

There will be 3 control plane nodes deployed in the cluster.

| **Hostname** | **Memory** | **CPU** | **Filesystem** |
| --- | --- | --- | --- |
|  | 64GB | 32 cores | 1TB |
|  | 64GB | 32 cores | 1TB |
|  | 64GB | 32 cores | 1TB |

*Table 6: Control plane nodes*

#### Compute nodes

There will be 9 compute nodes deployed in the management cluster, all designated as infrastructure nodes.

| **Hostname** | **Memory** | **CPU** | **Filesystem** |
| --- | --- | --- | --- |
|  | 64GB | 32 cores | 1TB |
|  | 64GB | 32 cores | 1TB |
|  | 64GB | 32 cores | 1TB |
|  | 64GB | 32 cores | 1TB |
|  | 64GB | 32 cores | 1TB |
|  | 64GB | 32 cores | 1TB |
|  | 64GB | 32 cores | 1TB |
|  | 64GB | 32 cores | 1TB |
|  | 64GB | 32 cores | 1TB |

*Table 7: Compute nodes*

Reference:  
<https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/installing_an_on-premise_cluster_with_the_agent-based_installer/index>

## Content mirroring

The simplest method to manage the mirroring process is to use the *oc-mirror* plugin to the OpenShift Client (oc) command.

The plugin keeps track of what has already been mirrored which means that it only needs to download new or updated packages.

The *oc-mirror* plugin needs either direct access, or access through the enterprise proxy server and firewalls to access the external Red Hat container registries and other container sources.

The following URIs need to be accessible.

| **URL** | **Ports** | **Function** |
| --- | --- | --- |
| [registry.redhat.io](http://registry.redhat.io/) | 443, 80 | Access to Core Container Images |
| [quay.io](http://quay.io/) |  | Access to Core Container Images |
| \*.[quay.io](http://quay.io/) | 443, 80 | Access to Core Container Images |
| [rhcos.mirror.openshift.com](http://rhcos.mirror.openshift.com/) | 443, 80 | Access to CoreOS Images for Cluster Hosts |
| [api.openshift.com](http://api.openshift.com/) | 443, 80 | Verifies Cluster Version and Updates |
| [registry.connect.redhat.com](http://registry.connect.redhat.com/) | 443, 80 | Partner Certified Operator Images |

*Table 8. Proxy URI allow list*

### Mirroring images

In a fully disconnected environment, container images must be mirrored into the DSTG’s on-prem container registry clusters to use as a source. This is a manual step as it requires transferring content from a connected environment where it can be downloaded into the disconnected environment..

The *oc-mirror* plugin to the OpenShift Client tool should be used to manage the content mirroring. This tool can also be used for other images from docker compatible repositories, and the images required by helm charts.

The following operators are mirrored from the Red Hat operators catalogue:

* Cert Manager
* Compliance Operator
* MetalLB
* NMState
* Node Feature Discovery Operator
* OADP
* OpenShift Gitops
* OpenShift Logging
* OpenShift Virtualization
* Service Mesh
* Web Terminal

The following operators are mirrored from the Red Hat certified operators catalogue:

* Dell Storage
* NVIDIA GPU Operator

The following operators are mirrored from the Red Hat community operators catalogue

* External Secrets
* Group sync operator

Other images that are mirrored are:

* OpenShift CLI container (ose-cli)
* OpenShift support tools
* OpenShift Universal Base Images (UBI)

The mirroring configuration file that is used in the mirroring process is saved in the DSTG GitLab repository containing the cluster configuration YAML and is also saved in [**Appendix A: Mirror configuration**](#_itavugrtwqfh).

#### Source registry authentication

Authentication is required for the source registries in order to download the content. Red Hat supplies a pre-filled authentication file known as a pull secret that contains the entitled account encoded credentials required for accessing the registries. The pull secret is accessible via the following URL (authentication required):

<https://console.redhat.com/openshift/install/pull-secret>

Download the pull secret and copy it to the required location and file name:

| $ cp pull-secret.txt ~/.docker/config.json |
| --- |

#### Destination registry authentication

Authentication is required for the destination registries in order to upload the content. This can either be a user account, service account or robot account but the account must have write permissions to the project within the repository used for OpenShift content

You can generate an auth file in the required location on the machine used to push the content using the *podman* or the *skopeo* command:

| $ podman login -authfile ~/.docker/config.json \ https://<registry-dns-name>:<port>   $ skopeo login -authfile ~/.docker/config.json \ https://<registry-dns-name>:<port> |
| --- |

#### Configuration

The oc mirror plugin uses configuration files to identify what content is being mirrored.

##### OpenShift platform images

Build the mirror configuration file on the internet connected host to identify the OpenShift platform image channel to download to disk. Specify a minimum version to download to avoid downloading images that are not going to be used.

To determine the available releases for a channel (eg: stable-4.16), run the following command:

| oc mirror list releases --channel=stable-4.19 |
| --- |

An example configuration file for the stable-4.19 channel mirroring only release 4.19.9 is:

| kind: ImageSetConfiguration  apiVersion: mirror.openshift.io/v2alpha1  archiveSize: 20  mirror:  platform:  channels:  - name: stable-4.19  minVersion: 4.19.9  maxVersion: 4.19.9 |
| --- |

Specify a minimum and maximum version to minimise the amount of data that needs to be transferred into the disconnected environment.

Note also the archive size parameter. This can be used to split the output files up into separate archive files of the specified size in GB.

##### Operator catalogues

The same configuration file can also specify which operators from the operator catalogues to mirror.

List all available catalogues for a version of OpenShift:

| $ oc mirror list operators --catalogs --version=4.19  Available OpenShift OperatorHub catalogs:  OpenShift 4.19  [registry.redhat.io/redhat/redhat-operator-index:v4.1](http://registry.redhat.io/redhat/redhat-operator-index:v4.16)9  [registry.redhat.io/redhat/certified-operator-index:v4.1](http://registry.redhat.io/redhat/certified-operator-index:v4.16)9  [registry.redhat.io/redhat/community-operator-index:v4.1](http://registry.redhat.io/redhat/community-operator-index:v4.16)9  [registry.redhat.io/redhat/redhat-marketplace-index:v4.1](http://registry.redhat.io/redhat/redhat-marketplace-index:v4.16)9 |
| --- |

List all available packages in a catalogue, for example the Red Hat Operator index:

| $ oc mirror list operators \  --catalog=registry.redhat.io/redhat/redhat-operator-index:v4.19    NAME DISPLAY NAME DEFAULT CHANNEL  …  cluster-logging stable-6.3  compliance-operator stable  … |
| --- |

Update the operators section of the configuration file to specify each repository you want to mirror from and list the operators to be mirrored. For example, to include the following operators from the Red Hat Operator index:

● OpenShift Logging

● Compliance Operator

The updated imageset-config.yaml will be:

| kind: ImageSetConfiguration  apiVersion: mirror.openshift.io/v2alpha1  archiveSize: 20  mirror:  platform:  channels:  - name: stable-4.19  minVersion: 4.19.9  maxVersion: 4.19.9  operators:  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.19  packages:  - name: cluster-logging  - name: compliance-operator |
| --- |

Use the same process to query the names of the operators to mirror from the other catalogues.

The final imageset-config.yaml as used is included in [Appendix A](#_34g0dwd) and is also saved in GitLab.

#### Mirroring

In the connected environment, use the following command to pull the images from the internet repositories and package them into one or more TAR file archives.

| $ oc mirror --config=./imageset-config.yaml \  file://<path\_to\_output\_directory> |
| --- |

Transfer the generated TAR archive files into the disconnected environment. Then use the *oc mirror* command to push the images from a local directory into the internal container registry.

| oc mirror -c imageset-config.yaml –from=file://<path-to-mirror-file-directory> docker://<container-registry>/openshift --v2 |
| --- |

The output from the mirroring process will generate YAML manifests that can be used to configure the installation configuration file and will need to be applied to the cluster post installation:

| 2025/10/15 [INFO] : === Results ===  2025/10/15 [INFO] : ✅ 190 / 190 release images mirrored successfully  2025/10/15 [INFO] : ✅ 11 / 11 operator images mirrored successfully  2025/10/15 [INFO] : 📄 Generating IDMS file...  2025/10/15 [INFO] : workspace/working-dir/cluster-resources/idms-oc-mirror.yaml file created  2025/10/15 [INFO] : 📄 Generating ITMS file...  2025/10/15 [INFO] : workspace/working-dir/cluster-resources/itms-oc-mirror.yaml file created  2025/10/15 [INFO] : 📄 Generating CatalogSource file...  2025/10/15 [INFO] : workspace/working-dir/cluster-resources/cs-redhat-operator-index-v4-19.yaml file created  2025/10/15 [INFO] : mirror time : 16m47.605785697s  2025/10/15 [INFO] : 👋 Goodbye, thank you for using oc-mirror |
| --- |

We recommend that these generated manifest files be copied into the cluster configuration project in GitLab.

Note that new OpenShift packages will result in new release signatures and new Operator images will result in new *ImageDigestMirrorSet* (IDMS), *ImageTagMirrorSet* (ITMS) and *CatalogSource* manifests being created.

Reference:

<https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/installing_an_on-premise_cluster_with_the_agent-based_installer/understanding-disconnected-installation-mirroring>

## Cluster Deployment

The clusters are installed using the OpenShift Installer Agent installation method

The cluster must be deployed in FIPS mode. FIPS mode is enabled by setting a flag in the installation configuration file and using the openshift FIPS installer binary. In addition, the installation ISOs must be generated on a RHEL machine that also has FIPS mode enabled.

~~The cluster is installed with the node’s disk encrypted using an existing Tang server already configured in the environment.~~

**Note:** due to issues documented in [Networking](#_v9vxak2t5yog), the clusters are initially being installed with the node’s disk encrypted with the onboard Trusted Platform Module (TPM) as the Tang servers can not be contacted until the OpenShift cluster configures the virtual NICs during startup. Encryption could be changed to NBDE if required after reconfiguring the networking.

The installation process consumes an installation configuration file that describes the cluster installation and an agent configuration file that describes the cluster networking and disk configuration.

An ssh key must be provided to the installation configuration that can be used to access the cluster nodes over ssh for troubleshooting purposes. To generate the ssh key:

| ssh-keygen -b 4096 |
| --- |

The installation configuration file specifies the following parameters:

| **Parameter** | **Description** |
| --- | --- |
| baseDomain | The domain that cluster is installed in |
| compute.replicas | The number of compute machines to provision |
| controlPlane.replicas | The number of control plane machines to provision |
| metadata.name | The name of the cluster |
| Cluster network CIDR | The CIDR range of the network used by the cluster nodes |
| Pod network CIDR | The CIDR range of the network used by pods within the cluster |
| Service Network CIDR | The CIDR range of the network used by services within the cluster |
| FIPS mode | Install the cluster in FIPS mode |
| pullSecret | Provides credentials for the cluster to access the local container registry mirror. |
| sshKey | An sshKey to enable emergency ssh access to the nodes |
| imageDigestSources | Configures the cluster to pull the required container images from the local container registry mirror. |
| additionalTrustBundle | The root and intermediate CAs used to sign certificates in the environment. |

*Table 9: Installation configuration parameters*

**Note** that the control plane machines are configured so that user workloads will not be allowed to run on these nodes. However, as the control plane machines will be under-utilised, if extra CPU and memory resources are required in the cluster then the control plane could be reconfigured to also run user workloads.

The installation configuration file is saved in the DSTG GitLab repository under bootstrap/install-manifests and is also saved in[**Appendix B: Installation configuration**](#_wjgur7oqx2hz).

The agent configuration file that describes the cluster networking and disk configuration specifies the following parameters:

| **Parameter** | **Description** |
| --- | --- |
| metadata.name | The name of the cluster |
| rendezvousIP | The IP address of the host that will bootstrap the installation control plane |
| hosts | Array of configuration parameters for each host. Each entry defines the following |
| hostname |  |
| interfaces | An array listing the network interfaces to be configured specifying interface name and MAC address |
| rootDeviceHints | Disk device name for the O/S |
| networkConfig | Detailed network config for each interface, specifying   * interface name * MAC address * MTU * type, eg: ethernet * state * ipv4 config, eg: dhcp or static * ipv6 config * DNS * Routing   **Note** that due to the requirement to trunk the VLANs to the interfaces, there are 2 NICs configured:   1. Physical NIC 2. Virtual VLAN NIC |

*Table 10: Agent configuration parameters*

The agent configuration file is saved in the DSTG GitLab repository containing the cluster configuration YAML under bootstrap/install-manifests and is also saved in [**Appendix B: Installation configuration**](#_wjgur7oqx2hz) of this document.

The installation also uses the latest Red hat CoreOS image in ISO format. The installer downloads this ISO from the internet or from the internal container registry and uses it to build an ISO with the cluster configuration that can be used to provision the OpenShift nodes.

Once the installation has been started, the installer will monitor and report on the progress. Initial credentials for accessing the cluster will be generated by the installer and saved in the install directory. An installation log file is also written to the install directory.

### Deployment steps

On the installation machine in the directory where the openshift-install-fips binary has been unpacked

1. Create a sub-directory for the installation

| mkdir cluster |
| --- |

1. Copy the install-config.yaml and agent-config.yaml file into the directory

| cp install-config.yaml agent-config.yaml cluster/ |
| --- |

1. Use the OpenShift installer to generate the install manifests

| openshift-install-fips agent create cluster-manifests --dir ./cluster |
| --- |

This command will consume the config YAML files and create manifest YAML file in the sub-directory

1. Edit the following generated file to add the encryption JSON  
   cluster/cluster-manifests/agent-cluster-install.yaml  
     
   The encryption JSON should be added after the spec: tag and will be of the following format:

| ~~spec:  diskEncryption:  enableOn: all  mode: tang   tangServers: '[{"url":"http://tang.example.com:7500","thumbprint":"PLjNyRdGw03zlRoGjQYMahSZGu9"}]'~~ |
| --- |

~~Note the tangServers: key and the value should all be on the one line~~

| spec:  diskEncryption:  enableOn: all  mode: tmpv2 |
| --- |

Save the file.

1. Use the OpenShift installer to generate the OpenShift agent ISO in the sub-directory

openshift-install-fips agent create image --dir ./cluster

Access the IDRAC interface on each machine to mount the ISO and boot the machine to initiate the cluster installation.

## Monitoring the installation

The installation can be monitored in multiple ways.

### Install machine

From the install machine, run the following command to monitor the progress of the installation:

| openshift-install-fips agent wait-for bootstrap-complete -dir ./cluster |
| --- |

Once bootstrap completes and the command exists, run the following command to continue monitoring the installation:

| openshift-install-fips agent wait-for install-complete -dir ./cluster |
| --- |

### Rendezvous machine console

Access the machine console of the machine designated as the *Rendezvous server* via the IDRAC to observe the number of machines becoming available and ready for install.

### Rendezvous machine CLI

Use the ssh key generated for the cluster installation to access the CLI for the Rendezvous machine:

| ssh -i <private.key> core@<rendezvouz-ip> |
| --- |

Use the *journalctl* command that is displayed at login to to monitor the installation service.

### Kubernetes API

Once the Kubernetes API comes up, you can use the openshift client to monitor the progress of the cluster operators. First authenticate with the cluster using the generated kubeconfig file:

| export KUBECONFIG=<install-path>/cluster/auth/kubeconfig |
| --- |

Then monitor the cluster operators:

| oc get clusteroperators |
| --- |

For example:

| $ oc get co  NAME VERSION AVAILABLE PROGRESSING DEGRADED SINCE MESSAGE  authentication 4.19.6 True False False 8s  baremetal 4.19.6 True False False 15m  cloud-controller-manager 4.19.6 True False False 15m  cloud-credential 4.19.6 True False False 15m  cluster-autoscaler 4.19.6 True False False 15m  config-operator 4.19.6 True False False 15m  console 4.19.6 True True False 6m SyncLoopRefreshProgressing: working toward version 4.19.6 , 1 replicas available  control-plane-machine-set 4.19.6 True False False 15m  csi-snapshot-controller 4.19.6 False True False 9s CSISnapshotControllerAvailable: Waiting for Deployment  dns 4.19.6 True False False 2m  Etcd 4.19.6 True False False 15m  image-registry 4.19.6 True False False 23d  ingress 4.19.6 True False False 15m  insights 4.19.6 True False False 5m  kube-apiserver 4.19.6 True True False 15m NodeInstallerProgressing: 1 node is at revision 42; 0 nodes have achieved new revision 43  kube-controller-manager 4.19.6 True False False 15m  kube-scheduler 4.19.6 True False False 15m  Kube-storage-version-mi… 4.19.6 True False False 6m  machine-api 4.19.6 True False False 15m  Machine-approver 4.19.6 True False False 15m  machine-config 4.19.6 True False True 15m Failed to resync 4.19.6 because: the server was unable to return a response in the time allotted, but may still be processing the request (get machineconfigpools.machineconfiguration.openshift.io master)  marketplace 4.19.6 True False False 15m  Monitoring 4.19.6 True False False 4m  network 4.19.6 True False False 15m  node-tuning 4.19.6 True False False 23d  olm 4.19.6 True False False 3m  openshift-apiserver 4.19.6 True False False 19s  Openshift-controller-ma… 4.19.6 True False False 1m  openshift-samples 4.19.6 True False False 7m  Operator-lifecycle-mana… 4.19.6 True False False 15m  operator-lifecycle-mana… 4.19.6 True False False 15m  operator-lifecycle-mana… 4.19.6 True False False 5m  service-ca 4.19.6 True False False 15m  storage 4.19.6 True False False 15m |
| --- |

Errors and failure conditions will be reported throughout the install process. These should not be a problem and these will likely resolve themselves over the 30 to 45 minutes required for the install.

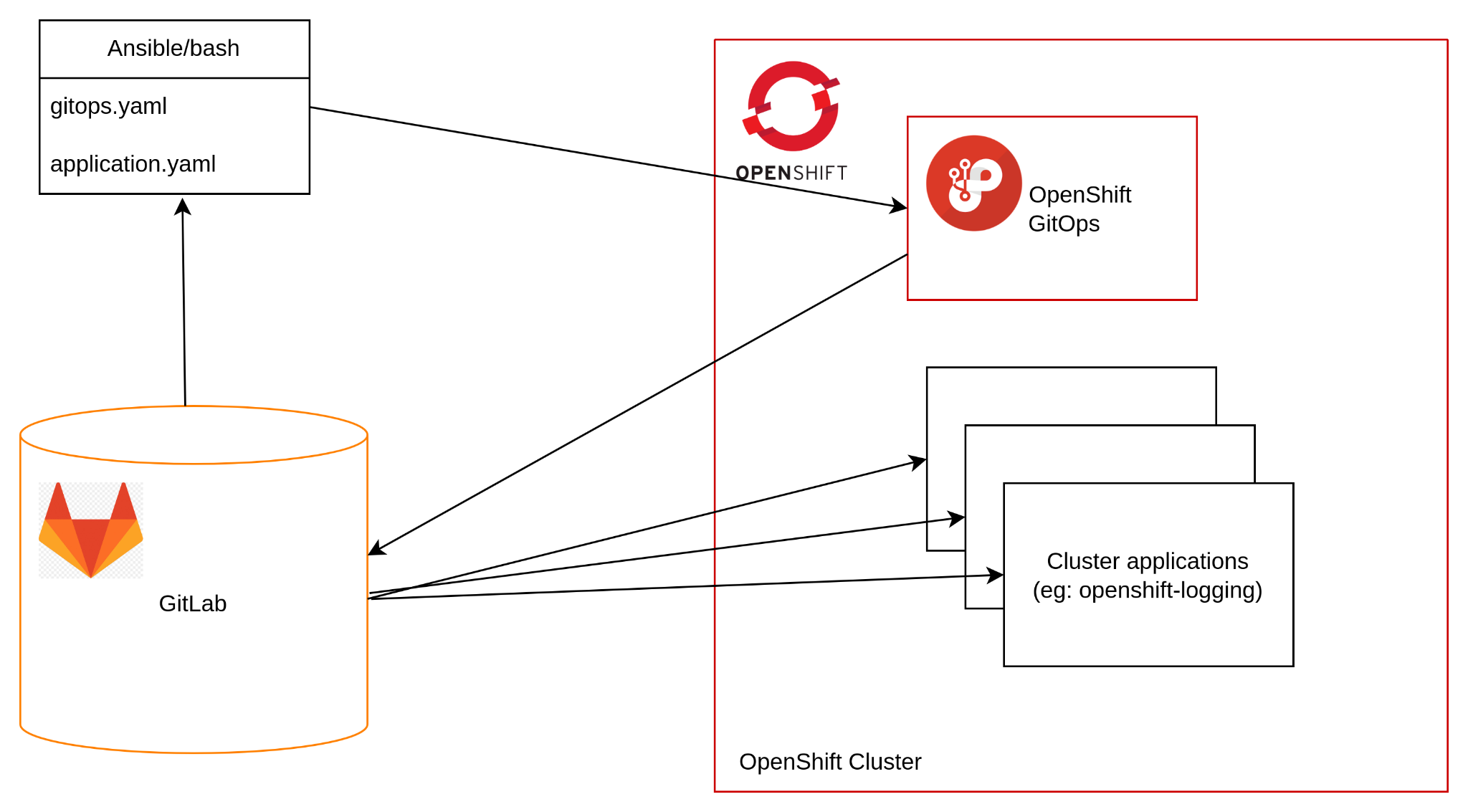
## Post deployment configuration

Post deployment configuration will be handled by the OpenShift GitOps Operator.

The OpenShift GitOps operator will need to be “bootstrapped” into the cluster. This is currently a manual process using the OpenShift client but could be automated using a tool such as Ansible. OpenShift GitOps is bootstrapped by applying YAML files from the bootstrap/openshift-gitops subdirectory in the GitLab source code management system. The cluster *kubeconfig* file will allow the OpenShift Client to authenticate. The *kubeconfig* file is created by the installer.

Applying the YAML manifests will configure the cluster to use the private container registry as the source for the mirrored operator images and then deploy the OpenShift GitOps operator. Then an “application of applications” OpenShift GitOps application YAML will be applied which will configure GitOps to apply the rest of the post-installation configuration from GitLab

The OpenShift GitOps “application of applications” is an application that references a location in the source code repository that holds the YAML configuration of the other OpenShift GitOps applications that perform the cluster configuration.



*Figure 2: OpenShift GitOps Applications*

OpenShift GitOps will continue to monitor the source code repository following installation and will apply any changes to the configuration that are committed in that repository. Any new configuration can be applied by creating another OpenShift GitOps application and committing it to the source code repository. OpenShift GitOps will also monitor the runtime configuration and can re-apply configuration that differs from the source code repository to ensure there is no configuration drift.

### Bootstrap OpenShift GitOps

OpenShift GitOps needs to be manually installed in the cluster post installation in order for it to be able to start applying the cluster configuration from the manifests in the GitLib project.

The first step is to apply the manifests that were created by the mirroring process that configure the cluster to look for the container images in the Harbor container registry rather than try to access them from the internet. The manifest file classes are:

* Catalogue source files that contain the indexes of the OpenShift operators
* Cluster Catalogues, an alternative newer definition of the catalogue sources
* The **ImageDigestMirrorSource** and **ImageTagMirrorSource** files that define the mirror registry for containers accessed via manifests and tags respectively.
* The OpenShift GitOps repository that describes the connection to GitLab including the credentials for a service account used for authentication.

The manifests are in the OpenShift GitOps project under bootstrap/cluster-resources. Apply these using the OpenShift client:

| export KUBECONFIG=<install-path>/cluster/auth/kubeconfig  oc apply -f <cloned-project-path>/bootstrap/cluster-resources/ |
| --- |

Install OpenShift GitOps from manifests in the OpenShift project in GitLab under bootstrap/openshift-gitops, this currently includes a secret containing the definition of the GitLab repository and the service account and password used to authenticate to GitLab:

| export KUBECONFIG=<install-path>/cluster/auth/kubeconfig  oc apply -f <cloned-project-path>/bootstrap/openshift-gitops/ |
| --- |

The first attempt to run this will result in an error as it attempts to create the GitOps application of applications. Until the operator has installed, the CRD for this application does not exist until the Operator has installed. Once the operator has finished installing, rerun the command above to create the application.

The configuration that is managed by OpenShift Gitops applications is:

* Disable the default internet based Operator catalogues,
* Configure integration with DSTG Active Directory servers for authentication,
* Configure integration with the DSTG Active Directory servers for group synchronisation,
* Configure integration to the Dell powerStore for persistent storage,
* Install the Cluster Logging operator and configure Log Forwarders to forward logs to the centralised logging endpoint,
* Install the NVidia GPU operator and prerequisites,
* Configure the project template with the resource quotas and default network policies,
* Remove the ability of non-admin users to create projects.

The applications that are configured to manage the configuration are shown in the following table.

| **Name** | **GitLab path** | **Notes** |
| --- | --- | --- |
| GitOps Applications | gitops-applications | The master application that creates all the other GitOps applications in the cluster. |
| General Config | general-config | Applies general configuration items. |
| Authentication | authentication-config | Configure integration with DSTG Active Directory servers for authentication |
| LDAP group sync | ldap-group-sync | Configure syncing LDAP groups from the DSTG AD servers |
| Network Policies | network-policies | Configure the second NIC on the iSCSI network |
| Openshift logging | openshift-logging | Forward audit logs to Splunk |
| Dell CSM operator | dell-csm-operator | Configure integration to the Dell powerStore for persistent storage |
| NVidia GPU Support | nfd-operator  gpu-operator | Install the NVidia GPU operator in preparation for GPU availability |
| Project config | project-configs | Configure the default project template |

*Table 11: OpenShift GitOps applications*

### General Config

This GitOps application holds multiple configuration items that are simple to apply, ie: with a single YAML manifest. The application makes the following updates:

* Disable the default internet based Operator catalogues
* Installs the DSTG CA certificates
* Applies the DSTG generated OpenShift ingress certificate
* Applies the DSTG generated OpenShift API certificate
* Configures IPSec encryption between pods in the cluster and to external endpoints

#### Operator catalogues

An OpenShift cluster is installed with the default Operator catalogue sources enabled. These catalogues attempt to connect to the Red Hat container registries over the internet. This configuration disables those default catalogue sources.

#### DSTG CA

The DSTG root CA certificate is installed into the OpenShift cluster and is made available to verify connectivity to external services. OpenShift GitOps applies a *ConfigMap* containing the root CA certificate in the *openshift-config* namespace and updates the Proxy configuration object to reference this *ConfigMap*.

#### OpenShift ingress certificate

Communications with the OpenShift Ingress endpoint, the entry point for applications running on the OpenShift cluster, is initially secured by a self signed wildcard certificate. This configuration replaces that certificate with the wildcard certificate signed by the DSTG CA. Further details are described above in section [Certificates](#_epym5yz23ng6).

As secrets management in GitLab has not yet been configured, the key and certificate generated for the cluster is saved in a *Secret* resource in YAML in GitLab. Storing secrets in source code repositories is not considered best practice and when a secrets management solution is configured, the key and certificate should be saved there.

**Reference:** <https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/security_and_compliance/configuring-certificates#replacing-default-ingress>

#### Openshift API Certificate

Communication with the OpenShift API endpoint is initially by a self signed certificate. This configuration replaces that certificate with the API certificate signed by the DSTG CA. Further details are described above in section [Certificates](#_epym5yz23ng6).

As secrets management in GitLab has not yet been configured, the key and certificate generated for the cluster is saved in a *Secret* resource in YAML in GitLab. Storing secrets in source code repositories is not considered best practice and when a secrets management solution is configured, the key and certificate should be saved there.

**Reference:** <https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/security_and_compliance/configuring-certificates#api-server-certificates>

#### IPSec encryption

IPSec encryption is enabled in the cluster to encrypt both internal pod-to-pod cluster traffic between nodes and external traffic between pods and endpoints outside of the cluster.

**Reference:** <https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/network_security/configuring-ipsec-ovn>

### Authentication

This GitOps application configures the integration with DSTG Active Directory servers for authentication using the LDAP protocol.

All users who have an LDAP account are able to authenticate to OpenShift although they will by default have no privileges to do anything in the cluster. This can be restricted further if required by configuring the RFC 2255 compliant URL to restrict access via an LDAP search filter.

A service account has been generated to act as the bind account to the AD server for the LDAP configuration. The bind account password is stored in a secret in the *openshift-config* namespace.

OpenShift authentication to the LDAP provider requires a separate *ConfigMap* object containing the CA for the LDAP server. This is a different CA *ConfigMap* to that created above in [General Config](#_h5g7g5nlezho) although the certificate is the same.

**Reference:** <https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/authentication_and_authorization/configuring-identity-providers#configuring-ldap-identity-provider>

### LDAP group sync

This GitOps application configures the synchronisation of groups from the DSTG AD servers into OpenShift. This configuration is discussed in detail in section [RBAC](#_sqyw64).

### Network Policies

This GitOps application installs the OpenShift **NMState** operator and configures the OpenShift nodes to configure the iSCSI network NICs to enable connectivity with the Dell PowerStore.

The network configuration applied by this application is documented in the section [iSCSI VLAN](#_t1xpguv93knv). The configuration is applied by the creation of a NodeNetworkConfigurationPolicy object describing the network configuration for each node.

**Reference:** <https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/kubernetes_nmstate/k8s-nmstate-updating-node-network-config>

### OpenShift Logging

This GitOps application installs the **OpenShift Logging** operator and configures it to forward audit logs to the DSTG Syslog server.

**Reference:** <https://docs.redhat.com/en/documentation/red_hat_openshift_logging/6.3>

### Dell CSM Operator

Storage for persistent volumes is provisioned from the Dell PowerStore 3200T appliance. The Dell PowerStore presents LUNs to the nodes over iSCSI. The Dell CSM driver provides the interface between the OpenShift cluster and the Dell PowerStore. When a developer or administrator requests storage, either by creating a PersistentVolume or PersistentVolumeClaim resource, the CSM driver interacts with the storage appliance to provision a volume and attaches it to the cluster over the iSCSI interfaces.

This GitOps application installs the Dell CSM driver and configures it with the details of the Dell PowerStore. The Dell CSM driver requires a service account be created on the Dell PowerStore to authenticate requests for storage.

The Dell CSM driver documentation referenced below includes a script that creates a TAR archive that contains all the container images and YAML manifests for installing the operator in a disconnected environment. The script was used to create the bundle in the lower environment and then to push the transferred content into the disconnected Harbor container registry.

The Dell *install.sh* script included in the bundle was used to install the operator.

The Dell CSM driver prerequisites require creating *MachineSet* resources that enable the iSCSI service and create the multi-path configuration on the compute nodes.

This GitOps application configures the Dell CSM driver and creates the storage classes that applications use to request persistent storage. The configuration includes a service account that authenticates the requests from OpenShift to the Dell PowerStore.

**Note:** another NIC and VLAN should be configured to allow multi-path access to the storage array

**Reference:** <https://dell.github.io/csm-docs/docs/getting-started/installation/openshift/powerstore/csmoperator/>

### NVidia GPU support

This GitOps application installs the **NVidia GPU Operator** and the **Node Feature Discovery** operator in preparation for the integration of nodes that are installed with GPUs. Once these nodes have been added, further configuration will be required.

**Reference:** https://docs.redhat.com/en/documentation/openshift\_container\_platform/4.13/html/architecture/nvidia-gpu-architecture-overview

### Project configuration

This GitOps application creates the customised project configuration for the DSTG.

By default, any user who can login to an OpenShift cluster can create a project for their own use. In the DSTG configuration this ability has been removed by this GitOps application by removing the default *self-provisioner* role mapping from the cluster. This means that a user who can login to the cluster but has no further role bindings can not do anything in the cluster and must be given permission to a project created by a cluster administrator.

#### Network Policy

The GitOps application creates a custom project template that applies to all new projects created in the cluster. New projects are created with a default set of network policies that limits the access to and from the applications running in a project.

The network policies are:

* *allow-from-openshift-ingress* - to allow the applications to be accessible to end users only via an OpenShift Route,
* *allow-same-namespace* - to allow applications within the same namespace to communicate freely. Note that a namespace and a project are effectively the same thing,
* *allow-from-openshift-monitoring* - to allow the in-built OpenShift monitoring stack to monitor resources in the project,
* *allow-from-kube-apiserver-operator* - to allow the Kubernetes API server to access resources in the project.

#### Quotas

The GitOps application adds storage quotas to the custom project template that applies to all new projects created in the cluster. New projects are created with a default limit on the amount of storage that can be allocated by all applications in the project. The storage quota limits the number of persistent volumes that can be allocated to 10 and the maximum size of each persistent volume to 50G. This limits the amount of storage that the project can consume to 500GB.

**References:**

<https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/building_applications/projects#disabling-project-self-provisioning_configuring-project-creation>

<https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/network_security/network-policy>

<https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/scalability_and_performance/compute-resource-quotas>

# 

# Management

Cluster administrators and developers will have access to the OpenShift web console and API from the workstations. The cluster can be managed from either the web console in a browser or the API uses the OpenShift Client CLI tool.

Changes to the cluster configuration should be made by updating the configuration files saved in the source code management system. Once committed, the changes will be applied to the clusters automatically by OpenShift GitOps.

## Cluster Monitoring

OpenShift includes a preconfigured, preinstalled, and self-updating monitoring stack that provides monitoring for core platform components and user-defined workloads accessible from the web console. A set of alerts are included by default that immediately notify administrators about issues with a cluster. Default dashboards in the web console include visual representations of cluster metrics to help administrators quickly understand the state of the cluster. The web console allows administrators to view and manage metrics, alerts, and review monitoring dashboards.

The core monitoring stack is automatically deployed. The main components are:

* Prometheus: A time series database and a rule evaluation engine for metrics.
* AlertManager: Handles alerts received from Prometheus and sends alerts to external notification systems.
* Thanos Querier: This aggregates core OpenShift Container Platform metrics for user-defined projects and workloads. It provides an interface to query metrics from Prometheus for external applications like Custom Metrics Autoscaler based on KEDA and others.

Alert targets can be configured as consumers of alerts. Alerts can be forwarded via email, or through integrations with Pager Duty, Slack or to a generic webhook endpoint.

**Reference:**

<https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/monitoring/about-openshift-container-platform-monitoring>

### Monitoring user workloads

The OpenShift monitoring stack has been configured to enable monitoring for user-defined projects This centralises monitoring for core platform components and user-defined projects.

**Reference:**

<https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/monitoring/configuring-user-workload-monitoring#enabling-monitoring-for-user-defined-projects-uwm_preparing-to-configure-the-monitoring-stack-uwm>

## Patching

Operators and other images that were previously mirrored from the various approved sources into the DSTG Harbor container registry will need to be regularly updated to ensure the latest patches and versions are available.

Updating the Red Hat platform and Operator images is performed manually using the *oc-mirror* plugin in the same way as the original mirroring described above in section [Content mirroring](#_cmi5h7wevng5). The *oc-mirror* plugin will only download updates to the configured content and will also purge any content that is no longer required.

The mirroring will need to be run manually and the content transferred into the environment. Once transferred, the *oc-mirror* plugin can be used to push the updated mirrored content into the Harbor container registry.

### Platform

OpenShift updates are published by Red Hat in the online Red Hat registries. Updates are new container images that run the OpenShift platform services and also include updates to the node’s Red Hat CoreOS operating system. There is no requirement for separate manual patching of the operating system.

The container images that provide the OpenShift platform services that were previously mirrored from the Red Hat source images into the DSTG container registry will need to be regularly updated to ensure the latest patches and versions are available.

An internet connected cluster will show administrators what updates are available for their cluster and will automatically manage the application of any required intermediate patches. In a disconnected environment, this functionality in the web console is not available. Before deciding on an update, administrators should manually determine what patch level to mirror using the Red Hat OpenShift Update Graph. The graph application is a web-based tool that helps administrators plan cluster upgrades by showing the available OpenShift versions and valid upgrade paths between them. The graph visually displays supported version transitions, highlighting recommended upgrade targets and any blocked or conditional paths. The Graph application can be accessed from the following URL:

**Reference:** <https://access.redhat.com/labs/ocpupgradegraph/update_path>

Once the new platform images have been identified and mirrored into the container registry, the cluster administrators can use a single command to apply the update patch level. The command needs the digest of the chosen version and must specify the force parameter to bypass the online validation process.

**Note:** only the desired target version needs to be mirrored. It is not necessary to mirror all the intermediate releases, only those identified by the Graph application.

To determine the digest and the apply the target patch, use the following commands:

| $ podman login harbor.example.com $ podman pull harbor.example.com/ocp/openshift-release-dev/ocp-release:4.19.14-x86\_64 $ podman inspect [harbor.example.com/ocp/openshift-release-dev/ocp-release:4.19.14-x86\_64](http://harbor.spenscot.ddns.net/ocp/openshift-release-dev/ocp-release:4.19.14-x86_64)|grep digest "io.openshift.release.base-image-digest": "sha256:fa40471ae16d79758c97d8e281328a8a0b6dd89a1b3392ce882be252058f2b58" "io.openshift.release.base-image-digest": "sha256:fa40471ae16d79758c97d8e281328a8a0b6dd89a1b3392ce882be252058f2b58" |
| --- |

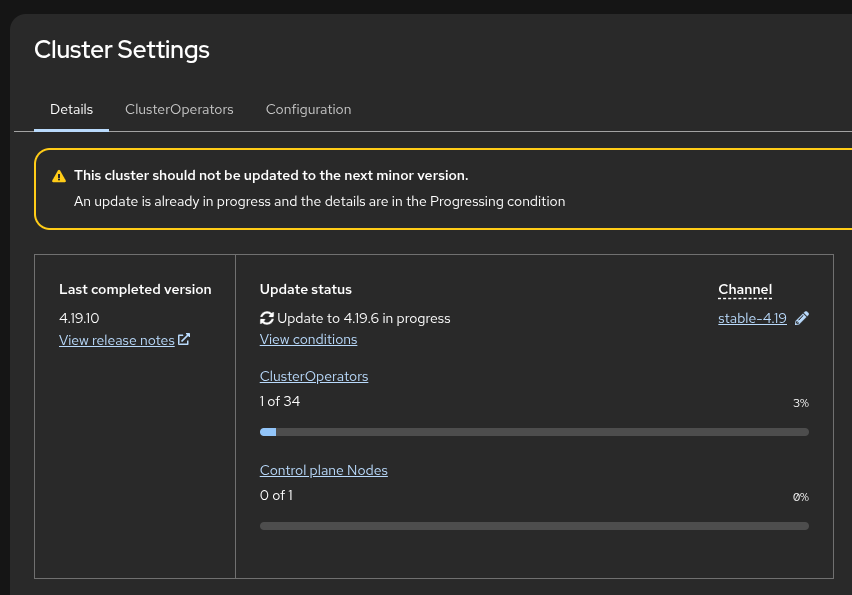
Use the digest display in the podman inspect output to initiate the upgrade:

| $ oc adm upgrade --allow-explicit-upgrade --to-image \ harbor.example.com/quay/openshift-release-dev/ocp-release@sha256:02ec914b5380b9e4e048b830c9521e8d11f7f613d4ff3977147107770288a595 \ --force=true |
| --- |

#### Monitoring the upgrade

The update can be monitored from the OpenShift web console. A banner will be displayed at the top of the web console identifying the target version.

Navigating to **Administration > Cluster Settings** will show the progress of the update, for example:



*Figure 3: Monitoring the installation*

This information is also available from the CLI:

| $ oc get clusterversion NAME VERSION AVAILABLE PROGRESSING SINCE STATUS version 4.19.10 True True 56s Working towards 4.19.6: 113 of 920 **done** (12% complete), waiting on etcd, kube-apiserver |
| --- |

The Cluster Operators tab under Administration > Cluster Settings and form the CLI will also show the update progress through the cluster components:

| $ oc get co  NAME VERSION AVAILABLE PROGRESSING DEGRADED SINCE MESSAGE  authentication 4.19.10 True False False 8s  baremetal 4.19.10 True False False 321d  cloud-controller-manager 4.19.10 True False False 321d  cloud-credential 4.19.10 True False False 321d  cluster-autoscaler 4.19.10 True False False 321d  config-operator 4.19.6 True False False 321d  console 4.19.10 True True False 60d authentication 4.19.10 True False False 8s  baremetal 4.19.10 True False False 321d  cloud-controller-manager 4.19.10 True False False 321d  cloud-credential 4.19.10 True False False 321d  cluster-autoscaler 4.19.10 True False False 321d  config-operator 4.19.6 True False False 321d  console 4.19.10 True True False 60d SyncLoopRefreshProgressing: working toward version 4.19.10, 1 replicas avaiSyncLoopRefreshProgressing: working toward version 4.19.10, 1 replicas available  control-plane-machine-set 4.19.10 True False False 321d  csi-snapshot-controller 4.19.10 False True False 9s CSISnapshotControllerAvailable: Waiting for Deployment  dns 4.19.10 True False False 2d22h  Etcd 4.19.6 True False False 321d  image-registry 4.19.10 True False False 23d  ingress 4.19.10 True False False 321d  insights 4.19.10 True False False 55d  kube-apiserver 4.19.10 True True False 321d NodeInstallerProgressing: 1 node is at revision 42; 0 nodes have achieved new revision 43  kube-controller-manager 4.19.10 True False False 321d  kube-scheduler 4.19.10 True False False 321d  Kube-storage-version-mi… 4.19.10 True False False 60d  machine-api 4.19.10 True False False 321d  Machine-approver 4.19.10 True False False 321d  machine-config 4.19.10 True False True 321d Failed to resync 4.19.10 because: the server was unable to return a response in the time allotted, but may still be processing the request (get machineconfigpools.machineconfiguration.openshift.io master)  marketplace 4.19.10 True False False 321d  Monitoring 4.19.10 True False False 24d  network 4.19.10 True False False 321d  node-tuning 4.19.10 True False False 23d  olm 4.19.10 True False False 39d  openshift-apiserver 4.19.10 True False False 19s  Openshift-controller-ma… 4.19.10 True False False 14d  openshift-samples 4.19.10 True False False 77d  Operator-lifecycle-mana… 4.19.10 True False False 321d  operator-lifecycle-mana… 4.19.10 True False False 321d  operator-lifecycle-mana… 4.19.10 True False False 51m  service-ca 4.19.10 True False False 321d  storage 4.19.10 True False False 321d |
| --- |

Errors and failure conditions will be reported throughout the update process, for example:

| NAME VERSION AVAILABLE PROGRESSING SINCE  clusterversion.config.openshift.io/version 4.19.10 True True 19m Unable to apply 4.19.6: an unknown error has occurred: MultipleErrors |
| --- |

As with the installation it should not be a problem and these will likely resolve themselves over the 30 to 45 minutes required for the upgrade. In most cases applications will continue to be available throughout the update process.

### Operators

Updates to OpenShift Operators can only be applied after the updates are mirrored into the DSTG managed ACR. Once mirrored, they can either be applied automatically or will require DSTG cluster admin to approve the updates depending on the Operator configuration. Operator updates are published under *channels*. A *channel* defines a specific update stream for an Operator, controlling how and when new versions of that Operator are made available. Each channel typically represents a release track or may correspond to a specific OpenShift version (e.g., *stable-4.14*).

In general, manual updates to Operators are recommended as this allows the updates to be scheduled as required by the workloads running in the clusters that consume or rely on the deployed Operators.

If an operator is set to automatic upgrades, the upgrade will automatically occur soon after the container images are mirrored into the disconnected container registry.

If an operator is set to manual upgrades, the upgrade must be manually approved by the administrator. Approval can be given either through the web console by clicking on the available upgrade, previewing the *installPlan* and then clicking the Approve button.

Some operators, such as the OpenShift Logging Operator, publish multiple channels, eg: *stable-6.3.* If a new channel is released, then the administrators can choose to switch to the new channel by updating the Operator subscription to the new channel value, eg: *stable-6.4*. Switching channels is always a manual process. Once the channel has been switched, the update to the new version will automatically start, unless manual upgrades are chosen in which case the *installPlan* will need to be approved.

An example of updating the OpenShift Logging operator from the CLI is to first query the installed operators (or ClusterServiceVersions)

| $ oc get csv NAME DISPLAY VERSION REPLACES PHASE cluster-logging.v6.3.0 Red Hat OpenShift Logging 6.3.0 Succeeded |
| --- |

Check the status of the operator by querying the subscription:

| $ oc -n openshift-logging get subscription cluster-logging \  -o jsonpath='{.status.state}' UpgradePending |
| --- |

Determine the *installPlan* that needs approval (i.e. APPROVED=false) and then approve that *installPlan*:

| $ oc -n openshift-logging get installplan NAME CSV APPROVAL APPROVED install-jcxxw cluster-logging.v6.3.0 Manual true install-pr7pw cluster-logging.v6.3.1 Manual false |
| --- |

| $ oc -n openshift-logging patch installplan install-pr7pw \  --type merge -p '{"spec":{"approved":true}}' installplan.operators.coreos.com/install-pr7pw patched |
| --- |

The operator upgrade will then progress.

## Service account management

Service accounts have been defined in external systems to allow the OpenShift cluster permissions to interact with those systems. These service accounts will need to have their passwords rotated regularly. The following table shows the service accounts that have been created and in what namespace the secret containing the password has been created.

| **Component** | **Purpose** | **Secret namespace** |
| --- | --- | --- |
| GitLab | Allow read access to the configuration code repository | openshift-gitops |
| Active Directory | Allow bind to AD for user authentication | openshift-authentication |
| Dell PowerStore | Allow Dell CSM driver to create, access and delete volumes | dell-csm-operator |

*Table 12: Service accounts*

## Certificate Management

The external certificates installed in the cluster will need to be renewed regularly.

All of the certificates that are used internally in the cluster for communications between pods and services and between internal components and the API server are automatically managed. These do not need to be manually renewed.

### Ingress key and certificate

Replacing the OpenShift Ingress wildcard key and certificate follows the same process as the initial installation documented in section [OpenShift ingress certificate](#_bw3v9yy12ij6).

In particular, as the key and certificate are coded in a *Secret* definition in the source code repository, the process is to update the *Secret* definition with the new key and certificate and commit the change. OpenShift GitOps will then apply the new certificate to the cluster.

### API key and certificate

Replacing the OpenShift Ingress wildcard key and certificate follows the same process as the initial installation documented in section [Openshift API Certificate](#_3h8fh01n3f39)

In particular, as the key and certificate are coded in a *Secret* definition in the source code repository, the process is to update the *Secret* definition with the new key and certificate and commit the change. OpenShift GitOps will then apply the new certificate to the cluster.

### Root CA certificate

If the root CA certificate is re-issued it will need to be replaced in the *ConfigMap* objects in the cluster.

The primary location for the root CA is in a *ConfigMap* referenced by the Proxy configuration as described in section [DSTG CA](#_oyclu3hoyxat). To update the CA certificate, update the CA config map YAML file in the GitLab *general-config* application with the new certificate and commit the changes. OpenShift GitOps will then apply the new certificate to the cluster.

Most applications can make use of the root CA certificate in the primary CA *ConfigMap* referenced from the Proxy configuration. Applications that require a *ConfigMap* with the CA certificate in their own namespace can create an automatically managed CA *ConfigMap* by creating a blank *ConfigMap* and labelling it to be a managed CA *ConfigMap* as in the following example:

| kind: ConfigMap apiVersion: v1 metadata:  labels:  config.openshift.io/inject-trusted-cabundle: "true"  name: my-ca-bundle  namespace: test spec: {} |
| --- |

When the main CA *ConfigMap* is updated, all of the dependent injected *ConfigMaps* will be updated.

In addition to the main CA *ConfigMap*, OpenShift Authentication requires a separately managed CA *ConfigMap* to be updated. This CA *ConfigMap* is in GitLab under the *authentication* project. Update the root certificate in this configmap YAML file and OpenShift GitOps will apply the update to the cluster.

## Power off or reboot a node

Powering off or rebooting a node can be done from the Dell IDRAC interface. Before a node is powered off or rebooted, it should be marked *unschedulable* so that the Kubernetes scheduler does not try and start any new workloads on the node, and the node drained so that existing pods are removed from the nodes and started up on other nodes.

To drain a node and mark it *unschedulable*, use the openshift-client from the CLI:

| oc adm drain node <nodename> --ignore-daemonsets=true \  --delete-emptydir-data --force=true |
| --- |

Once the drain has completed reboot the node using the IDRAC.

After the node has rebooted, it will become *Ready* but will remain in *Unschedulable* state:

Make the node ready and available for workloads again by *uncordoning* it:

| oc adm uncordon <node\_name> |
| --- |

**Reference:**

<https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/scalability_and_performance/managing-bare-metal-hosts#powering-off-bare-metal-hosts-web-console_managing-bare-metal-hosts>

## Cluster shutdown and startup

Shutting down a cluster consists of draining each compute node and marking it as *unschedulable* from the CLI. You can not do this from the web console as it will become unavailable during the process.

Once all the compute nodes are unschedulable and drained, shut the compute nodes and then the control plane down from the IDRAC console.

Restarting the control plane nodes and then the compute nodes from the IDRAC interface. Once the control plane becomes available, you can set the compute nodes back to the ready state as described above in [Power off or reboot a node](#_qzr6oouskffr).

**References:** <https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/backup_and_restore/graceful-shutdown-cluster>

<https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/backup_and_restore/graceful-restart-cluster>

## Adding nodes

Adding new nodes to the cluster is a similar process to that described above in [Cluster Installation](#_4f1mdlm).

First create a hosts file that contains details about the new node(s) similar to the agent-config.yaml created for the initial installation. Then use the following command to generate an ISO for the new nodes to boot:

| oc adm node-image create nodes-config.yaml |
| --- |

Use that ISO to boot the new node from the IDRAC. The node will install and be automatically added to the cluster.

**Reference:** <https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/nodes/working-with-nodes#adding-node-iso>

# Security Architecture

Network allocations for the clusters include unique IP address ranges for the non-routable service and pod networks. While IP addresses in these ranges can not be accessed outside of the cluster, they will appear in pod and audit logs from the cluster and are unique so that the originating cluster can be easily identified by the network monitoring and security teams.

The following tables show the network allocation ranges for the clusters.

| **Subnet type** | **CIDR** |
| --- | --- |
| Control plane nodes |  |
| Compute nodes |  |
| Service CIDR |  |
| Pod CIDR |  |

*Table 13: Cluster CIDR ranges*

## Network Connectivity

The following network connections are required by each of the clusters.

| **Source** | **Port** | **Protocol** | **Direction** | **Destination** | **Detail** |
| --- | --- | --- | --- | --- | --- |
| DSTG on premises | 3389 | Remote Desktop | In | Jump box | Admin access |
| Jump box | 22 | ssh | In | Linux Management VM | Admin access |
| External container registries | 443 | Docker v2 | Pull in | Management VM | Explicit oc mirror |
| Linux Management VM | 443 | Docker v2 | Push out | DSTG container registry | Explicit oc mirror |
| Linux Management VM | 6443 | REST | In | OpenShift | API access |
| Linux Management VM | 443 | HTTPS | In | OpenShift | Console UI access |
| OpenShift | 443 | REST | Out | vSphere API |  |
| OpenShift | 443 |  | Out | Storage cluster |  |
| OpenShift | 443 | REST | Out | Secrets vault |  |
| OpenShift | 443 | Docker v2 | Out | DSTG container registry |  |
| OpenShift | 514 | syslog | Out | Syslog server | Logging |
| OpenShift |  |  | Out | Splunk | Logging |
| OpenShift | 443 | HTTPS | In | GitLab | GitOps |
| OpenShift | 636 | LDAP | In | DSTG on-prem AD | Authentication |
| OpenShift | 636 | LDAP | In | DSTG on-prem AD | Group sync |

## Encryption

### At Rest

The node O/S disks are encrypted using LUKS encryption. The disk is unencrypted via the on-board Trusted Platform Module (TPM) module on the servers.

Network Bound Disk Encryption (NBDE) is preferred but due to issues documented in [Networking](#_v9vxak2t5yog), the clusters are initially being installed with the node’s disk encrypted with the onboard TPM as the Tang servers can not be contacted until the OpenShift cluster configures the virtual NICs during startup. Node disk encryption could be changed to NBDE if required after reconfiguring the networking.

Encryption of persistent data in the Dell PowerStore is TBD.

### In Transit

All external communications into the OpenShift cluster will be over HTTPS and secured with DSTG signed SSL certificates.

All pod to pod communications within the cluster and all communications from the cluster to the external endpoints will be secured with IPSec encryption.

## Authentication

OpenShift will be configured to connect to Active Directory in the DSTG internal network. Active Directory will provide identity management and authentication services for users logging into the CLI and the web UI.

All administrators requiring access to the OpenShift console UI and OpenShift API will be required to authenticate with their DSTG accounts. Similarly, developers requiring access to the OpenShift console UIs and OpenShift API will be required to authenticate with their DSTG accounts.

Reference: <https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/authentication_and_authorization/configuring-identity-providers#configuring-ldap-identity-provider>

## RBAC

Role Based Access Control (RBAC) maps users or groups to OpenShift roles. OpenShift applies authorisation through roles such as

* cluster-admin: has full permissions to everything in the cluster,
* admin: has full permission within one or more *Projects* as applied,
* edit: has edit permissions within one or more *Projects* as applied,]
* view: read only access

There are many more roles available in the platform and custom roles can be created as required to meet specific use cases.

Groups and users can be bound to zero or more roles to manage the access that any given user has on the platform.

Groups and group membership are automatically synchronised from AD into OpenShift on a schedule so that when a user logs into the cluster, they will automatically be assigned the RBAC permissions mapped to the group via the appropriate role.

Automatic group sync consists of the following configuration items:

* *Namespace*, to hold all the resources, eg: **ldap-sync**,
* *ConfigMap* containing the DSTG CA certificate, this config map is created as an empty config map and the label service.beta.openshift.io/inject-cabundle=true used to inject the CA certificate from the system CA,
* *Secret* containing the LDAP service account bind password,
* *ServiceAccount*, *ClusterRole* and *ClusterRoleBinding* allowing the group sync to manage groups,
* *ConfigMap* to specify the LDAP sync configuration,
* An allow list to name the groups to sync into OpenShift.   
  **Note:** that an allowlist is mandatory when syncing nested groups,
* *CronJob* that runs the oc adm group sync command; the *CronJob* schedule uses the Linux crontab format.

The LDAP sync configuration in the *ConfigMap* uses the augmented Active Directory schema with nested members, allowing for the nested groups in AD to be flattened when synced into OpenShift.

The following AD group to role mappings will be applied in each cluster.

| AD Group Name | OpenShift Group Name | OpenShift Role |
| --- | --- | --- |
|  |  | cluster-admin |
|  |  |  |

*Table 14: OpenShift cluster group mappings*

## Logging

OpenShift components produce different log types. Container logs are generated from both OpenShift component containers and application workload containers and can be used for application monitoring and troubleshooting. The OpenShift and Kubernetes API servers and the Red Hat CoreOS operating system produce audit logs for security auditing and incident detection purposes.

The audit logs produce a lot of data, much of relating to internal processes within the cluster and much of the audit data relates to read activity. Red Hat recommends using the Cluster Logging operator filter options to filter the audit data so that only audit data related to an update or delete activity be forwarded to the logging endpoint.

DSTG utilises Splunk for centralised logging. OpenShift will ultimately utilise DSTG’s existing logging solution, and forward logs to their internal Splunk instance. However until the Splunk configuration to provide an endpoint for OpenShift is complete, the logs will be forwarded to the existing *syslog* server.

| **Logging Source** | **Destination** | **Notes** |
| --- | --- | --- |
| Application logs | none |  |
| Infrastructure logs | none |  |
| Audit logs | DSTG syslog server | Filtered to exclude read (get/list/watch) API calls |

*Table 15: Logging targets*

References:

<https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/authentication_and_authorization/ldap-syncing>

<https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/authentication_and_authorization/ldap-syncing#ldap-syncing-nesting_ldap-syncing-groups>

## Secrets

Kubernetes secrets that are used by applications will ultimately be stored in the DSTG GitLab instance as project or global variables. The External Secrets Operator enables applications to

Applications can then use the secret, but the secret does not persist on the system after the application pod is destroyed.

Until GitLab has been configured to hold the data that will populate the secrets, Kubernetes secrets in the cluster will be used to hold this data.

# Appendix A: Mirror configuration

Imageset-config.yaml

# Appendix B: Installation configuration

## install-config.yaml

## agent-config.yaml