proNX Optical Director

Installation Guide

Modified: 2019-04-15

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

proNX Optical Director Installation Guide

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at https://support.juniper.net/support/eula/. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

CHAPTER 1

Introduction

proNX Optical Director Overview

NOTE: This section is intended to provide a brief and general overview of the proNX Optical Director and might contain a description of features not found in the release that you are running. See the *TCX Series Optical Transport System Release Notes* for information on features for the release you are running.

The proNX Optical Director is a software controller and management system for open optical line systems (OLS), initially providing support for TCX1000 Series devices.

The proNX Optical Director provides the following functionality:

- Dynamic real-time control of optical links in OLS networks. This includes automatic span loss management, automatic nodal loss management, and automatic channel power control.

  In traditional optical networks, this control function resides on the ROADMs themselves where the ROADMs exchange proprietary control messages with each other on an optical supervisory channel (OSC). This makes interworking across vendor equipment difficult and often leads to the deployment of single-sourced networks. Moving this function to a centralized software controller makes heterogeneous networks with equipment from multiple vendors possible.

- Network management of OLS networks including network topology, network visualization, and network monitoring and troubleshooting.

  The proNX Optical Director displays the topology of the network and provides various visual indicators so that you can see the health of the network at a glance and deal with problem areas in a proactive manner.

- Device management of OLS elements including device configuration, device visualization, and device monitoring and troubleshooting.

  The proNX Optical Director discovers OLS elements and reads and displays their configuration. You can change the configuration, view the equipment inventory, pull up a visual representation of the device, or view performance monitoring counters and alarm details.

  NOTE: TCX Series devices do not support a built-in user interface such as a command line interface. You must use the proNX Optical Director to manage TCX Series devices.

- Service management of optical services across an OLS network including service provisioning, service activation, and service monitoring and troubleshooting.

  The proNX Optical Director supports A-to-Z provisioning and activation of optical services. You select the two service endpoints and the proNX Optical Director provides you a list of paths that you can choose for that service. When you activate the service, the proNX Optical Director automatically configures the service across all the devices in the path.

- Endpoint management of supported transceivers on Juniper Networks equipment.

  The OLS network provides optical service connectivity between endpoint transponders (typically). These transponders can be standalone or integrated within routers and switches. Although these endpoints are not technically part of the OLS network, you can use the proNX Optical Director to configure these endpoints on Juniper Networks equipment that supports coherent DWDM interfaces.

- Support for IPv4 and IPv6 networks. The proNX Optical Director can control and manage IPv4 and IPv6 devices.

- Northbound RESTCONF interface for connecting to higher level management systems. See "Northbound Interface" on page 17.

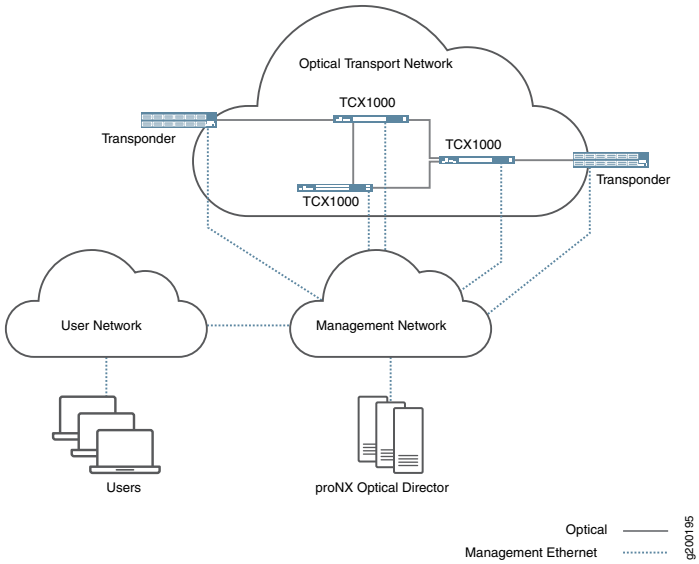
- Web-based user interface. You can access the proNX Optical Director user interface from supported web browsers.

Figure 1 on page 7 shows a high level view of a typical deployment where the proNX Optical Director cluster controls and manages a TCX1000 Series optical transport network.

The proNX Optical Director runs on a cluster of three Linux servers. Within the cluster, one server acts as the master node. The master node is a regular node that is additionally responsible for orchestration and scheduling functions.

The proNX Optical Director connects to managed devices over a management network. Users connect to the proNX Optical Director using supported web browsers on their own computers.

Figure 1: Typical proNX Optical Director Deployment



Having network-wide visibility allows the proNX Optical Director to use advanced control algorithms that optimize optical transmission not only on a link but along the whole optical service path. The TCX1000 Series devices constantly send streams of real-time optical link measurements to the proNX Optical Director, which then uses the data to build an always current view of the optical links in the network. This allows the proNX Optical Director to make real-time control decisions on all aspects of optical link management, including the following:

- channel power control and equalization
- span loss compensation
- gain ripple and tilt compensation
- graceful ramp up and ramp down of channel powers as optical services (wavelengths) are added and removed

These control decisions are translated into commands that are communicated to the managed devices for execution. This ongoing control loop allows the proNX Optical Director to deliver optimal optical transmission performance for the managed devices by dynamically and automatically controlling all aspects of optical link output.

proNX Optical Director Microservices Architecture

The proNX Optical Director application consists of a set of microservices that work together to provide the overall functionality of the controller and management system. The microservices architecture facilitates the development of complex, large scale systems by breaking an application down into smaller, self-contained, independently deployable building blocks. These building blocks or microservices run concurrently in their own processes and interact with each other through narrow, well-defined mechanisms.

When designed properly, microservices and their applications exhibit the following fundamental characteristics:

- Each microservice can be deployed and removed independently from and without affecting the other microservices. This results in reduced risk when performing software upgrades. Instead of a monolithic software upgrade, only the microservices that need to be updated are redeployed. Microservices that do not have updates are not disturbed.
- Fine-grained scaling can be achieved by deploying multiple instances of a particular microservice when needed. Additionally, because microservices are stateless, events can be load balanced across all instances of a particular microservice with no regard for prior affiliation when coupled with a well-designed caching strategy.
- The modular nature of microservices enables new microservices to be developed and tested quickly.
- Failure in one microservice does not affect the other microservices. This built-in fault tolerance enables the application to continue its operation, possibly at reduced capacity or with reduced capability, instead of failing completely when part of the system has a catastrophic failure.

The proNX Optical Director software leverages best-in-class technologies to provide an easy-to-deploy, robust, and scalable solution. Although knowledge of the underlying technologies is not required, advanced users might find the information in the following sections useful.

The proNX Optical Director application uses a Linux Docker Kubernetes (LDK) stack.

Linux

Atomic Host Linux is a stripped-down Linux distribution that is optimized to run containers within an immutable infrastructure. An immutable infrastructure is a design philosophy where components are replaced rather than upgraded in place.

Docker

Docker is a widely used container platform that enables applications to be developed as a set of independent, modular components. This modularity promotes isolation and improves robustness and scaling. Docker containers provide a lightweight virtual operating system for the contained microservices and supply all the components necessary for them to run, such as the runtime environment, environment variables, tools, and libraries. Because containers share the same kernel as the host operating system, containers provide most of the isolation benefits of virtual machines but without the overhead of a hypervisor.

You are not required to be familiar with Docker in order to install and use the proNX Optical Director.

Kubernetes

Kubernetes is a widely used container management and orchestration platform that provides the underlying capability for proNX Optical Director components to be deployed,

scheduled, and scaled, not only within a node but across nodes (such as in a server cluster). A node is a bare metal computer where you have installed the LDK stack.

A Kubernetes cluster consists of a master node and regular nodes. The master node is a regular node that has the additional responsibilities of coordinating all activity within the cluster, such as scheduling Kubernetes pods, maintaining a pod's desired state, scaling pods, and rolling out new updates. A Kubernetes pod is the smallest unit that can be managed by Kubernetes and consists of one or more tightly coupled containers.

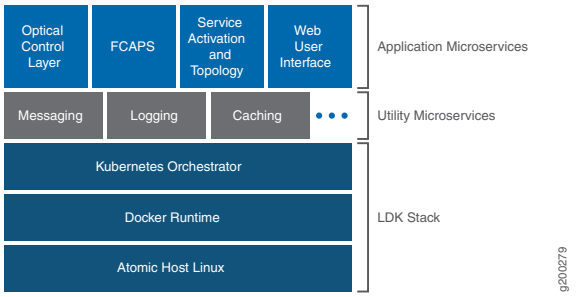
When you deploy the proNX Optical Director, the master schedules the pod instances to run on individual nodes in the cluster. After the pod instances are created, the master continually monitors those instances to ensure they remain in the desired state and can recreate them if necessary.

You are not required to be familiar with Kubernetes in order to install and use the proNX Optical Director.

proNX Optical Director Software Stack

Figure 2 on page 9 shows an abstracted view of the proNX Optical Director software stack:

Figure 2: proNX Optical Director Software Stack



At the bottom is the LDK stack, which provides the host operating system, container runtime, and container orchestration capabilities.

The middle layer consists of a number of common utility microservices that are available for application microservices to use. These include utilities that help microservices communicate with each other, collect logs and metrics, and provide access to cached and persistent storage.

The top layer is the set of applications that define the proNX Optical Director's functionality.

- The Optical Control Layer (OCL) is responsible for controlling the optical links in the network. It processes Optical Telemetry Interface (OTI) messages from the optical devices and programs their internal amplifiers and attenuators to keep the optical links stable. OCL runs autonomously. Users do not interact with OCL directly.
- FCAPS provides the fault, configuration, accounting/administration, performance management, and security functions for the management system. Included within FCAPS are the Device and Operation microservices (not shown).
  - The Device microservice is responsible for retrieving alarms from devices, updating system information, configuring NTP servers, and configuring interfaces and links.
  - The Operation microservice is responsible for providing support for the tasks in the Operations drop-down list (in the Devices tab) such as backup and restore, software upgrades, and retrieving logs and metrics. Also included are discovering and undiscovering devices and retrieving the list of tasks that have been launched.
- Service Activation and Topology is responsible for understanding how the devices in the network are interconnected and for setting up optical services across that network topology. This functionality is provided through the Network microservice (not shown).
- The Web User Interface is the user interface to the proNX Optical Director. For the TCX Series devices, this is the sole user interface to manage the device. There is no command line interface on TCX Series devices.

Network Requirements

The management network that connects the proNX Optical Director to the devices that it manages is called the Data Communication Network (DCN). Connectivity from a managed device to the DCN can be in-band on optical links that carry user traffic or out-of-band on the device's management ports.

Along with the management traffic, the DCN carries the optical control traffic that allows the proNX Optical Director to control the optical links on the devices that it manages. In this role, the proNX Optical Director processes streams of optical link metrics from the devices and constantly adjusts the attenuators and amplifiers within each device to ensure optimal link performance at all times even as optical wavelength services are added and deleted from the network. This real-time control loop requires a continuous and reliable communication path between the proNX Optical Director and each device under control and management. This places strict requirements on the DCN.

- High Availability on page 10
- Server Failure on page 13
- Protocol Port Usage on page 13
- Other Network Requirements on page 15

High Availability

In order to provide the reliability required for the proNX Optical Director to act as an optical link controller, the DCN must be designed for high availability (HA). An HA system is intended for continuous operation and has redundant components and adequate

backup and failover strategies. The DCN must be highly fault tolerant where a single failure does not isolate part of the network. The data center where the proNX Optical Director servers are situated must be hardened and be able to survive power outages and disastrous events.

While HA system design is beyond the scope of this document, the following describes how the proNX Optical Director and the TCX1000 Series devices can connect to an HA network.

The proNX Optical Director installation consists of a cluster of three servers in an HA data center. These servers communicate constantly with each other and should be colocated for low latency to ensure proper operation. Within the cluster is a master node and two regular nodes. The master node provides orchestration and scheduling for the regular nodes. The master node and regular nodes must remain connected to the management network at all times through redundant links. One way of doing this is to create a bonded interface with an active/backup link to provide fault tolerance. In this scenario, all traffic to and from the node goes over the active link. If the active link fails, the backup link becomes active and begins to carry traffic.

Similarly, every managed TCX Series device must have redundant links to the HA network. The TCX Series device can connect to the HA network in multiple ways:

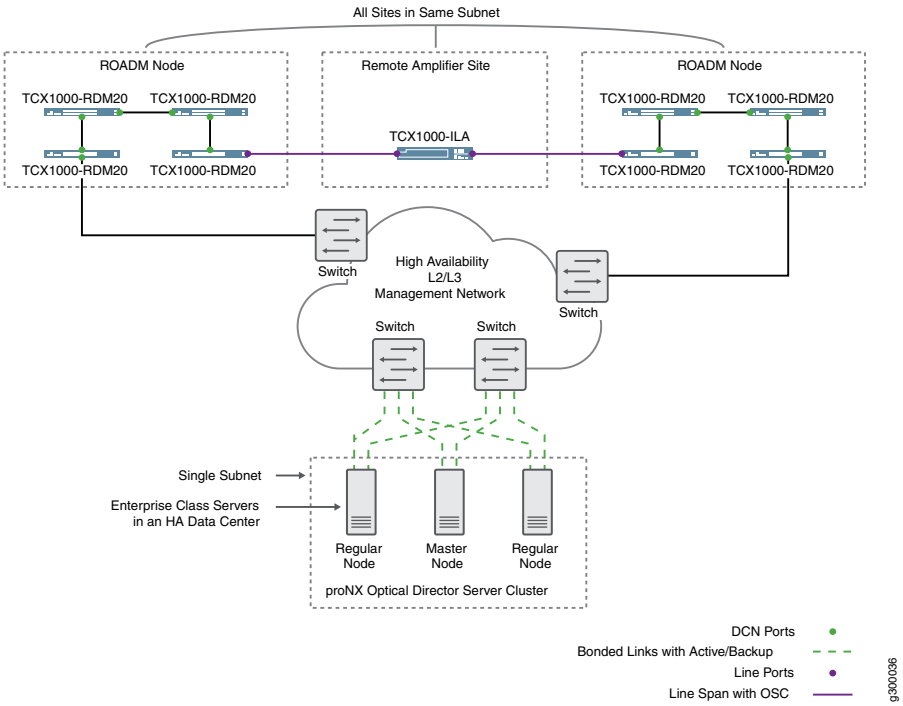
- over its management port(s) - there are two management ports on the TCX1000-RDM20 (DCN0 and DCN1) and one management port on the TCX1000-ILA (MGMT)
- over the optical service channel (OSC) on its line spans - the OSC is a point-to-point channel that allows optical devices to communicate with each other for management and control purposes

Figure 3 on page 12 shows an example of a DCN connecting the proNX Optical Director installation to a pair of ROADM nodes and an amplifier site. Connectivity from the ROADM nodes to the DCN is provided by two TCX1000-RDM20 devices that act as gateways, switching management traffic between the other devices and the external switches.

Within each ROADM node, management connectivity can be implemented by daisy chaining TCX1000-RDM20 devices through their DCN ports. This is not the only way to connect the devices to the management network but this method allows you to reduce the number of ports used on the externally-connected switch. The alternative is to connect both DCN ports on every device to a pair of external switches.

In this example, you can see that each device within the ROADM node and amplifier site has two connections that lead to the DCN. Failure of any one of those connections does not isolate the device. For even higher redundancy, you can configure each device to have more than two connections to the DCN (not shown).

Figure 3: Management Network Connectivity



Internally, the management and OSC ports are connected to an integrated Ethernet switch (not shown). By default, this Ethernet switch supports RSTP to ensure loop-free layer two access. You must enable RSTP on all external Ethernet switches that connect to a TCX Series device. For more information on management connectivity rules and how to configure TCX Series devices for management connectivity, see the *TCX Series Optical Transport System Feature Guide*.

Table 1 on page 12 summarizes the general requirements for the DCN and the proNX Optical Director.

Table 1: Network Requirements Summary

System	Network Requirements
DCN	• The management network must be an HA network. An HA network provides 99.999% availability, supports diverse links, and has no single point of failure. • All TCX Series devices must have redundant connectivity to the HA management network. • TCX Series devices act as layer two devices on the HA management network. Therefore, all TCX Series devices connected together by their management port(s) and/or OSC must be on the same subnet. If the HA management network is a layer three network, then ensure the TCX Series devices at the layer two network boundaries are configured to disable the switching of management traffic on the optical service channels that cross those network boundaries. • Switches that connect to the TCX Series devices must participate in RSTP to prevent forwarding loops. NOTE: TCX1000-RDM20 devices running software releases 2.2 and lower do not support RSTP and do not support management traffic over its OSC.

Table 1: Network Requirements Summary (continued)

System	Network Requirements
proNX Optical Director servers	• The servers running the proNX Optical Director must be enterprise class servers, and the server cluster must be installed in an HA data center that has redundant systems and uninterruptible power supplies (UPS). • Servers in the proNX Optical Director server cluster must reside in the same subnet. • All servers in the proNX Optical Director server cluster must have multiple links to multiple switches in the HA network.

Server Failure

The proNX Optical Director is designed to run in a cluster of three servers in an HA environment. In the unlikely event that a server in the server cluster fails or becomes disconnected from the DCN in this HA environment, the proNX Optical Director continues to run, but possibly in a degraded and vulnerable state. Note that a server failure does not cause any data loss because data is replicated on all three servers in the cluster.

NOTE: **You must address and remedy the failure at your earliest opportunity and before continuing with new provisioning.**

Failure of a single server in the cluster does not affect optical controls, which continue to run for all existing links and services. User traffic continues to be carried over controlled links in the optical network.

Other functionality in the proNX Optical Director, however, might be affected to varying degrees depending on the server that failed. When a server fails or becomes disconnected, the remaining servers make every effort to rebalance their databases, but this is not always possible, leading to degraded functionality including loss of web UI access.

When the failed server comes back online, it will attempt to rejoin the cluster. In most cases, this is done automatically but can take some time as the servers rebalance their databases.

In either situation, if the databases are not rebalanced and the web UI is not accessible after an hour, manual intervention using the kubectl utility might be required.

Protocol Port Usage

Communication paths exist between the user running the browser-based GUI, the proNX Optical Director server cluster, and the devices under management. It is important that these paths be unhindered by any firewalls that exist in the network.

Table 2 on page 14 describes the destination ports used for communication from the managed device (source) to the proNX Optical Director cluster (destination).

Table 2: Device to proNX Optical Director Cluster

Protocol / Destination Port	Description
UDP / 1620	SNMP traps
UDP / 50000	Optical Telemetry Interface (OTI) messages

Table 3 on page 14 describes the destination ports used for communication from the proNX Optical Director cluster (source) to the managed device (destination).

Table 3: proNX Optical Director Cluster to Device

Protocol / Destination Port	Description
TCP / 830	NETCONF
TCP / 1830	IPv6 NETCONF access to the TCX1000-ILA
UDP / 161	SNMP (for managing MX Series or PTX Series routers, QFX Series switches, ACX6360 routers and transponders, and BTI7800 devices)

Table 4 on page 14 and Table 5 on page 15 describe the destination ports used for communication from the user machine where the user is running the browser-based GUI (source) to the proNX Optical Director cluster (destination).

Table 4: User Machine GUI to proNX Optical Director Cluster (Release 2.2 and Lower)

Protocol / Destination Port	Description
TCP / 80	Main user interface
TCP / 3000	Grafana (for analyzing performance monitoring metrics)
TCP / 5601	Kibana (for analyzing logs)
TCP / 8082 TCP / 8084 TCP / 8085	Access to the network, operation, and device microservices
TCP / 8086 TCP / 9200	Communication with various internal modules
TCP / 9090	Prometheus on the master node (for monitoring Kubernetes pods in the proNX Optical Director application) This is optional. See "Prometheus" on page 65.

Table 5: User Machine GUI to proNX Optical Director Cluster (Release 18.4 and Higher)

Protocol / Destination Port	Description
TCP / 443	Main user interface
UDP / 2620	Telemetry metrics

Table 6 on page 15 describes the destination ports used for communication between nodes in a proNX Optical Director cluster. The nodes within a cluster communicate regularly with each other.

Table 6: Communication Between Nodes in a proNX Optical Director Cluster

Protocol / Destination Port	Description
TCP / 443 TCP / 2380 TCP / 5000 TCP / 8086 TCP / 9200 TCP / 10250 to TCP / 10252 TCP / 10255	Communication between various internal modules
UDP / 8472	VXLAN (for the network fabric)
IP Protocol 112	VRRP (for virtual IP address election)

Other Network Requirements

Table 7: Other Network Requirements

Requirement	Description
Internet access	The machine where the user is running the browser-based GUI requires access to the Internet for the display of maps in the Network Map view.
MTU Size	The maximum transmission unit for all devices in the network must be set to 1500 bytes.
UDP packet fragmentation	Optical Telemetry Interface (OTI) messages sent from the managed device to the proNX Optical Director server cluster contain performance monitoring metrics that the proNX Optical Director analyzes in order to control the optical links on the device. These messages typically contain more data than can fit into a single MTU-sized packet, which means that OTI messages might be sent in multiple UDP fragments. In some networks, firewalls are configured to discard UDP packets with the more fragment (MF) bit set. In order to support OTI messages, the network must be configured to allow UDP fragments.

Table 7: Other Network Requirements (continued)

Requirement	Description
Private subnets	By default, some proNX Optical Director software components have internal IP addresses on the following private subnets: • 192.168.248.0/22 • 10.254.0.0/16 Ensure that the management network does not have subnets or IP addresses that overlap or conflict with the above. NOTE: These default subnets are defined in the **<install-dir>/group_vars/all.yml** file on the control machine. If you want to change the default subnets used, edit that file prior to installing the proNX Optical Director software.
IPv6 (release 18.4 and higher)	The proNX Optical Director supports IPv6 addressing with the following restrictions: • The proNX Optical Director software relies on some third party components that support IPv4 addressing only. Therefore, the proNX Optical Director servers must be deployed in a network that supports both IPv4 and IPv6. • A proNX Optical Director IPv4 installation supports the control and management of IPv4 devices only. • A proNX Optical Director IPv6 installation supports the control and management of IPv6 devices only.

Software Licenses

Table 8: proNX Optical Director Software Licenses

SKU	Description
PRONX-OPT-DIR	proNX Optical Director Software License This software license allows you to download and install the proNX Optical Director. You do not require a right-to-use (RTU) license to use the proNX Optical Director to manage TCX1000 Series devices, but you do require RTU licenses to use the proNX Optical Director to manage other devices.
PRONX-OD-RTU-G1	proNX Optical Director RTU License - Group 1 Devices This RTU license allows you to use the proNX Optical Director to manage MX Series routers and BTI7801 and BTI7802 devices.
PRONX-OD-RTU-G2	proNX Optical Director RTU License - Group 2 Devices This RTU license allows you to use the proNX Optical Director to manage PTX Series routers, QFX Series switches, ACX6360 routers and transponders, and BTI7814 devices.
NOTE: All software licenses are perpetual.	

Northbound Interface

The proNX Optical Director supports a northbound RESTCONF interface that is available for higher level management systems to utilize. RESTCONF is an HTTP-based protocol that uses Representational State Transfer (REST) principles to provide a programmatic interface for external systems to access data in YANG modules. This interface allows external systems (for example, OSS or BSS systems) to communicate directly with the main microservices in the proNX Optical Director.

By opening up this northbound interface, the proNX Optical Director enables you to have a tighter integration with your existing systems by allowing you to develop applications to have greater visibility of the network, topology, and devices under management from those systems. For example, you can develop an application on your existing management system to retrieve the optical topology of the network so that you can make informed routing decisions.

The RESTCONF interface is authenticated with username and password. Requests are authorized using JSON web tokens.

You can view detailed API documentation online at the URLs shown in Table 9 on page 17.

Table 9: Accessing Online API Documentation for the Northbound Interface

Microservice	Description	API Documentation URL
Device	The Device microservice provides support for the device layer. This includes retrieving alarms from devices, updating system information, configuring NTP servers, and configuring interfaces and links.	Release 2.2 - http://<server-ip>:8085/docs/index.html Release 18.4 and higher - https://<cluster-name>/device/docs/index.html
Network	The Network microservice provides support for the network layer. This includes providing an overall view of the nodes and topology in the network, as well as the management of optical services.	Release 2.2 - http://<server-ip>:8082/docs/index.html Release 18.4 and higher - https://<cluster-name>/network/docs/index.html
Operation	The Operation microservice provides support for the tasks in the Operations drop-down list (in the Devices tab) such as backup and restore, software upgrades, and retrieving logs and metrics. Also included are discovering and undiscovering devices and retrieving the list of tasks that have been launched.	Release 2.2 - http://<server-ip>:8084/docs/index.html Release 18.4 and higher - https://<cluster-name>/operation/docs/index.html

CHAPTER 2

Installation

Installation Overview

NOTE: The installation procedures require you to be familiar with Linux. If you are not comfortable installing Linux or running Linux commands from the command line, ensure that a craftsperson with Linux administration responsibilities is on hand throughout the installation process.

NOTE: The installation procedures in this document show sample commands and outputs that are current as of the writing of those procedures for the specified operating systems. Because these examples include the installation of third-party tools and software, the commands and outputs might change over time without notice. Be sure to consult the latest third-party documentation where applicable for the most up-to-date installation instructions and examples.

The proNX Optical Director software is installed on a cluster of (typically) three Linux machines. You need to set up each cluster server with the required Linux distribution prior to installing the proNX Optical Director. See "Cluster Server Requirements" on page 20 for the hardware and software requirements for the cluster servers.

You must use an additional computer to download, distribute, and install the required images onto the servers in the cluster. This additional computer is called the control machine. Once you set up the control machine, you can use the supplied scripts to carry out the installation on the cluster without having to work with each cluster member individually.

You can set up a new computer for the control machine or you can use an existing computer if you have one available. See "Control Machine Requirements" on page 22 for the hardware and software requirements for the control machine.

The proNX Optical Director software is supplied as a single gzipped tarball. This tarball includes everything you need to install the proNX Optical Director including the installation scripts and the full set of container images. The proNX Optical Director installation process leverages the use of Ansible scripts to facilitate installation.

Ansible

The proNX Optical Director installation uses Ansible playbooks to set up the cluster. Ansible is a widely used automation engine that automates application management and deployment.

You are not required to be familiar with Ansible in order to install and use the proNX Optical Director.

Cluster Server Requirements

Table 10 on page 20 and Table 11 on page 21 show the hardware and operating system requirements for a single bare metal server in a three-server cluster. Table 12 on page 21 shows the network infrastructure requirements for the cluster server.

NOTE: **Running the proNX Optical Director server software in a VM is not supported in the current release.**

Table 10: Cluster Server Requirements

Attributes		Requirements
Processor	CPU Family	Intel Xeon E5 or better
	Architecture	x86 64-bit
	Minimum Number of Cores	16
	Hyperthreading	Yes
Number of Processors		2

Table 10: Cluster Server Requirements (continued)

Attributes		Requirements
Memory	Minimum Amount	64 GB ECC
	Type	DDR4 Registered DIMMs
	Minimum Performance	2400 MT/s (million transfers per second)
Hard Disk Drive	Type	12 Gbps SAS
	Disk Space	2 x 2 TB
	Controller	RAID (0/1/5) Controller 2 GB NV Cache (Battery Backup Unit)
NIC		Dedicated Quad Port Gigabit Ethernet
Management		Intelligent Platform Management Interface (IPMI)
User Machine Browser (for GUI)		A desktop or laptop computer running one of the following browsers: Mozilla Firefox Google Chrome Apple Safari

Table 11: Cluster Server Operating System Requirements

proNX Optical Director Software Version	Operating System Required
Release 18.4	Atomic Host Linux 7.1808 (CentOS)
Release 2.2	Atomic Host Linux 7.1805 (CentOS)
Release 2.1	Atomic Host Linux 7.1711 (CentOS)

Table 12: Cluster Server Network Infrastructure Requirements

Attributes	Requirements
Networking	All cluster servers must be in the same subnet.
IPv4 DNS Server	The host OS must be configured to use an IPv4 DNS server. This is required regardless of whether you are running an IPv4 or IPv6 proNX Optical Director installation.
IPv6 DNS Server	The host OS must be configured to use an IPv6 DNS server if you are running an IPv6 proNX Optical Director installation.

Table 12: Cluster Server Network Infrastructure Requirements (continued)

Attributes	Requirements
NTP Server	The host OS must be configured to use an NTP server.

[1] See "Installing the Host OS on the Cluster Server" on page 23 for more details.

Control Machine Requirements

Table 13: Control Machine Requirements

Attributes	Requirements
Processor	x86 64-bit
Memory	4 GB or higher
Hard Drive	Minimum 10 GB of free disk space
Operating System	Linux or Mac OS X
NIC	Gigabit Ethernet
Other[1]	Bash version 2 or higher
	Ansible version 2.6.3 or higher
	Python version 2.7 or higher NOTE: Python version 3 is not supported
	Python netaddr module
	Kubernetes kubectl command line tool (optional): • minimum version 1.5 • maximum version 1.6 NOTE: Other versions might work but have not been tested with the server cluster.

[1] See "Setting Up the Control Machine (Examples)" on page 51 for examples on installing these packages.

Installing the Host OS on the Cluster Server

Use this procedure to install the Atomic Host OS onto a bare metal server. The Atomic Host OS is the required operating system for the cluster servers.

1. Download the Atomic Host Installer ISO from the CentOS Atomic Host download site.

   Go to http://cloud.centos.org/centos/7/atomic/images and select the installable ISO image for the release specified in "Cluster Server Requirements" on page 20.

2. Create a DVD or USB drive with that ISO image. You will need to use the appropriate disk imaging tool for the operating system on the machine on which you are creating the drive. You cannot simply copy the image onto the DVD or USB drive.

3. Boot up the target machine with the DVD. You might need to change the BIOS boot-up sequence to boot from the DVD or USB drive.

4. Follow the setup screens to set up Linux as you normally do:
   - Configure the language support and keyboard settings.
   - Specify the installation destination (hard drive) where you want to install the OS.
   - Configure the networking parameters and hostname and enable the Ethernet interface. All cluster members must be on the same subnet and all cluster members must be able to reach the devices in the managed network.
     - Set the IPv4 address, the IPv4 default gateway, and the IPv4 Domain Name Server (DNS). This is required even if you are performing an IPv6 proNX Optical Director installation later.
     - Optionally, set the IPv6 address, the IPv6 default gateway, and the IPv6 Domain Name Server (DNS). This is only required if you are performing an IPv6 proNX Optical Director installation later. IPv6 proNX Optical Director installation is supported starting in release 18.4.
     - Set the hostname. It is required that you set a valid and unique static hostname for each server. A valid hostname consists of a host portion followed by a dot followed by a domain name (for example, server1.mydomain). Ensure the hostname is resolvable (that is, remember to add a DNS entry so that the hostname can resolve to the configured IP address).
     - Enable the Ethernet interface.
   - Configure the date and time settings (including NTP server settings).

NOTE: **Do not override or change any partitioning settings. The proNX Optical Director software installation script automatically configures the partitioning later.**

5. Configure the root password.

6. Reboot the machine when prompted.

7. After the machine reboots, log in as root.

8. Verify that the machine has successfully synchronized with the NTP server(s) that you specified during OS installation. By default, the machine uses **chrony** for NTP server synchronization.

   For example:

```
# chronyc sources -v

210 Number of sources = 1

  .-- Source mode  '^' = server, '=' = peer, '#' = local clock.
 / .- Source state '*' = current synced, '+' = combined , '-' = not combined,
| /   '?' = unreachable, 'x' = time may be in error, '~' = time too variable.
||                                                 .- xxxx [ yyyy ] +/- zzzz
||      Reachability register (octal) -.           |  xxxx = adjusted offset,
||      Log2(Polling interval) --.      |          |  yyyy = measured offset,
||                                \     |          |  zzzz = estimated error.
||                                 |    |           \
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* <ntp-server-1>                1   6    17     5   -253us[-1766us] +/-
60ms
```

NOTE: If you forgot to specify the NTP server(s) during OS installation, edit and specify the NTP server(s) in the **/etc/chrony.conf** file. To make the changes take effect, stop and start the chronyd service (**systemctl stop chronyd.service** followed by **systemctl start chronyd.service**).

   Before proceeding to the next step, ensure at least one NTP server is shown in "*" state.

9. Optionally, set up a user that the control machine installation script can use to log in to this machine. You can delete this user once the installation is complete. You can skip this step if you choose to let the control machine installation script log in as root.

   For example, this sets up a user called **deploy**.

```
# useradd deploy
# passwd deploy
Changing password for user deploy.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

Give this user no-password sudo access (that is, sudo access where no password is prompted for).

To do this, edit the sudoers file:

```
# visudo
```

Add the following line to the file for the **deploy** user:

```
deploy  ALL=(ALL)       NOPASSWD:ALL
```

This line specifies that the **deploy** user is allowed sudo access and that the **deploy** user is not prompted for a password when running sudo commands.

This cluster server is now ready for proNX Optical Director software installation. Repeat this procedure on all servers in the cluster. Ensure that the root password (and the **deploy** username and password if applicable) is the same on all servers.

Upgrading the Host OS on the Cluster Server

Prerequisites

- Ensure all Kubernetes pods are running normally. Issue the **kubectl get pods** command from the control machine or the master node and verify that all Kubernetes pods are READY and have a STATUS of Running.
- The server that you are upgrading has access to the Internet to download the new host OS release.
- You have the common secret that you specified when you ran the proNX Optical Director installation script for the current installation.

NOTE: If you have forgotten the common secret, you can retrieve it by issuing the following commands from the master node:

- Type **kubectl get secret common.secret -o yaml** to retrieve the encoded password.
- Type **echo '<encoded-password>' | base64 --decode** to see the encoded password in clear text.

Use this procedure to upgrade the host operating system on each server in the cluster. Once you start, you must upgrade the host operating system on all three servers. The typical procedure is to upgrade the regular nodes first and the master node last.

NOTE: Although this procedure does not cause any application downtime, it is recommended that you not use the proNX Optical Director to make any configuration changes while this upgrade is taking place.

This procedure shows an upgrade from Atomic Host 7.1805 to 7.1808 as an example. The same steps apply to upgrading from and to other releases.

1. Log in to one of the servers. For example:

```
# ssh 10.228.4.51
```

2. Show the Atomic Host version you are currently running. For example:

```
# atomic host status
State: idle
Deployments:
  ostree://centos-atomic-host:centos-atomic-host/7/x86_64/standard
                   Version: 7.1805 (2018-06-11 11:03:09)
                    Commit:
6275caab694c56515c57f5e5ef923d999e16e8fb1241992d155ff03cfb528261
               GPGSignature: Valid signature by
64E3E7558572B59A319452AAF17E745691BA8335
                  Unlocked: hotfix

  ostree://centos-atomic-host:centos-atomic-host/7/x86_64/standard
                   Version: 7.1805 (2018-06-11 11:03:09)
                    Commit:
6275caab694c56515c57f5e5ef923d999e16e8fb1241992d155ff03cfb528261
               GPGSignature: Valid signature by
64E3E7558572B59A319452AAF17E745691BA8335
[root@server1 ~]#
```

3. Upgrade the host OS. For example:

```
# atomic host deploy 7.1808
Resolving version '7.1808'
2 metadata, 0 content objects fetched; 18 KiB transferred in 1 seconds

768 metadata, 4542 content objects fetched; 334286 KiB transferred in 161
seconds
Copying /etc changes: 26 modified, 13 removed, 211 added
Transaction complete; bootconfig swap: yes deployment count change: 0
Upgraded:
  NetworkManager 1:1.10.2-14.el7_5 -> 1:1.10.2-16.el7_5
  NetworkManager-libnm 1:1.10.2-14.el7_5 -> 1:1.10.2-16.el7_5
  atomic 1:1.22.1-3.git2fd0860.el7 -> 1:1.22.1-22.git5a342e3.el7
  atomic-registries 1:1.22.1-3.git2fd0860.el7 -> 1:1.22.1-22.git5a342e3.el7

  audit-libs 2.8.1-3.el7 -> 2.8.1-3.el7_5.1
```

```
  audit-libs-python 2.8.1-3.el7 -> 2.8.1-3.el7_5.1


<trimmed>

Run "systemctl reboot" to start a reboot
```

4. Reboot the server.

```
# systemctl reboot
```

5. After the server has rebooted, log back in to the server to verify that the server is running the new version of the OS. For example:

```
# atomic host status
State: idle; auto updates disabled
Deployments:
  ostree://centos-atomic-host:centos-atomic-host/7/x86_64/standard
                   Version: 7.1808 (2018-09-12 16:35:46)
                    Commit:
b62513420243def54db7d6c36a79d8f0eda94c5ac410c1f64ee6e0b44e5dca99
               GPGSignature: Valid signature by
64E3E7558572B59A319452AAF17E745691BA8335

  ostree://centos-atomic-host:centos-atomic-host/7/x86_64/standard
                   Version: 7.1805 (2018-06-11 11:03:09)
                    Commit:
6275caab694c56515c57f5e5ef923d999e16e8fb1241992d155ff03cfb528261
               GPGSignature: Valid signature by
64E3E7558572B59A319452AAF17E745691BA8335
                  Unlocked: hotfix
```

6. From the control machine, verify that the Kubernetes pods are READY and have a STATUS of Running and the databases have been rebalanced. You should perform this step from the control machine because the kubectl utility has not yet been reinstalled on the node with the upgraded host operating system.

   - Check the application database pods. For example:

```
# kubectl get pods | grep joc-db
NAME                              READY     STATUS    RESTARTS   AGE
joc-db-blue-99052229              1/1       Running   1          6h
joc-db-green-1848954212-hq0vn     1/1       Running   1          6h
joc-db-red-29840614               1/1       Running   1          6h
```

   - Check to make sure each database has been rebalanced. Rebalancing can take an hour or longer. If the rebalance is complete, you will see a status of notRunning. For example:

```
# kubectl exec -it joc-db-green-1848954212-hq0vn -- couchbase-cli
rebalance-status -c localhost -u Administrator -p <common-secret>
{
  "status": "notRunning",
```

```
    "msg": "Rebalance is not running",
    "details": {}
  }
```

where **<common-secret>** is the common secret you specified when you installed the current version of the proNX Optical Director software.

Do not proceed until you have confirmed that the Kubernetes pods are READY and have a STATUS of Running and the databases have been rebalanced.

7. Repeat this procedure on the next server until all the servers are upgraded.

Downloading the proNX Optical Director Software

Use this procedure to download the proNX Optical Director software package from the Juniper Networks software download page. proNX Optical Director software is provided as a gzipped-compressed tar archive. You can use any machine (including the control machine) to download the software.

NOTE: To make the examples in this procedure generic to all releases, the example output does not show release numbers in the filenames.

Prerequisites

- A computer running Linux, Mac OS X, or Windows
- Internet access capable of transferring large files
- A user login account on www.juniper.net

1. Use your browser to go to https://www.juniper.net/support/downloads.
2. Start typing **proNX Optical Director** in the product name box and click **proNX Optical Director** as the completion suggestions are provided.
3. Select the desired software **VERSION** from the drop-down list.
4. Scroll down to the Application Media section.
5. Click the gzipped archive (tgz) that you want to download.
6. If you are not already logged in, the browser displays a login screen.
   a. Enter your **User ID** and **Password**.
   b. Read through the EULA, select **I Agree**, and click **Proceed** if you agree with the terms of the EULA.
7. Select the option to download to the **localhost**.

8. If your operating system prompts you to save or open the file, choose the **Save File** option or equivalent.

   Depending on the speed of your Internet connection, the download might take 30 minutes or longer.

   The proNX Optical Director software package is a gzipped-compressed archive. The filename is in the format **pod-*<version>*.tgz**.

9. On your browser, go back to the download page and click **Checksums** to view the checksum of the corresponding download.

   The MD5 and multiple SHA variants of the checksums appear in a pop-up window.

10. Verify the checksum of the downloaded image with the expected checksum. The examples below verify the MD5 checksum.

    a. On a Linux command line, for example:

    ```
    $ ls
    pod-<version>.tgz
    $ md5sum pod-<version>.tgz
    file-checksum pod-<version>.tgz
    ```

    b. In a Mac OS X terminal window, for example:

    ```
    $ ls
    pod-<version>.tgz
    $ md5 pod-<version>.tgz
    file-checksum pod-<version>.tgz
    ```

    c. On a Windows command line, for example:

    ```
    C:\Users\Public\Downloads>dir pod-<version>.tgz
     Volume in drive C is Windows
     Volume Serial Number is D095-F11F

     Directory of C:\Users\Public\Downloads

    12/22/2017  12:21 PM     2,037,637,148 pod-<version>.tgz
                   1 File(s)  2,037,637,148 bytes
                   0 Dir(s)  52,650,622,976 bytes free

    C:\Users\Public\Downloads>certutil -hashfile pod-<version>.tgz MD5
    MD5 hash of file pod-<version>.tgz:
    file-checksum
    CertUtil: -hashfile command completed successfully.
    ```

Compare the *file-checksum* from the output of the command with the checksum displayed in step 9. If the checksums do not match, download the image again and re-verify the checksum.

11. Copy the downloaded file to the directory on the control machine where you want to run the installation scripts.

    If you have set up the control machine and the cluster servers, you can now proceed to install the proNX Optical Director software.

Installing the proNX Optical Director Software (Release 2.2)

Prerequisites

- The control machine is set up. See "Setting Up the Control Machine (Examples)" on page 51.
- You have copied the proNX Optical Director software package into the directory on the control machine where you want to run the installation scripts. This directory is called the **<install-dir>** in the examples in this procedure. See "Downloading the proNX Optical Director Software" on page 28.
- The cluster servers are set up with the proper, freshly-installed operating system and are accessible by the control machine. See "Installing the Host OS on the Cluster Server" on page 23.

NOTE: The operating system must be freshly installed. If you are currently running an existing version of the proNX Optical Director software and you would like to install the latest version, you must reinstall the operating system first.

NOTE: If you are currently running an existing version of the proNX Optical Director software, close all browser windows to that proNX Optical Director before you install the new version.

Use this procedure on the control machine to perform a fresh installation of the proNX Optical Director software on all the servers in the cluster. This procedure applies to the installation of release 2.2 only. You do not require Internet access to use this procedure.

NOTE: This procedure can take two to three hours or longer since it takes time for the servers to synchronize their databases.

NOTE: To make the examples in this procedure generic to all releases, the example output does not show release numbers in the filenames.

1. Untar and uncompress the downloaded archive, for example:

```
# cd <install-dir>
# ls
pod-<version>.tgz

# tar -xzvf pod-<version>.tgz

...

# ls
pod-installer-<version>      pod-<version>.tgz
```

2. Specify the cluster members.

   Use a text editor to edit the **<install-dir>/pod-installer-<version>/inventory** file and specify one of the cluster servers as the master node and all of the cluster servers (including the master) as cluster nodes. It does not matter which server you specify as the master.

```
# vi pod-installer-<version>/inventory
```

   The inventory file contains a [masters] section and a [nodes] section. Specify the master node in the [masters] section and all nodes (including the master node) in the [nodes] section.

   For example, this sets up a three-node cluster with master node at 10.228.4.241 and the other cluster members at 10.228.4.106 and 10.228.4.51. These three nodes must have the required Atomic Host OS installed and must be on the same subnet.

```
[masters]
10.228.4.241

[nodes]
10.228.4.241
10.228.4.106
10.228.4.51
```

NOTE: Although the above example uses IP addresses, it is recommended that you use resolvable hostnames instead.

3. Install the software. The supplied script installs the software on all cluster servers. You do not need to install the software on each server individually.

   When you install the software, you have to specify a virtual IP address for the cluster. The virtual IP address is the IP address that users and devices use to communicate with the proNX Optical Director. It is virtual in that the IP address is not permanently associated with an individual server. By decoupling the IP address from the hardware, users and devices have a consistent IP address to use, regardless of which actual server is handling the communication, making it easier to handle individual server failures.

   - To install the software with a virtual IP address:

     ```
     # cd <install_dir>/pod-installer-<version>
     # ./scripts/install.sh
     ```

     NOTE: The installation script must execute in a bash shell. In most cases, this is done automatically as the script itself directs the current shell to run the script in a bash shell. In the event that the script fails with a syntax error, rerun the script by explicitly specifying the bash shell. For example:

     ```
     # bash scripts/install.sh
     ```

   You are prompted for the following:

   - A username and password to log in to the cluster servers. This user must be root or a user with no-password sudo access. See "Installing the Host OS on the Cluster Server" on page 23 for more information.
   - The virtual IP address to use. This virtual IP address must be on the same subnet as the cluster servers. The example below uses virtual IP address 10.228.4.151.
   - A shared secret to be used by the proNX Optical Director to secure internal data. This secret is used internally by Kubernetes pods and should be retained for advanced debugging purposes. This secret is encoded as a Kubernetes secret (and can be retrieved using standard kubectl commands after installation if the secret is accidentally lost).

     NOTE: The control machine does not store the username, password, or shared secret.

   When the script finishes, the command outputs configuration information that you can use to configure the kubectl configuration file on the control machine. The

configuration can be found between the [BEGIN] and [END] tags in the output of the install command.

Here is the command and an example of the script output:

```
# ./scripts/install.sh

Logging to <install_dir>/pod-installer-<version>/logs/deploy-cluster.log

=INVENTORY=
MASTER: 10.228.4.241
NODES:  10.228.4.241  10.228.4.106 10.228.4.51

Checking dependencies...                          [COMPLETE]
Collecting environment information...             [PENDING]

Please enter an ansible_user: [root]
Please enter a password for the "root" user:
Please enter a virtual IPv4 address for the cluster: 10.228.4.151
Please enter a common application secret:
Please confirm the common application secret:

Collecting environment information...             [COMPLETE]
Validating cluster connectivity...                [COMPLETE]
Logging cluster details...                        [COMPLETE]
Executing pre-installation checks...              [COMPLETE]
Preparing cluster installation...                 [COMPLETE]
Updating cluster install settings...              [COMPLETE]
Installing cluster...                             [COMPLETE]
Labelling cluster nodes...                        [COMPLETE]
Installing cluster registry...                    [COMPLETE]
Installing cluster addons...                      [COMPLETE]
Executing post-installation tasks...              [COMPLETE]
Installing cluster applications...                [COMPLETE]
Executing post-installation node tasks...         [COMPLETE]
Echoing kubectl config...                         [PENDING]


[BEGIN]
<configuration>
[END]

Echoing kubectl config...                         [COMPLETE]
Cluster installation...                           [COMPLETE]
```

NOTE: If the installation script returns an error, ensure that the control machine has been set up correctly.

The installation log file is located at **<install-dir>/pod-installer-<version>/logs/deploy-cluster.log**.

4. Optionally, create and populate the kubectl configuration file.

   The kubectl configuration file is required by the kubectl utility to manage the nodes in the cluster. You have a choice of running the kubectl utility on the following:

- on the master node (automatically set up as part of the installation)
- on the control machine (if you have the kubectl utility set up)
- on another machine (for example, on a machine that you currently use to manage your other kubernetes installations)

If you only want to run the kubectl utility on the master node, then you can skip this step. If you want to run the kubectl utility on either the control machine or on another machine, then you will need to create or modify the kubectl configuration file on that machine.

a. Create or edit the configuration file on the machine where you are running the kubectl utility. You might need to create the **.kube** directory and the **config** file if one or both do not exist. This example shows **.kube** in the home directory. By default, kubectl looks for the **config** file in the **~/.kube** directory.

```
# cd ~/.kube
# vi config
```

b. Copy and paste the output of the install.sh command into this file. The text to copy is between the [BEGIN] and [END] tags. Do not copy the [BEGIN] and [END] tags themselves. If you are modifying an existing config file that is being used to manage other kubernetes installations, then append the output of the install.sh command to the end of the existing config file.

c. Save the file.

5. Verify that the installation is successful. Perform this step from the master node or from the machine where you modified the config file in step 4.

On the master node, the kubectl utility is automatically set up in the path so you can issue the command from any directory. The following command can be used to display the state of all pods on all nodes.

```
# kubectl get pods -o wide
```

```
NAME                READY STATUS   RESTARTS AGE IP               NODE
joc-control-3647293 0/1   Running  4        4m  192.168.250.5    10.228.4.123
joc-db-blue-9905222 0/1   Running  0        4m  192.168.250.6    10.228.4.123
joc-db-green-196300 0/1   Running  0        4m  192.168.251.4    10.228.4.140
joc-db-red-29840614 0/1   Running  0        4m  192.168.249.8    10.228.4.223
joc-device-35lbs    1/1   Running  1        4m  192.168.251.5    10.228.4.140
joc-device-nf66j    1/1   Running  1        4m  192.168.250.9    10.228.4.123
joc-ingress-789qv   1/1   Running  0        4m  192.168.249.9    10.228.4.223
joc-ingress-d3533   1/1   Running  0        4m  192.168.250.8    10.228.4.123
joc-ingress-g6vnd   1/1   Running  0        4m  192.168.251.6    10.228.4.140
joc-messaging-30354 1/1   Running  0        4m  192.168.250.7    10.228.4.123
joc-network-4160334 1/1   Running  0        4m  192.168.251.7    10.228.4.140
joc-operation-33548 1/1   Running  0        4m  192.168.250.10   10.228.4.123
joc-registry-151109 1/1   Running  0        4m  192.168.249.5    10.228.4.223
joc-userinterface-3 0/1   Pending  0        4m  <none>           <none>
joc-vip-2wbh0       1/1   Running  0        4m  10.228.4.123     10.228.4.123
joc-vip-9zbf1       1/1   Running  0        4m  10.228.4.140     10.228.4.140
joc-vip-ql9hh       1/1   Running  1        4m  10.228.4.223     10.228.4.223
```

On other machines, if the kubectl utility is not set up in the path, then you will need to issue the kubectl commands in the directory where you installed the kubectl utility, for example:

```
# ls
kubectl

# ./kubectl get pods -o wide
```

NOTE: If you issue this command on other machines and the command returns an error or prompts you for a username and password, verify that you have copied the install.sh output to the kubectl config file faithfully.

As the proNX Optical Director starts up, you will see the STATUS of some Kubernetes pods cycle between Running and other values. This is normal. A Kubernetes pod is operational when it has a STATUS of Running and a READY state of 1/1 (number of containers in the pod that are ready / total number of containers in the pod). The installation is complete once all the Kubernetes pods become operational.

You have now completed the proNX Optical Director installation. You can connect to the user interface by pointing your browser to **http**://<ip_address> where <ip_address> is the virtual IP address that you configured.

NOTE: If you did not remember to close all browser windows to the previous version of the proNX Optical Director before you installed this version of the proNX Optical Director, perform a browser window refresh on each open browser window.

Installing the proNX Optical Director Software (Release 18.4 and Higher)

Prerequisites

- The control machine is set up. See "Setting Up the Control Machine (Examples)" on page 51.
- You have copied the proNX Optical Director software package into the directory on the control machine where you want to run the installation scripts. This directory is called the **<install-dir>** in the examples in this procedure. See "Downloading the proNX Optical Director Software" on page 28.
- The cluster servers are set up with the proper, freshly-installed operating system and are accessible by the control machine. See "Installing the Host OS on the Cluster Server" on page 23.
- You have created or have been assigned a trusted TLS certificate for the cluster.

NOTE: If you are currently running an existing version of the proNX Optical Director software, close all browser windows to that proNX Optical Director before you install the new version.

Use this procedure on the control machine to perform a fresh installation of the proNX Optical Director software on all the servers in the cluster. This procedure applies to the installation of release 18.4 or higher. You do not require Internet access to use this procedure. Both IPv4 and IPv6 installations are covered.

If you are running an existing version of the proNX Optical Director software and would like to upgrade, see "Upgrading the proNX Optical Director Software (Release 18.4 and Higher)" on page 44. Upgrading preserves proNX Optical Director settings and runtime information.

NOTE: This procedure can take two to three hours or longer since it takes time for the servers to synchronize their databases.

NOTE: To make the examples in this procedure generic to all releases, the example output does not show release numbers in the filenames.

1. Untar and uncompress the downloaded archive, for example:

```
# cd <install-dir>
# ls
pod-<version>.tgz

# tar -xzvf pod-<version>.tgz

...

# ls
pod-installer-<version>      pod-<version>.tgz
```

2. Specify the cluster members.

   Use a text editor to edit the **<install-dir>/pod-installer-<version>/inventory** file and specify one of the cluster servers as the master node and all of the cluster servers (including the master) as cluster nodes. It does not matter which server you specify as the master.

```
# vi pod-installer-<version>/inventory
```

   The inventory file contains a [masters] section and a [nodes] section. Specify the master node in the [masters] section and all nodes (including the master node) in the [nodes] section.

NOTE: You must only use IPv4 addresses or hostnames that resolve to IPv4 addresses in this file even if you are planning on performing an IPv6 installation. The proNX Optical Director installation relies on some third-party packages that only support IPv4.

For example, this sets up a three-node cluster with master node at 10.228.4.241 and the other cluster members at 10.228.4.106 and 10.228.4.51. These three nodes must have the required Atomic Host OS installed and must be on the same subnet.

```
[masters]
10.228.4.241

[nodes]
10.228.4.241
10.228.4.106
10.228.4.51
```

NOTE: Although the above example uses IP addresses, it is recommended that you use resolvable hostnames instead.

3. Install a valid and trusted TLS certificate and associated private key for the cluster in **<install_dir>/pod-installer-<version>/certs**. This TLS certificate will be used by the client browser to authenticate the proNX Optical Director server when you later access the proNX Optical Director UI.

   This example shows a certificate and associated private key called pod-cluster.crt and pod-cluster.key respectively.

   ```
   # ls <install_dir>/pod-installer-<version>/certs
   pod-cluster.crt  pod-cluster.key
   ```

   NOTE: You must have exactly one certificate in this directory. If you have no certificate or more than one certificate, the installation will fail.

   If you do not have a TLS certificate, you can create one for testing purposes, as follows:

   ```
   # cd <install_dir>/pod-installer-<version>/certs
   # openssl req -newkey rsa:2048 -nodes -keyout pod-cluster.key -subj
   "/C=CA/ST=ON/O=Juniper/CN=pod-cluster.mydomain" -x509 -days 365 -out
   pod-cluster.crt
   ```

   This certificate will allow you to install and try out the proNX Optical Director, but because this certificate has not been signed by a trusted Certificate Authority, you will receive security warnings when trying to access the proNX Optical Director UI.

   In this example, the certificate is created with a Common Name (CN) of pod-cluster.mydomain. This is the resolvable TLS hostname that you will use to access the proNX Optical Director UI.

NOTE: The certificate must contain a Common Name and not a Subject Alternative Name (SAN) or IP address.

4. Install the software. The supplied script installs the software on all cluster servers. You do not need to install the software on each server individually.

   When you install the software, you have to specify a virtual IP address for the cluster. The virtual IP address is the IP address that browsers and devices use to communicate with the proNX Optical Director. It is virtual in that the IP address is not permanently associated with an individual server. By decoupling the IP address from the hardware, users and devices have a consistent IP address to use, regardless of which actual server is handling the communication, making it easier to handle individual server failures.

   - To perform an IPv4 installation:

     ```
     # cd <install_dir>/pod-installer-<version>
     # ./scripts/install.sh
     ```

   - To perform an IPv6 installation:

     ```
     # cd <install_dir>/pod-installer-<version>
     # ./scripts/install.sh -6
     ```

   NOTE: The installation script must execute in a bash shell. In most cases, this is done automatically as the script itself directs the current shell to run the script in a bash shell. In the event that the script fails with a syntax error, rerun the script by explicitly specifying the bash shell. For example:

   ```
   # bash scripts/install.sh
   ```

   You are prompted for the following:

   - Confirmation that the TLS hostname parsed from the discovered certificate is expected. Note that this hostname must be resolvable to the virtual IP address that you provision in this step. In this example, this requires that you add a DNS entry to map pod-cluster.mydomain to the virtual IP address 10.228.4.151.
   - A username and password to log in to the cluster servers. This user must be root or a user with no-password sudo access. See "Installing the Host OS on the Cluster Server" on page 23 for more information.
   - The virtual IPv4 address to use. This virtual IP address must be on the same subnet as the cluster servers. The IPv4 example output below uses virtual IP address 10.228.4.151.
   - A shared secret to be used by the proNX Optical Director to secure internal data. This common application secret is used internally by Kubernetes pods and should be retained for advanced debugging purposes. This secret is encoded as a

Kubernetes secret (and can be retrieved using standard kubectl commands after installation if the secret is accidentally lost).

NOTE: The control machine does not store the username, password, or shared secret.

If you are performing an IPv6 installation, you are prompted for the following additional information:

- The virtual IPv6 address to use. This virtual IPv6 address must be on the same subnet as the cluster servers. The IPv6 example output below uses virtual IP address 2001:DB8:1:1::254.
- An IPv6 Docker network for each server. The Docker IPv6 network must be at least /80 in size. The IPv6 example output below uses IPv6 subnets 2001:DB8:2:1:241::/80, 2001:DB8:2:1:106::/80, and 2001:DB8:2:1:51::/80.

NOTE: These IPv6 Docker subnets must be externally routable. Ensure that you set up static routes on the gateway router for these three subnets via their respective interface addresses. For example, on a Juniper Networks router, in the routing-options branch:

```
rib inet6.0 {
   static {
       route 2001:DB8:2:1:241::/80 next-hop 2001:DB8:1:1::241;
       route 2001:DB8:2:1:106::/80 next-hop 2001:DB8:1:1::106;
       route 2001:DB8:2:1:51::/80  next-hop 2001:DB8:1:1::51;
   }
}
```

where the next-hop addresses are the IPv6 addresses that you set up on the respective cluster servers when you installed the OS

When the script finishes, the command outputs configuration information that you can use to configure the kubectl configuration file on the control machine. The configuration can be found between the [BEGIN] and [END] tags in the output of the install command.

Here is an example of the script output for an IPv4 installation:

```
# ./scripts/install.sh

Logging to <install_dir>/pod-installer-<version>/logs/deploy-cluster.log

[INFO] Long running tasks will be annotated with [*]

=INVENTORY=
MASTER: 10.228.4.241
NODES:  10.228.4.241  10.228.4.106 10.228.4.51
```

```
Checking dependencies                                [COMPLETE]
Checking TLS certificate and key                     [COMPLETE]


Cluster UI based on certificate will be: https://pod-cluster.mydomain

NOTE: The cluster UI will NOT work on any other URL!!!

Press enter Y to confirm the cluster URL: y

Collecting environment information                   [PENDING]

Please enter the remote username: [deploy] root
Please enter a password for the "root" user:
Please enter a virtual IPv4 address for the cluster: 10.228.4.151
Please enter a common application secret:
Please confirm the common application secret:

Collecting environment information                   [COMPLETE]
Validating cluster connectivity                      [COMPLETE]
Logging cluster details                              [COMPLETE]
Executing pre-installation checks                    [COMPLETE]
Preparing cluster nodes                              [COMPLETE]
Transferring node images to cluster nodes [*]        [COMPLETE]
Transferring infra images to cluster nodes [*]       [COMPLETE]
Preparing cluster master installation                [COMPLETE]
Transferring infra images to cluster master [*]      [COMPLETE]
Updating cluster installation settings               [COMPLETE]
Installing cluster                                   [COMPLETE]
Updating cluster nodes                               [COMPLETE]
Labelling cluster nodes                              [COMPLETE]
Installing in-cluster registry                       [COMPLETE]
Transferring images to in-cluster registry [*]       [COMPLETE]
Installing cluster addons                            [COMPLETE]
Executing TLS configuration tasks                    [COMPLETE]
Executing post-installation tasks                    [COMPLETE]
Installing cluster applications                      [COMPLETE]
Clustering and replicating DB [*]                    [COMPLETE]
Starting cluster applications                        [COMPLETE]
Executing post-installation node tasks               [COMPLETE]
Echoing kubectl config                               [PENDING]


[BEGIN]
<configuration>
[END]

Echoing kubectl config...                            [COMPLETE]
Cluster installation...                              [COMPLETE]
```

Here is an example of the script output for an IPv6 installation:

```
# ./scripts/install.sh -6

Logging to <install_dir>/pod-installer-<version>/logs/deploy-cluster.log

[INFO] Long running tasks will be annotated with [*]

=INVENTORY=
MASTER: 10.228.4.241
```

```
NODES:  10.228.4.241  10.228.4.106 10.228.4.51

Checking dependencies                              [COMPLETE]
Checking TLS certificate and key                   [COMPLETE]


Cluster UI based on certificate will be: https://pod-cluster.mydomain

NOTE: The cluster UI will NOT work on any other URL!!!

Press enter Y to confirm the cluster URL: y

Collecting environment information                 [PENDING]

Please enter the remote username: [deploy] root
Please enter a password for the "root" user:
Please enter a virtual IPv4 address for the cluster: 10.228.4.151
Please enter a virtual IPv6 address for the cluster: 2001:DB8:1:1::254
Please enter an IPv6 docker network (CIDR notation) for node 10.228.4.241:
 2001:DB8:2:1:241::/80
Please enter an IPv6 docker network (CIDR notation) for node 10.228.4.106:
 2001:DB8:2:1:106::/80
Please enter an IPv6 docker network (CIDR notation) for node 10.228.4.51:
 2001:DB8:2:1:51::/80
Please enter a common application secret:
Please confirm the common application secret:

Collecting environment information                 [COMPLETE]
Validating cluster connectivity                    [COMPLETE]
Logging cluster details                            [COMPLETE]
Executing pre-installation checks                  [COMPLETE]
Preparing cluster nodes                            [COMPLETE]
Transferring node images to cluster nodes [*]      [COMPLETE]
Transferring infra images to cluster nodes [*]     [COMPLETE]
Preparing cluster master installation              [COMPLETE]
Transferring infra images to cluster master [*]    [COMPLETE]
Updating cluster installation settings             [COMPLETE]
Installing cluster                                 [COMPLETE]
Updating cluster nodes                             [COMPLETE]
Labelling cluster nodes                            [COMPLETE]
Installing in-cluster registry                     [COMPLETE]
Transferring images to in-cluster registry [*]     [COMPLETE]
Installing cluster addons                          [COMPLETE]
Executing TLS configuration tasks                  [COMPLETE]
Executing post-installation tasks                  [COMPLETE]
Installing cluster applications                    [COMPLETE]
Clustering and replicating DB [*]                  [COMPLETE]
Starting cluster applications                      [COMPLETE]
Executing post-installation node tasks             [COMPLETE]
Echoing kubectl config                             [PENDING]


[BEGIN]
<configuration>
[END]

Echoing kubectl config...                          [COMPLETE]
Cluster installation...                            [COMPLETE]
```

NOTE: If the installation script returns an error, ensure that the control machine has been set up correctly.

The installation log file is located at **<install-dir>/pod-installer-<version>/logs/deploy-cluster.log**.

5. Optionally, create and populate the kubectl configuration file.

   The kubectl configuration file is required by the kubectl utility to manage the nodes in the cluster. You have a choice of running the kubectl utility on the following:

   - on the master node (automatically set up as part of the installation)
   - on the control machine (if you have the kubectl utility set up)
   - on another machine (for example, on a machine that you currently use to manage your other kubernetes installations)

   If you only want to run the kubectl utility on the master node, then you can skip this step. If you want to run the kubectl utility on either the control machine or on another machine, then you will need to create or modify the kubectl configuration file on that machine.

   a. Create or edit the configuration file on the machine where you are running the kubectl utility. You might need to create the **.kube** directory and the **config** file if one or both do not exist. This example shows **.kube** in the home directory. By default, kubectl looks for the **config** file in the **~/.kube** directory.

      ```
      # cd ~/.kube
      # vi config
      ```

   b. Copy and paste the output of the install.sh command into this file. The text to copy is between the [BEGIN] and [END] tags. Do not copy the [BEGIN] and [END] tags themselves. If you are modifying an existing config file that is being used to manage other kubernetes installations, then append the output of the install.sh command to the end of the existing config file.

   c. Save the file.

6. Verify that the installation is successful. Perform this step from the master node or from the machine where you modified the config file in step 4.

   On the master node, the kubectl utility is automatically set up in the path so you can issue the command from any directory. The following command can be used to display the state of all pods on all nodes.

   ```
   # kubectl get pods -o wide
   ```

   ```
   NAME                READY STATUS   RESTARTS AGE IP               NODE
   joc-control-3647293 0/1   Running  4        4m  192.168.250.5    10.228.4.123
   joc-db-blue-9905222 0/1   Running  0        4m  192.168.250.6    10.228.4.123
   joc-db-green-196300 0/1   Running  0        4m  192.168.251.4    10.228.4.140
   joc-db-red-29840614 0/1   Running  0        4m  192.168.249.8    10.228.4.223
   ```

```
joc-device-35lbs    1/1   Running  1        4m  192.168.251.5    10.228.4.140
joc-device-nf66j    1/1   Running  1        4m  192.168.250.9    10.228.4.123
joc-ingress-789qv   1/1   Running  0        4m  192.168.249.9    10.228.4.223
joc-ingress-d3533   1/1   Running  0        4m  192.168.250.8    10.228.4.123
joc-ingress-g6vnd   1/1   Running  0        4m  192.168.251.6    10.228.4.140
joc-messaging-30354 1/1   Running  0        4m  192.168.250.7    10.228.4.123
joc-network-4160334 1/1   Running  0        4m  192.168.251.7    10.228.4.140
joc-operation-33548 1/1   Running  0        4m  192.168.250.10   10.228.4.123
joc-registry-151109 1/1   Running  0        4m  192.168.249.5    10.228.4.223
joc-userinterface-3 0/1   Pending  0        4m  <none>           <none>
joc-vip-2wbh0       1/1   Running  0        4m  10.228.4.123     10.228.4.123
joc-vip-9zbf1       1/1   Running  0        4m  10.228.4.140     10.228.4.140
joc-vip-ql9hh       1/1   Running  1        4m  10.228.4.223     10.228.4.223
```

On other machines, if the kubectl utility is not set up in the path, then you will need to issue the kubectl commands in the directory where you installed the kubectl utility, for example:

```
# ls
kubectl

# ./kubectl get pods -o wide
```

NOTE: If you issue this command on other machines and the command returns an error or prompts you for a username and password, verify that you have copied the install.sh output to the kubectl config file faithfully.

As the proNX Optical Director starts up, you will see the STATUS of some Kubernetes pods cycle between Running and other values. This is normal. A Kubernetes pod is operational when it has a STATUS of Running and a READY state of 1/1 (number of containers in the pod that are ready / total number of containers in the pod). The installation is complete once all the Kubernetes pods become operational.

7. If you are installing or upgrading to proNX Optical Director Release 18.4, create the secondary database indexes. This is required to provide increased redundancy in cases of server failure. If you are installing or upgrading to release 19.1 or higher, you are not required to perform this step because this step is automatically performed in those releases.

   To create the secondary database indexes, log in to the master node and run the following commands:

NOTE: You cannot run these commands from the control machine.

```
# curl -s http://Administrator:<common-secret>@$(kubectl get pods
--no-headers -o jsonpath='{.items[*].status.podIP}' -l
joc-app=joc-db-green):8093/query/service -d 'statement=create index
idx_auth_green on
operation((meta().id),(meta().cas),login,_class,`password`)'
```

```
# curl -s http://Administrator:<common-secret>@$(kubectl get pods
--no-headers -o jsonpath='{.items[*].status.podIP}' -l
joc-app=joc-db-red):8093/query/service -d 'statement=create index
idx_auth_red on operation((meta().id),(meta().cas),login,_class,`password`)'

# curl -s http://Administrator:<common-secret>@$(kubectl get pods
--no-headers -o jsonpath='{.items[*].status.podIP}' -l
joc-app=joc-db-blue):8093/query/service -d 'statement=create index
idx_auth_blue on operation((meta().id),(meta().cas),login,_class,`password`)'
```

where **<common-secret>** is the common secret you specified as part of the installation in step 4.

You have now completed the proNX Optical Director installation. You can connect to the user interface by pointing your browser to https://<cluster-name> where <cluster-name> is the Common Name used when creating the cluster certificate. In the example certificate above, you connect to the UI using https://pod-cluster.mydomain.

NOTE: If you did not remember to close all browser windows to the previous version of the proNX Optical Director before you installed this version of the proNX Optical Director, perform a browser window refresh on each open browser window.

Related Documentation

-

Upgrading the proNX Optical Director Software (Release 18.4 and Higher)

Prerequisites

- The control machine is set up. For convenience, you should use the same control machine that you used to install the existing version of the software.
- The cluster server host operating system has been upgraded to the version required by the new proNX Optical Director software release. For information on how to do this, see "Upgrading the Host OS on the Cluster Server" on page 25.
- You have copied the proNX Optical Director software package into the directory on the control machine where you want to run the installation scripts. This directory is called the **<install-dir>** in the examples in this procedure.
- The existing proNX Optical Director installation is fully operational with all cluster servers running.
- You have created or have been assigned a trusted TLS certificate for the cluster.
- You have the common secret that you specified when you ran the proNX Optical Director installation script for the installation that you are upgrading from.

NOTE: If you have forgotten the common secret, you can retrieve it by issuing the following commands from the master node:

- Type **kubectl get secret common.secret -o yaml** to retrieve the encoded password.
- Type **echo '<encoded-password>' | base64 --decode** to see the encoded password in clear text.

NOTE: If you are currently running an existing version of the proNX Optical Director software, close all browser windows to that proNX Optical Director before you upgrade to the new version.

Use this procedure on the control machine to upgrade the proNX Optical Director software to release 18.4 or higher on all the servers in the cluster. When you upgrade the installation, proNX Optical Director settings and runtime information are preserved. You do not require Internet access to use this procedure.

NOTE: To make the examples in this procedure generic to all releases, the example output does not show release numbers in the filenames.

1. Untar and uncompress the downloaded archive, for example:

```
# cd <install-dir>
# ls
pod-<new-version>.tgz

# tar -xzvf pod-<new-version>.tgz

...

# ls
pod-installer-<new-version>      pod-<new-version>.tgz
```

2. Copy the inventory file from the existing installation to the new installation.

```
# cp <install-dir>/pod-installer-<existing-version>/inventory
<install-dir>/pod-installer-<new-version>/inventory
```

3. If your existing installation was installed with a TLS certificate, then you can skip this step.

   Otherwise, install a valid and trusted TLS certificate and associated private key for the cluster in **<install_dir>/pod-installer-<new-version>/certs**. This TLS certificate will be used by the client browser to authenticate the proNX Optical Director server when you later access the proNX Optical Director UI.

This example shows a certificate and associated private key called pod-cluster.crt and pod-cluster.key respectively.

```
# ls <install_dir>/pod-installer-<new-version>/certs
pod-cluster.crt  pod-cluster.key
```

NOTE: You must have exactly one certificate in this directory. If you have no certificate or more than one certificate, the installation will fail.

If you do not have a TLS certificate, you can create one for testing purposes, as follows:

```
# cd <install_dir>/pod-installer-<new-version>/certs
# openssl req -newkey rsa:2048 -nodes -keyout pod-cluster.key -subj
"/C=CA/ST=ON/O=Juniper/CN=pod-cluster.mydomain" -x509 -days 365 -out
pod-cluster.crt
```

This certificate will allow you to install and try out the proNX Optical Director, but because this certificate has not been signed by a trusted Certificate Authority, you will receive security warnings when trying to access the proNX Optical Director UI.

In this example, the certificate is created with a Common Name (CN) of pod-cluster.mydomain. This is the resolvable TLS hostname that you will use to access the proNX Optical Director UI.

NOTE: The certificate must contain a Common Name and not a Subject Alternative Name (SAN) or IP address.

4. Prepare the upgrade. This step backs up current configuration and runtime information, learns about existing cluster settings, deploys new version containers, and copies updated runtime configuration. The proNX Optical Director application remains up in this step.

```
# cd <install_dir>/pod-installer-<new-version>
# ./scripts/install.sh -p
```

You are prompted for the following:

- A username and password to log in to the cluster servers. This user must be root or a user with no-password sudo access.

Here is an example of the script output:

```
# ./scripts/install.sh -p

Logging to <install-dir>/pod-installer-<new-version>/logs/deploy-cluster.log

[INFO] Long running tasks will be annotated with [*]

=INVENTORY=
MASTER: 10.228.4.241
NODES:  10.228.4.241  10.228.4.106 10.228.4.51
```

```
Checking dependencies                                [COMPLETE]
Collecting environment information                   [PENDING]

Please enter the remote username: [deploy] root
Please enter a password for the "root" user:

Collecting environment information                   [COMPLETE]
Validating cluster connectivity                      [COMPLETE]
Logging cluster details                              [COMPLETE]
Staging cluster upgrade                              [PENDING]
Backing up existing applications                     [COMPLETE]
Transferring images to in-cluster registry [*]       [COMPLETE]
Staging cluster upgrade                              [COMPLETE]
```

The installation log file is located at **<install-dir>/pod-installer-<version>/logs/deploy-cluster.log**.

5. Commit the upgrade. This step upgrades the previously prepared cluster by migrating existing application instances to the new version instances. The proNX Optical Director application goes down in this step.

```
# cd <install_dir>/pod-installer-<new-version>
# ./scripts/install.sh -u
```

You are prompted for the following:

- A username and password to log in to the cluster servers. This user must be root or a user with no-password sudo access.

Here is an example of the script output:

```
# ./scripts/install.sh -u

Logging to <install_dir>/pod-installer-<new-version>/logs/deploy-cluster.log

[INFO] Long running tasks will be annotated with [*]

=INVENTORY=
MASTER: 10.228.4.241
NODES:  10.228.4.241  10.228.4.106 10.228.4.51

Commit upgrade option chosen, enter Y to confirm application downtime: y

Checking dependencies                                [COMPLETE]
Collecting environment information                   [PENDING]

Please enter the remote username: [root]
Please enter a password for the "root" user:

Collecting environment information                   [COMPLETE]
Validating cluster connectivity                      [COMPLETE]
Cluster upgrade                                      [PENDING]
Checking TLS certificate and key                     [COMPLETE]


Cluster UI based on certificate will be: https://pod-cluster.mydomain
```

```
NOTE: The cluster UI will NOT work on any other URL!!!

Press enter Y to confirm the cluster URL: y


Installing cluster applications                    [COMPLETE]
Upgrading applications [*]                         [COMPLETE]
Cluster upgrade                                    [COMPLETE]
```

The installation log file is located at **<install-dir>/pod-installer-<version>/logs/deploy-cluster.log**.

6. Verify that the upgrade is successful.

NOTE: Depending on the installation, the kube config file on the control machine might no longer be valid after the upgrade. If the kubectl commands on the control machine return an error, recreate the **~/.kube/config** file from the **/etc/kubernetes/kubectl.kubeconfig** file on the master node.

```
# kubectl get pods --show-labels

NAME                              READY     STATUS    RESTARTS   AGE
LABELS
joc-control-sd34k                 1/1       Running   1          2m
joc-app=joc-control,joc-version=<version>
joc-db-green-2211827101-fn07w     1/1       Running   0          1h
joc-app=joc-db-green,joc-version=<version>,joc=db,pod-template-hash=2211827101
joc-device-1gf6c                  1/1       Running   0          2m
joc-app=joc-device,joc-lang=java,joc-version=<version>
joc-ingress-tcp-hn158             1/1       Running   0          2m
joc-app=joc-ingress-tcp,joc-version=<version>

<trimmed>
```

NOTE: It is normal to see some joc-version values set to the old version because not all pods might be upgraded.

As the proNX Optical Director starts up, you will see the STATUS of some Kubernetes pods cycle between Running and other values. This is normal. A Kubernetes pod is operational when it has a STATUS of Running and a READY state of 1/1 (number of containers in the pod that are ready / total number of containers in the pod). The installation is complete once all the Kubernetes pods become operational.

7. If you are installing or upgrading to proNX Optical Director Release 18.4, create the secondary database indexes. This is required to provide increased redundancy in cases of server failure. If you are installing or upgrading to release 19.1 or higher, you are not

required to perform this step because this step is automatically performed in those releases.

To create the secondary database indexes, log in to the master node and run the following commands:

NOTE: You cannot run these commands from the control machine.

```
# curl -s http://Administrator:<common-secret>@$(kubectl get pods
--no-headers -o jsonpath='{.items[*].status.podIP}' -l
joc-app=joc-db-green):8093/query/service -d 'statement=create index
idx_auth_green on
operation((meta().id),(meta().cas),login,_class,`password`)'

# curl -s http://Administrator:<common-secret>@$(kubectl get pods
--no-headers -o jsonpath='{.items[*].status.podIP}' -l
joc-app=joc-db-red):8093/query/service -d 'statement=create index
idx_auth_red on operation((meta().id),(meta().cas),login,_class,`password`)'

# curl -s http://Administrator:<common-secret>@$(kubectl get pods
--no-headers -o jsonpath='{.items[*].status.podIP}' -l
joc-app=joc-db-blue):8093/query/service -d 'statement=create index
idx_auth_blue on operation((meta().id),(meta().cas),login,_class,`password`)'
```

where **<common-secret>** is the common secret you specified when you installed the proNX Optical Director software version that you just upgraded from. The **<common-secret>** does not change when you upgrade.

You have now completed the proNX Optical Director upgrade. You can connect to the user interface by pointing your browser to https://<cluster-name> where <cluster-name> is the Common Name used when creating the cluster certificate. In the example certificate above, you connect to the UI using https://pod-cluster.mydomain.

NOTE: If you did not remember to close all browser windows to the previous version of the proNX Optical Director before you installed this version of the proNX Optical Director, perform a browser window refresh on each open browser window.

CHAPTER 3

Appendix

Setting Up the Control Machine (Examples)

Setting Up the Control Machine in a CentOS-Based Distribution (Example)

WARNING: **This example procedure is unsupported and is included purely for informational purposes.**

This procedure uses sample commands and shows outputs that are current as of the writing of this procedure. Because these examples include the installation of third-party tools and software, the commands and outputs might change over time without notice. Be sure to consult the latest third-party documentation for the most up-to-date installation instructions and examples.

Prerequisites

- The control machine has access to the Internet.

Use this procedure to set up the control machine using a CentOS-based distribution.

There are a number of ways to set up the control machine. You can use an existing Linux installation or create a new one.

This procedure shows an example of setting up the control machine with a new minimal CentOS Linux installation. With the minimal CentOS Linux installation, you will need to download a few additional packages. There are many other ways to set up the control machine. Your procedure might be different. Because Linux installation can vary from release to release (even for the same Linux distribution), use this procedure as a guide only.

NOTE: This example assumes you are setting up a temporary control machine for the sole purpose of installing the proNX Optical Director software. As a temporary machine, you only need to select the most basic installation options. If you intend to use the control machine for other purposes, then select the options that match the requirements for those other purposes.

1. Download the Minimal ISO from the CentOS download site (https://www.centos.org/download).

2. Create a DVD or USB drive with that ISO image. You will need to use the appropriate disk imaging tool for the operating system on the machine on which you are creating the drive. You cannot simply copy the image onto the DVD or USB drive.

3. Plug the network cable into the target control machine and ensure the machine has access to the Internet.

   The control machine must be able to reach the Internet for this procedure to download third-party software.

4. Boot up the target control machine with the DVD or USB drive. You might need to change the BIOS boot-up sequence to boot from the DVD or USB drive before the hard drive.

5. Follow the setup screens to set up Linux as you normally do:
   - Configure the language support and keyboard settings.
   - Configure the date and time settings.
   - Specify the installation destination (hard drive) where you want to install the OS. Choose the **Automatically configure partitioning** option.
   - Configure the network and host name and enable the Ethernet interface.
6. Configure the root password.

7. Reboot the machine when prompted.

8. After the machine reboots, log in as root.

9. Update the existing packages. CentOS uses Yellowdog Updater Modified (yum) for installing and updating packages. For example:

```
# yum -y update
Loaded plugins: fastestmirror
base                                                                      |
 3.6 kB  00:00:00
extras                                                                    |
 3.4 kB  00:00:00
updates                                                                  |
3.4 kB  00:00:00
(1/4): base/7/x86_64/group_gz                          | 156 kB  00:00:00
(2/4): extras/7/x86_64/primary_db                    | 145 kB  00:00:01
(3/4): updates/7/x86_64/primary_db                 | 4.6 MB  00:00:04
(4/4): base/7/x86_64/primary_db                      | 5.7 MB  00:00:18
Determining fastest mirrors
 * base:
 * extras:
 * updates:

...
```

10. Install the Extra Packages for Enterprise Linux (epel). For example:

```
# yum -y install epel-release
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base:
 * extras:
 * updates:

...
```

11. Install Ansible.

```
# yum -y install ansible
Loaded plugins: fastestmirror
epel/x86_64/metalink                                |  15 kB  00:00:00
epel                                                                | 4.7 kB
00:00:00
epel/x86_64/primary_db
(1/3): epel/x86_64/group_gz                      | 266 kB  00:00:00
(2/3): epel/x86_64/updateinfo                   | 862 kB  00:00:01
(3/3): epel/x86_64/primary_db                  | 6.2 MB  00:00:07
Loading mirror speeds from cached hostfile
 * base:
 * epel:
 * extras:
 * updates:

...
```

12. Install pip (package manager for Python).

```
# yum -y install python-pip
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base:
 * epel:
 * extras:
 * updates:

...
```

13. Install the Python netaddr module.

```
# pip install netaddr
Collecting netaddr
  Downloading netaddr-xxx (1.6MB)
    100% |                                        | 1.6MB 371kB/s
Installing collected packages: netaddr
Successfully installed netaddr-xxx
```

14. Install wget.

```
# yum -y install wget
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base:
 * epel:
 * extras:
 * updates:

...
```

15. Optionally, install the Kubernetes command line tool (kubectl) in the home directory. This allows you to manage the cluster servers from the control machine.

    See https://kubernetes.io/docs/tasks/tools/install-kubectl for instructions on how to install the Kubernetes command line tool.

    For example, this installs kubectl version 1.6.1 using curl:

```
# cd
# curl -LO
https://storage.googleapis.com/kubernetes-release/release/v1.6.1/bin/linux/amd64/kubectl

  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100 67.4M  100 67.4M    0     0  6736k      0  0:00:10  0:00:10 --:--:--
9097k
```

NOTE: The amd64 path is for both 64-bit AMD and Intel architectures.

16. Change kubectl to enable execute permissions.

```
# chmod +x kubectl
```

17. Set up kubectl so that it is in the execution path.

    kubectl is installed in the home directory. The following configures a link to that executable from the **/usr/local/bin** directory.

```
# cd /usr/local/bin
# ln -s ~/kubectl
# which kubectl
/usr/local/bin/kubectl
```

18. Verify that the Bash, Ansible, Python, and kubectl versions meet the requirements specified in "Control Machine Requirements" on page 22.

    For example:

```
# bash --version
GNU bash, version 4.2.46(2)-release (x86_64-redhat-linux-gnu)
Copyright (C) 2011 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

# ansible --version
ansible 2.7.0
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/root/.ansible/plugins/modules',
u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible

  executable location = /usr/bin/ansible
  python version = 2.7.5

...

 ]# kubectl version
Client Version: version.Info{Major:"1", Minor:"6", GitVersion:"v1.6.1", ...
```

You have now set up the control machine. If you have also set up the required operating system on the cluster servers, then you are ready to start proNX Optical Director software installation.

Setting Up the Control Machine in a Debian-Based Distribution (Example)

WARNING: **This example procedure is unsupported and is included purely for informational purposes.**

This procedure uses sample commands and shows outputs that are current as of the writing of this procedure. Because these examples include the installation of third-party tools and software, the commands and outputs might change over time without notice. Be sure to consult the latest third-party documentation for the most up-to-date installation instructions and examples.

Prerequisites

- The control machine has access to the Internet.

Use this procedure to set up the control machine using a Debian-based (Ubuntu) distribution.

There are a number of ways to set up the control machine. You can use an existing Linux installation or create a new one.

This procedure shows an example of setting up the control machine with a new Linux Ubuntu Server installation. There are many other ways to set up the control machine. Your procedure might be different. Because Linux installation can vary from release to release (even for the same Linux distribution), use this procedure as a guide only.

NOTE: **This example assumes you are setting up a temporary control machine for the sole purpose of installing the proNX Optical Director software. As a temporary machine, you only need to select the most basic installation options. If you intend to use the control machine for other purposes, then select the options that match the requirements for those other purposes.**

1. Download the Ubuntu Server ISO from the Ubuntu download site (https://www.ubuntu.com/download). Select the Ubuntu Server option.

2. Create a DVD or USB drive with that ISO image. You will need to use the appropriate disk imaging tool for the operating system on the machine on which you are creating the drive. You cannot simply copy the image onto the DVD or USB drive.

3. Plug the network cable into the target control machine and ensure the machine has access to the Internet.

   The control machine must be able to reach the Internet for this procedure to download third-party software.

4. Boot up the target control machine with the DVD or USB drive. You might need to change the BIOS boot-up sequence to boot from the DVD or USB drive before the hard drive.

   Select the **Install Ubuntu Server** option. This option contains the most stable version of the kernel.

5. Follow the setup screens to set up Linux as you normally do.

   Use the Tab key to navigate between fields. Use the arrow keys to navigate to a selection within a field. Use the Space bar to select or unselect an option. Use the Enter key to activate your selections.

   - Configure the language, location, and keyboard settings.
   - If your machine has more than one network adapter, select the network adapter to use. This should be the adapter with the Ethernet cable plugged in.
   - Specify the host name you want to use.
   - Configure the user name and password.
   - Confirm your time zone. The time zone is automatically set based on your location.
   - If the control machine has existing partitions, choose the option to unmount them and then select the **Guided - use entire disk** partitioning option. Confirm all settings.

     NOTE: This erases all data from the hard drive.

   - Choose the additional system software that you want to install. As a minimum, select **standard system utilities** and **OpenSSH server**.

6. Reboot the machine when prompted.

7. After the machine reboots, log in as the user you created in step 5.

8. Update the existing packages. Debian-based distributions use the Aptitude package manager (apt) for installing and updating packages.

   If current packages are automatically downloaded as part of the OS installation process, then you do not need to run this command. However, it is always good practice to update your packages just in case.

   ```
   # sudo apt-get -y update
   [sudo] password for user:
   Hit:1 http://us.archive.ubuntu.com/ubuntu xenial InRelease
   Hit:2 http://us.archive.ubuntu.com/ubuntu xenial-updates InRelease
   Hit:3 http://us.archive.ubuntu.com/ubuntu xenial-backports InRelease
   Hit:4 http://security.ubuntu.com/ubuntu xenial-security InRelease
   Reading package lists... Done
   ```

9. Install the software-properties-common package. Some releases might already include this package.

```
# sudo apt-get -y  install software-properties-common
Reading package lists... Done
Building dependency tree
Reading state information... Done
software-properties-common is already the newest version (0.96.20.7).
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
```

10. Install Ansible.

```
# sudo apt-add-repository ppa:ansible/ansible
 Ansible is a radically simple IT automation platform that makes your
applications and systems easier to deploy. Avoid writing scripts or custom
 code to deploy and update your applications— automate in a language that
approaches plain English, using SSH, with no agents to install on remote
systems.

http://ansible.com/
 More info: https://launchpad.net/~ansible/+archive/ubuntu/ansible
Press [ENTER] to continue or ctrl-c to cancel adding it

gpg: keyring `/tmp/tmps3f6dq_m/secring.gpg' created
gpg: keyring `/tmp/tmps3f6dq_m/pubring.gpg' created
gpg: requesting key 7BB9C367 from hkp server keyserver.ubuntu.com

...

# sudo apt-get -y update
Hit:1 http://security.ubuntu.com/ubuntu xenial-security InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu xenial InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu xenial-updates InRelease
Get:4 http://ppa.launchpad.net/ansible/ansible/ubuntu xenial InRelease [18.0
 kB]

...

# sudo apt-get -y  install ansible
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libpython-stdlib libpython2.7-minimal libpython2.7-stdlib libyaml-0-2
python python-cffi-backend
  python-crypto python-cryptography python-ecdsa python-enum34
python-httplib2 python-idna
  python-ipaddress python-jinja2 python-markupsafe python-minimal
python-paramiko python-pkg-resources
  python-pyasn1 python-setuptools python-six python-yaml python2.7
python2.7-minimal sshpass

...
```

11. Install pip (package manager for Python).

```
# sudo apt-get -y  install python-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

```
The following additional packages will be installed:
  binutils build-essential cpp cpp-5 dpkg-dev fakeroot g++ g++-5 gcc gcc-5
 libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan2 libatomic1
libc-dev-bin libc6-dev libcc1-0
  libcilkrts5 libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl
 libgcc-5-dev libgomp1 libisl15

....
```

12. Install the Python netaddr module.

```
# pip install netaddr
Collecting netaddr
  Downloading netaddr-xxx (1.6MB)
    100% |
     | 1.6MB 217kB/s
Installing collected packages: netaddr
Successfully installed netaddr
```

13. Optionally, install the Kubernetes command line tool (kubectl) in the home directory. This allows you to manage the cluster servers from the control machine.

    See https://kubernetes.io/docs/tasks/tools/install-kubectl for instructions on how to install the Kubernetes command line tool.

    For example, this installs kubectl version 1.6.1 using curl:

```
# cd
# curl -LO
https://storage.googleapis.com/kubernetes-release/release/v1.6.1/bin/linux/amd64/kubectl

  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100 67.4M  100 67.4M    0     0  6736k      0  0:00:10  0:00:10 --:--:--
9097k
```

NOTE: The amd64 path is for both 64-bit AMD and Intel architectures.

14. Change kubectl to enable execute permissions.

```
# chmod +x kubectl
```

15. Set up kubectl so that it is in the execution path.

    kubectl is installed in the home directory. The following configures a link to that executable from the **/usr/local/bin** directory.

```
# cd /usr/local/bin
# sudo ln -s ~/kubectl
# which kubectl
/usr/local/bin/kubectl
```

16. Verify that the Bash, Ansible, Python, and kubectl versions meet the requirements specified in "Control Machine Requirements" on page 22.

    For example:

```
# bash --version
GNU bash, version 4.3.48(1)-release (x86_64-pc-linux-gnu)
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

# ansible --version
ansible 2.7.0
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/root/.ansible/plugins/modules',
u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible

  executable location = /usr/bin/ansible
  python version = 2.7.5

...

 ]# kubectl version
Client Version: version.Info{Major:"1", Minor:"6", GitVersion:"v1.6.1", ...
```

You have now set up the control machine. If you have also set up the required operating system on the cluster servers, then you are ready to start proNX Optical Director software installation.

Setting Up the Control Machine in Mac OS X (Example)

WARNING: **This example procedure is unsupported and is included purely for informational purposes.**

This procedure uses sample commands and shows outputs that are current as of the writing of this procedure. Because these examples include the installation of third-party tools and software, the commands and outputs might change over time without notice. Be sure to consult the latest third-party documentation for the most up-to-date installation instructions and examples.

Prerequisites

- The control machine has access to the Internet.

Use this procedure to set up the control machine in Mac OS X.

1. Launch a Terminal window.

2. Install the Homebrew package manager if your machine does not already have it installed.

   To check if Homebrew is already installed:

   ```
   $ brew doctor
   ```

   If the output of this command indicates that Homebrew is ready to brew, proceed to step 3.

   Otherwise, follow the instructions at https://brew.sh to install Homebrew.

   For example:

   ```
   $ /usr/bin/ruby -e "$(curl -fsSL
   https://raw.githubusercontent.com/Homebrew/install/master/install)"

   ==> This script will install:
   /usr/local/bin/brew
   /usr/local/share/doc/homebrew
   /usr/local/share/man/man1/brew.1
   /usr/local/share/zsh/site-functions/_brew
   /usr/local/etc/bash_completion.d/brew
   /usr/local/Homebrew

   ...

   $ brew doctor
   Your system is ready to brew.
   ```

   NOTE: In some versions of Mac OS X, you might need to change the ownership of the /usr/local directory for Homebrew to install successfully (for example, sudo chown *user*:admin /usr/local and sudo chown -R *user*:admin /usr/local), where *user* is your username.

   Homebrew is installed in **/usr/local**.

3. Install Ansible. For example:

   ```
   $ brew install ansible
   ==> Installing dependencies for ansible: readline, sqlite, gdbm, openssl,
   python, libyaml, openssl@1.1
   ==> Installing ansible dependency: readline
   ==> Downloading
   https://homebrew.bintray.com/bottles/readline-7.0.3_1.sierra.bottl
   ######################################################################
   100.0%
   ==> Pouring readline-7.0.3_1.sierra.bottle.tar.gz
   ```

```
==> Caveats

…
```

4. Install the Python netaddr module. For example:

```
$ pip2 install netaddr
Collecting netaddr
  Downloading netaddr-xxx (1.6MB)
    100% |                                                                | 1.6MB
634kB/s
Installing collected packages: netaddr
Successfully installed netaddr-xxx
```

NOTE: If the pip2 command is not found, use the pip install netaddr command.

5. Install sshpass. The sshpass utility is used for non-interactive SSH login, which takes place during proNX Optical Director installation.

   a. Remove the existing **sshpass.rb** file if it exists. For example:

```
$ rm
/usr/local/homebrew/Library/Taps/homebrew/homebrew-core/Formula/sshpass.rb
```

   b. Create a new **sshpass.rb** file. This command exits by placing you in an edit session of the new **sshpass.rb** file.

```
$ brew create
https://sourceforge.net/projects/sshpass/files/sshpass/1.06/sshpass-1.06.tar.gz
 --force

==> Downloading
https://sourceforge.net/projects/sshpass/files/sshpass/1.06/sshpass-1.06.tar.gz
==> Downloading from
https://sourceforge.net/projects/sshpass/files/sshpass/1.06/sshpass-1.06.tar.gz/download
######################################################################
 100.0%
Please `brew audit --new-formula sshpass` before submitting, thanks.
Editing
/usr/local/Homebrew/Library/Taps/homebrew/homebrew-core/Formula/sshpass.rb
Warning: Using vim because no editor was set in the environment.
This may change in the future, so we recommend setting EDITOR,
or HOMEBREW_EDITOR to your preferred text editor.

# Documentation: https://docs.brew.sh/Formula-Cookbook.html
#
http://www.rubydoc.info/github/Homebrew/brew/master/Formula
# PLEASE REMOVE ALL GENERATED COMMENTS BEFORE SUBMITTING YOUR PULL
REQUEST!

class Sshpass < Formula
  desc ""
```

```
  homepage ""
  url
"https://sourceforge.net/projects/sshpass/files/sshpass/1.06/sshpass-1.06.tar.gz"

  sha256
"c6324fcee608b99a58f9870157dfa754837f8c48be3df0f5e2f3accf145dee60"

  # depends_on "cmake" => :build

  def install
   # ENV.deparallelize  # if your formula fails when building in parallel


    # Remove unrecognized options if warned by configure
    system "./configure", "--disable-debug",
                          "--disable-dependency-tracking",
                          "--disable-silent-rules",
                          "--prefix=#{prefix}"
    # system "cmake", ".", *std_cmake_args
    system "make", "install" # if this fails, try separate make/make
install steps
  end

  test do
    # `test do` will create, run in and delete a temporary directory.
    #
    # This test will fail and we won't accept that! For
Homebrew/homebrew-core
    # this will need to be a test that verifies the functionality of the

    # software. Run the test with `brew test sshpass`. Options passed
    # to `brew install` such as `--HEAD` also need to be provided to
`brew test`.
    #
    # The installed folder is not in the path, so use the entire path
to any
    # executables being tested: `system "#{bin}/program", "do",
"something"`.
    system "false"
  end
end
~

~

~
```

c. Simply exit the edit session. For vim, type **:q**.

d. Install sshpass.

```
$ brew install sshpass
Updating Homebrew...
==> Auto-updated Homebrew!
Updated 1 tap (caskroom/cask).
No changes to formulae.

==> Downloading
https://sourceforge.net/projects/sshpass/files/sshpass/1.06/sshpass-1.06.tar.gz
Already downloaded:
```

```
  /Users/audrey/Library/Caches/Homebrew/sshpass-1.06.tar.gz
  ==> ./configure --disable-silent-rules
  --prefix=/usr/local/Cellar/sshpass/1.06
  ==> make install
    /usr/local/Cellar/sshpass/1.06: 9 files, 42.7KB, built in 13 seconds
```

6. Optionally, install the Kubernetes command line tool (kubectl) in the home directory. This allows you to manage the cluster servers from the control machine.

   See https://kubernetes.io/docs/tasks/tools/install-kubectl for instructions on how to install the Kubernetes command line tool.

   For example, this installs kubectl version 1.6.1 using curl:

```
# cd
# sudo curl -LO
https://storage.googleapis.com/kubernetes-release/release/v1.6.1/bin/darwin/amd64/kubectl
Password:
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100 66.9M  100 66.9M    0     0   398k      0  0:02:52  0:02:52 --:--:--
387k
```

   NOTE: The amd64 path is for both 64-bit AMD and Intel architectures.

7. Verify that the Bash, Ansible, Python, and kubectl versions meet the requirements specified in “Control Machine Requirements” on page 22.

   For example:

```
# bash --version
GNU bash, version 3.2.57(1)-release (x86_64-apple-darwin16)
Copyright (C) 2007 Free Software Foundation, Inc.

# ansible --version
ansible 2.7.1
  config file = None
  configured module search path = [u'/Users/audrey/.ansible/plugins/modules',
 u'/usr/share/ansible/plugins/modules']
  ansible python module location = /Library/Python/2.7/site-packages/ansible

  executable location = /usr/local/bin/ansible
  python version = 2.7.10

...

# kubectl version
Client Version: version.Info{Major:"1", Minor:"6", GitVersion:"v1.6.1", ...
```

You have now set up the control machine. If you have also set up the required operating system on the cluster servers, then you are ready to start proNX Optical Director software installation.

Prometheus

Overview

The proNX Optical Director supports the use of Prometheus to monitor the Kubernetes pods used in the proNX Optical Director application. Prometheus is an open source systems monitoring toolkit that collects and displays metrics from monitored targets. Advanced users can use Prometheus to check on the health of the proNX Optical Director installation and look at various indicators to confirm that the proNX Optical Director is running normally.

The proNX Optical Director supports two types of metrics:

- Gauges - A gauge is a metric that represents a measured value that can go up or down. An example of a gauge is the number of HTTP sessions that are active in the RESTCONF layer described in "Northbound Interface" on page 17.
- Counters - A counter is a cumulative metric that is monotonically increasing. An example of a counter is the number of successful authentication requests.

The proNX Optical Director software package includes a Prometheus configuration file that is preconfigured to monitor the Device, Network, and Operations pods used in the proNX Optical Director. The Prometheus configuration file is located at **/var/data/conf/prometheus-conf.yml** on the master node. You do not need to make any changes to the configuration file, but you are required to download the Prometheus docker image because the image is not included in the proNX Optical Director software package.

You can use the procedures in the following sections to get started with Prometheus. However, a detailed explanation of Prometheus and the supported metrics is beyond the scope of this document. For information on Prometheus, see https://prometheus.io/docs/.

Downloading and Installing Prometheus

Prerequisites

- The master node where you are installing Prometheus has access to the Internet.

Use this procedure to download and install Prometheus on the master node in your proNX Optical Director server cluster.

1. Log in to the Linux shell on the master node using SSH.

   If you are on a Linux or MAC OS X machine, launch a terminal window and start an SSH session. If you are on a Windows machine, install and launch a terminal application (for example, PuTTY).

   For example:

   ```
   ssh -p22 user@<master-node-ip>
   ```

2. Download the docker image.

   For example:

   ```
   # docker pull prom/prometheus
   Using default tag: latest
   Trying to pull repository docker.io/prom/prometheus ...
   latest: Pulling from docker.io/prom/prometheus
   aab39f0bc16d: Pull complete
   a3ed95caeb02: Pull complete
   2cd9e239cea6: Pull complete
   0266ca3d0dd9: Pull complete
   341681dba10c: Pull complete
   8f6074d68b9e: Pull complete
   2fa612efb95d: Pull complete
   151829c004a9: Pull complete
   75e765061965: Pull complete
   b5a15632e9ab: Pull complete
   Digest: sha256:14cafab6a73222c859cf46c16bda6b94839d31b7ab25e70ac8aa19809c6055a8
   ```

3. Verify that Prometheus has been downloaded.

   For example:

   ```
   # docker images | grep prometheus
   docker.io/prom/prometheus    latest   cc866859f8df  9 weeks ago 113.3 MB
   ```

4. Start the Prometheus container.

   For example:

   ```
   # kubectl create -f /etc/kubernetes/apps/joc/debug/joc-prometheus.yml
   ```

5. Verify that the Prometheus container is up and running.

   For example:

   ```
   # kubectl get pods | grep prometheus

   joc-prometheus                       1/1       Running   0          2m
   ```

Prometheus is up and running when the joc-prometheus container has a READY value of 1/1 and a STATUS of Running.

Using Prometheus

Use this procedure to monitor the proNX Optical Director using Prometheus.

NOTE: This procedure provides a cursory look at Prometheus. For more detailed information on Prometheus, see the official documentation at https://prometheus.io/docs/.

1. Access the Prometheus user interface. Prometheus can be reached from your browser on port 9090.

   For example: http://<server-ip>:9090

2. Select the **Graph** tab.

3. Select the time range and end time.

4. Use the drop-down list to select the metric to display and click **Execute**.

   NOTE: If you get the message “No datapoints found”, expand the time range until the message disappears.

   For example, the following shows a graph of the number of threads that the Device, Network, and Operations pods are using in the selected timeframe.

Uninstalling Prometheus

Use this procedure to uninstall Prometheus on the master node in your proNX Optical Director server cluster.

1. Log in to the Linux shell on the master node using SSH.

   If you are on a Linux or MAC OS X machine, launch a terminal window and start an SSH session. If you are on a Windows machine, install and launch a terminal application (for example, PuTTY).

   For example:

   ```
   ssh -p22 user@<master-node-ip>
   ```

2. Delete the Prometheus container.

   For example:

   ```
   # kubectl delete -f /etc/kubernetes/apps/joc/debug/joc-prometheus.yml
   pod "joc-prometheus" deleted
   ```

3. Verify that the Prometheus container has indeed been deleted.

   For example:

   ```
   # kubectl get pods

   NAME                                READY     STATUS    RESTARTS   AGE
   joc-control-338968877-rvdl3         1/1       Running   3          3d
   joc-db-master-1914135614-zh3pg      1/1       Running   0          3d
   joc-device-mhxzg                    1/1       Running   2          3d
   joc-ingress-tcp-7ws22               1/1       Running   0          3d
   joc-ingress-udp-g82kv               1/1       Running   0          3d
   joc-messaging-3471602017-776tp      1/1       Running   0          3d
   joc-network-2733954367-cffq6        1/1       Running   2          3d
   joc-operation-1732558567-p80j5      1/1       Running   3          3d
   joc-registry-275104000-zjlcd        1/1       Running   0          3d
   joc-snmp-vip-zt5kk                  1/1       Running   1          3d
   joc-userinterface-1159436082-t3scw  1/1       Running   3          3d
   ```

   The joc-prometheus container should no longer be in the list of pods.