JUNIPER NETWORKS | Engineering Simplicity

Paragon Automation Installation Guide

Published
2021-12-16

RELEASE
21.2

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

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Paragon Automation Installation Guide
21.2
Copyright © 2021 Juniper Networks, Inc. All rights reserved.

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

About This Guide

Use this guide to install Paragon Automation on a Linux server.

RELATED DOCUMENTATION

- Paragon Automation User Guide
- Paragon Automation Release Notes, Release 21.2

CHAPTER 1

Introduction

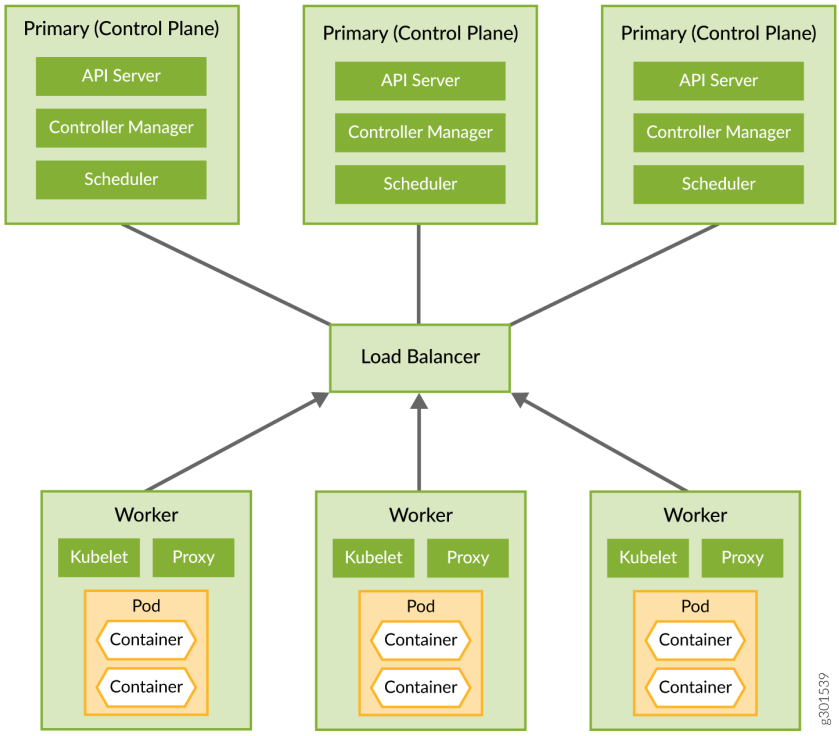
Paragon Automation Portfolio Installation Overview

Juniper® Paragon Automation Portfolio is a cloud-ready solution for network planning, configuration, provisioning, traffic engineering, monitoring, and lifecycle management. This solution brings advanced visualization capabilities and analytics to network management and monitoring. Paragon Automation is deployed as an on-premises (customer managed) application. Paragon Automation Portfolio offers users a suite of microservices-based applications including Paragon Insights (previously known as HealthBot), Paragon Planner (previously known as NorthStar Planner), and Paragon Pathfinder (previously known as NorthStar Controller).

Paragon Automation is built on a Kubernetes cluster, and is a collection of microservices that interact with one other through APIs. A Kubernetes cluster contains one or more primary nodes (control plane nodes) and one or more worker nodes (compute nodes) as illustrated in Figure 1 on page 3. These nodes can be virtual machines (VMs) or bare metal servers, but must be running the same OS (Ubuntu, CentOS, or RHEL) version of Linux. The control plane manages the cluster, and the compute nodes run the application workloads. The applications run in containers. Containers are not tied to individual nodes, but are abstracted across the cluster. A Paragon Automation Kubernetes cluster consists of

primary and worker nodes. The total number of cluster nodes depends on the intended capacity of the system.

Figure 1: Kubernetes Cluster



Installation is automated using Ansible playbooks. The Ansible playbooks are packaged in a Docker image, and can be executed on a separate dedicated host (control host), which has Docker installed, and can mount local directories into a Docker container. You must have a dedicated machine functioning as the control host which will install the required software on all the cluster nodes

You can install Paragon Automation by downloading an installation bundle and running the installer on the control host. You must download the installation bundle on the control host, and then create, and configure the installation files required for installation. Installation is controlled through several variables that are defined in configuration files that are created and updated during the installation process. Based on the configuration files, the Ansible playbook deploys the Kubernetes cluster. You must have internet access to download the packages on the control host. You must also have internet access on the cluster nodes to download any additional software such as Docker and OS patches.

This guide describes how to install Paragon Automation and is intended for network operators and administrators who install, configure, and manage the network infrastructure. This guide explains how to:

- Install and update Paragon Automation
- Uninstall Paragon Automation
- Add and remove nodes
- Back up and restore a configuration
- Migrate data from your existing setup to Paragon Automation
- Perform common installation troubleshooting tasks

NOTE: You must perform a fresh installation of Paragon Automation Release 21.2. You **cannot** upgrade from Release 21.1 to Release 21.2.

RELATED DOCUMENTATION

Paragon Automation Overview

Installation Prerequisites | **11**

Install Paragon Automation | **20**

2

CHAPTER

System Requirements

Paragon Automation System Requirements

IN THIS SECTION

Before you install the Paragon Automation software, ensure that your system meets the requirements described in these sections.

Minimum Hardware Requirements

Dimensioning of your Paragon Automation production deployment is based on the network scale and features used. Table 1 on page 6 lists the minimum hardware requirements.

Table 1: Minimum Hardware Requirements

Node	Minimum Number of Nodes Required	Hardware Requirement	**Storage Requirement**	Role
Control Host	1	2–4-core CPU, 12-GB RAM, 100-GB HDD	NA	Carry out Ansible operations to install the cluster.
Primary	1	8-core CPU, 32-GB RAM, 250-GB SSD	NA	Kubernetes primary node

Table 1: Minimum Hardware Requirements *(Continued)*

Node	Minimum Number of Nodes Required	Hardware Requirement	**Storage Requirement**	Role
Worker	3	8-core CPU, 32-GB RAM, 250-GB SSD	Minimum three nodes with unpartitioned disks or unformatted disk partitions attached for system storage requirements. See "Disk Partition Requirements" on page 8.	Kubernetes worker node

The numbers listed here are minimum requirements. The actual system resources is determined by the intended capacity of the system, including number of devices to be monitored, type of sensors, frequency of telemetry messages, and number of playbooks and rules.

Proof-of-concept (POC) systems typically support up to two device groups with two devices and two to three playbooks per device group across all Paragon Automation components, which is supported with the minimum resources specified in Table 1 on page 6. Increase in the number of device groups, devices, or playbooks, will need higher CPU and memory capacities.

Similarly, the total number of cluster nodes depends on the intended capacity of the system, you need at least one primary and one worker. While it is possible to have a single node acting as both primary and worker it is not recommended.

For a fully redundant setup, you must have a minimum of three primary nodes and one worker node. In this case, master scheduling must be enabled when the installation configuration file is created.

In release 21.2, Paragon Automation uses Ceph to provide system storage. For successful installation, you need three or more worker nodes with unpartitioned disks or unformatted disk partitions available. If master scheduling is configured, then primary nodes can also provide storage.

NOTE: To get a production deployment scale and size estimate and to discuss detailed dimensioning requirements, contact your Juniper Partner or Juniper Sales Representative.

Software Requirements

- Ubuntu version 18.04.4 or later, or RHEL version 8.3, or CentOS versions later than version 7.1 must be installed on each node.
- Docker must be installed on the control host. If you are using Docker CE, the recommended version is 18.09 and later.

  If you are using Docker EE, the recommended version is 18.03.1-ee-1 and later. Also, to use Docker EE, you must install Docker EE on the cluster nodes.

  Docker allows you to run the Paragon Automation installer file, which is packaged with Ansible (version 2.9.5) and the roles and playbooks that are required to install the cluster.

> **NOTE**: Installation will fail if you do not have the correct versions. Commands to verify these versions are described in this guide in subsequent sections.

Disk Partition Requirements

The following disk partitions are recommended for the cluster nodes.

- Root partition mounted at / with minimum 20-GB space.
- Unformatted partition or disk for Ceph storage with minimum 20-GB space.

The following optional disk partitions are recommended.

- Docker partition mounted at **/var/lib/docker** with minimum 50-GB space.
- Data partition mounted at **/export** with minimum 50-GB space. The data partition is used for Postgres, Zookeeper, Kafka, and Elasticsearch.
- Data partition mounted at **/var/local** with minimum 50-GB space, The data partition is used for Paragon Insights influxdb.

  You can mount **/export** and **/var/local** on the same partition, but you must mount **/var/lib/docker** on a separate file system to increase the stability of the cluster. If one of the databases fills up its available disk space, Kubernetes can respond by shutting down applications and reschedule them on other nodes.

If you do not create the optional partitions, you must add their minimum size requirements to the root partition.

Web Browser Requirements

Table 2 on page 9 lists the 64-bit Web browsers that support Paragon Automation.

Table 2: Supported Web Browsers

Browser	Supported Versions	Supported OS Versions
Chrome	85 and later	Windows 10
Firefox	79 and later	Windows 10
Safari	14.0.3	MacOS 10.15 and later

RELATED DOCUMENTATION

Installation Prerequisites | 11

Install Paragon Automation | 20

3

CHAPTER

Install and Update Paragon Automation

Installation Prerequisites

IN THIS SECTION

To successfully install and deploy a Paragon Automation cluster, you must have a dedicated machine that functions as the control host and installs the distribution software on a number of cluster nodes. You can download the distribution software on the control host, and then create and configure the installation files to run the installation from the control host. You must have internet access to download the packages on the control host. You must also have internet access on the cluster nodes to download any additional software such as Docker, and OS patches.

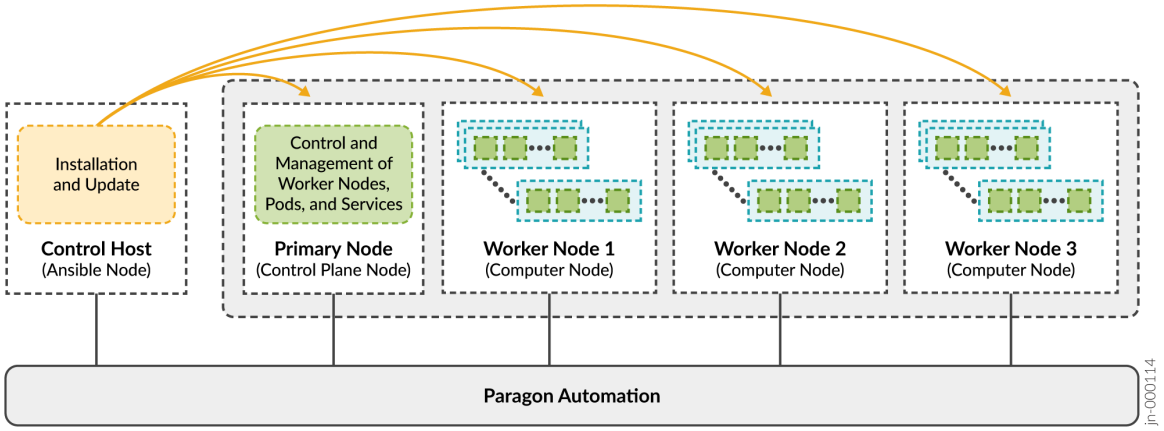
Before you download and install the distribution software, you must preconfigure the control host and the cluster nodes as described in this topic.

Prepare the Control Host

The control host is a dedicated machine that is used to orchestrate the installation of a Paragon Automation cluster. You must download the installer packages on the control host. The control host carries out the Ansible operations that runs the software installer and installs the software on the cluster nodes as illustrated in Figure 2 on page 12. The control host also installs any additional packages such as optional OS packages, Docker, and Elasticsearch on the cluster nodes. The control node requires internet access to download software. All microservices, including third-party microservices, are downloaded onto the control host, and do not access any public registries during installation. The

control host can be on a different broadcast domain from the cluster nodes, but needs SSH access to the nodes.

Figure 2: Control Host



Once installation is complete, the control host plays no role in the functioning of the cluster. However, you will need the control host to update the software or any component, make changes to the cluster, or re-install it if a node fails. You can also use the control host to archive configuration files. We recommend that you keep the control host available, and not use it for something else, after installation.

Ensure that the control host meets the following prerequisites:

- A base OS of any Linux distribution that allows installation of Docker CE or Docker EE must be installed.
- Docker must be installed and configured on the control host to implement the Linux container environment. Paragon Automation Release 21.2 supports Docker EE in addition to Docker CE. The Docker version you choose to install in the control host is independent of the Docker version you plan to use in the cluster nodes.

  If you want to install Docker EE, ensure that you have a trial or subscription before installation. For more information on Docker EE, supported systems, and installation instructions, see https://www.docker.com/blog/docker-enterprise-edition/.

  To download and install Docker CE, perform the following steps:

  - **On RHEL:**

The following commands will install the latest stable version on x86 machines.

```
$ sudo dnf config-manager --add-repo=https://download.docker.com/linux/centos/docker-ce.repo
$ sudo dnf install docker-ce --nobest -y
$ sudo systemctl disable firewalld
$ sudo systemctl start docker
$ sudo systemctl enable docker
```

To verify that Docker is installed and running, use the `# docker run hello-world` command.

To verify the Docker version installed, use the `# docker version` or `# docker --version` commands.

- **On Ubuntu OS**:

  The following commands install the latest stable version on x86 machines.

```
# sudo apt-get install -y apt-transport-https ca-certificates curl gnupg-agent software-properties-common
# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
# sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
# sudo apt-get update
# sudo apt-get install -y docker-ce docker-ce-cli containerd.io
```

  To verify that Docker is installed and running, use the `# docker run hello-world` command.

  To verify the Docker version installed, use the `# docker version` or `# docker --version` commands.

  For full instructions and more information, see https://docs.docker.com/engine/install/ubuntu/.

- **On CentOS**:

  The following commands will install the latest stable version on x86 machines.

```
$ sudo yum install -y yum-utils \
   device-mapper-persistent-data \
   lvm2
$ sudo yum-config-manager \
     --add-repo \
     https://download.docker.com/linux/centos/docker-ce.repo
```

```
$ sudo yum install docker-ce
$ sudo systemctl start docker
```

To verify that Docker is installed and running, use the `$ docker run hello-world` command.

To verify the Docker version installed, use the `$ docker version` or `$ docker --version` commands.

For full instructions and more information, see https://docs.docker.com/engine/install/centos/.

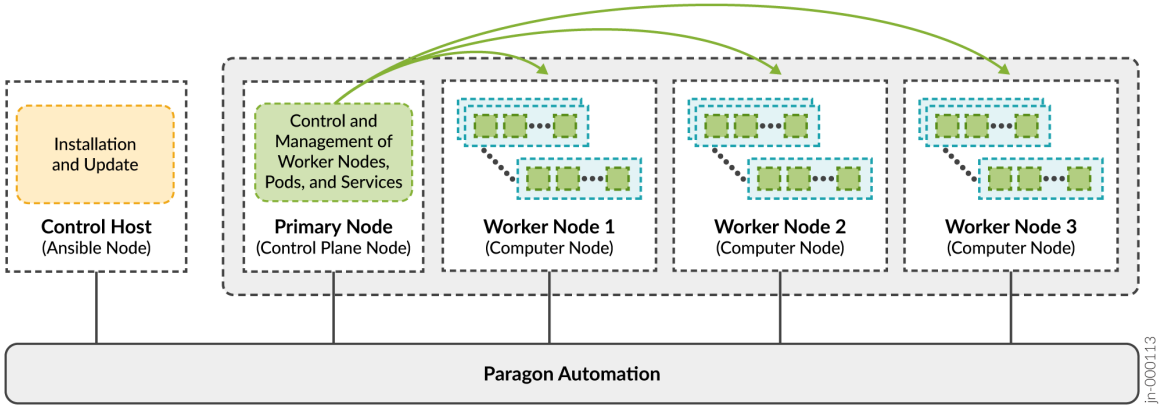
- The installer running on the control host must be connected to the cluster nodes through SSH using the install user account.
- The `wget` package must be installed. Use the `wget` tool to download the Paragon Automation distribution software.
  - On RHEL, use the `$ dnf install wget` command.
  - On Ubuntu OS, use the `# apt install wget` command.
  - On CentOS, use the `$ yum install wget` command.

Prepare Cluster Nodes

Paragon Automation is installed on a Kubernetes cluster of one or more primary nodes (control plane nodes) and one or more worker nodes (compute nodes) as illustrated in Figure 3 on page 15. The

control plane manages the cluster, and the compute nodes run the application workloads. The primary and worker nodes are collectively called the cluster nodes.

Figure 3: Cluster Nodes



Ensure that the cluster nodes meet the following prerequisites:

- Each cluster node must have a static, unique IP address. We recommend that all the nodes be in the same broadcast domain. The hostnames of the node can have only lower case alphabets and can have no special characters other than “-” and “.” .

  For cluster nodes in different broadcast domains, see "Load Balancing Configuration" on page 18 for additonal load balancing configuration.

  The cluster nodes need not be accessible from outside the cluster. Access to the Kubernetes cluster is managed by separate virtual IP addresses. For more information, see "Virtual IP Address Considerations" on page 18.

- A base OS of Ubuntu version 18.04.4 or later, or RHEL version 8.3, or CentOS version later than 7.1, must be installed on each node. To verify the installed OS version, use the `lsb_release -a` command.

- The cluster nodes must have raw storage block devices with unpartitioned disks or unformatted disk partitions attached. The nodes can also be partitioned such that a portion of the disk space available is used for the root partition and other files systems. The remaining space must be unpartitioned with no file systems and reserved for Ceph to use. For redundancy, you must have a minimum of three cluster nodes with storage space attached. Installation *will* fail if disks are unavailable. For more information, see "Disk Partition Requirements" on page 8.

  Ceph requires relatively newer Kernel versions. If your Linux kernel is very old, consider upgrading or reinstalling a new one. For a list of minimum Linux kernel versions supported by Ceph for your OS, see https://docs.ceph.com/en/latest/start/os-recommendations.

- All nodes must run NTP or other time-synchronization at all times.
- The install user must be a root user or have superuser (sudo) privileges.
- An SSH server must be running on all nodes. The installer running on the control host connects to the cluster nodes through SSH using the install user account. You might need to edit the **sshd_config** file to allow root login, depending on the authentication method selected. See "2.d" on page 23 of the installation process.
- Select one of the following Docker versions to install.
  - Docker CE—If you want to use Docker CE, you need not explicitly install it on the cluster nodes. The `deploy` script installs Docker CE on the nodes during installation of Paragon Automation.
  - Docker EE—If you want to use Docker EE, you *must* install Docker EE on *all* the cluster nodes. If you install Docker EE on the nodes, the `deploy` script uses the installed version and does not attempt to install Docker CE in its place. For more information on Docker EE, supported systems, download, and installation instructions, see https://www.docker.com/blog/docker-enterprise-edition/.

  The Docker version you choose to install in the cluster nodes is independent of the Docker version installed in the control host.
- Kubernetes requires iptables rules to accept forwarding traffic. Installing Docker might create a firewall that prevents forwarding traffic. Configure the iptables firewall settings to accept all packets by default using the `iptables -P FORWARD ACCEPT` command.

  Inter-cluster communication between the nodes must be allowed. In particular, the ports listed in *Table 1* must be kept open for communication.

Table 3: Ports That Must Be Allowed by External Firewalls

Port Numbers	Purpose
6443, 2379-2380, 10250, 10252, 10255	TCP
30000-32767	Kubernetes port assignment range
UI access	
22	SSH daemon
80	HTTP

Table 3: Ports That Must Be Allowed by External Firewalls *(Continued)*

Port Numbers	Purpose
443	HTTPS
7000	Paragon Planner communications
Communication between network elements	
7804	NETCONF callback
161	SNMP (UDP)

- Python must be installed on the cluster nodes. If not pre-installed on your OS, install Python 3 on the cluster nodes:
  - On RHEL:

    To install Python 3, use the `# yum install python3` command.

    To verify the Python version installed, use the `$ python3 --version` command.
  - On Ubuntu OS:

    To install Python 3.8, use the `# apt install python3.8` command.

    To verify the Python version installed, use `# python -V` or `# python --version` commands.
  - On CentOS:

    To install Python 3, use the `$ yum install -y python3` command.

    To verify the Python version installed, use `$ python -V` or `$ python --version` commands.

  Python 2.7 is also supported.

Virtual IP Address Considerations

IN THIS SECTION

Access to the Paragon Automation cluster from outside the cluster is through virtual IP addresses (VIPs) that are managed by a load balancer. You require up to five VIPs for a cluster. The VIPs can be within the same broadcast domain as the cluster nodes or in a different broadcast domain.

You must identify the following VIPs before you install Paragon Automation.

- In case of a multi-primary node setup, you need one VIP in the same broadcast domain as the cluster nodes. This IP address is used for communication between the primary and worker nodes. This IP address is referred to as the Kubernetes Master Virtual IP address in the installation wizard.
- You also need a VIP for each of the following load-balanced services:
  - Ingress controller—Paragon Automation provides a common Web server that provides access for installing applications. Access to the server is managed through the Kubernetes Ingress Controller. This VIP is used for Web access of the Paragon Automation GUI.
  - Paragon Insights services—This VIP is used for DHCP services such as SNMP, syslog, and DHCP relay.
  - Paragon Pathfinder PCE server—Used to establish PCEP sessions with devices in the network.
- SNMP trap receiver proxy (Optional)—You should configure a VIP for the SNMP trap receiver proxy only if this functionality is required.

Load Balancing Configuration

VIPs are managed in Layer 2 by default. When all cluster nodes are in the same broadcast domain, each VIP is assigned to one cluster node at a time. If the cluster nodes are in different broadcast domains, you must configure a load balancer in Layer 3 to load balance between the nodes.

You must configure a BGP router to advertise the VIP to the network. The BGP router should be configured to use ECMP to balance TCP/IP sessions between different hosts. Connect the BGP router directly to the cluster nodes.

To configure load balancing on the cluster nodes, edit the **config.yml** file. For example:

```
metallb_config:
  peers:
    - peer-address: 192.x.x.1 ## address of BGP router
      peer-asn: 64501 ## autonomous system number of BGP router
      my-asn: 64500 ## ASN of cluster
  address-pools:
    - name: default
      protocol: bgp
      addresses:
        - 10.x.x.0/24
```

In this example, The BGP router at 192.x.x.1 is responsible to advertise reachability for the VIPs with the 10.x.x.0/24 prefix to the rest of the network. The cluster allocates the VIP of this range and advertises the address for the cluster nodes that can handle the address.

DNS Server Configuration (Optional)

You can access the main Web gateway either through the ingress controller VIP or through a hostname that is configured in the DNS that resolves to the ingress controller VIP. You need to configure DNS only if you want to use a hostname to access the Web gateway.

Add the hostname to DNS as A, AAAA, or CNAME record. For lab and POC setups, you can add the hostname to the **/etc/hosts** file on the cluster nodes.

RELATED DOCUMENTATION

Paragon Automation System Requirements | 6

Install Paragon Automation | 20

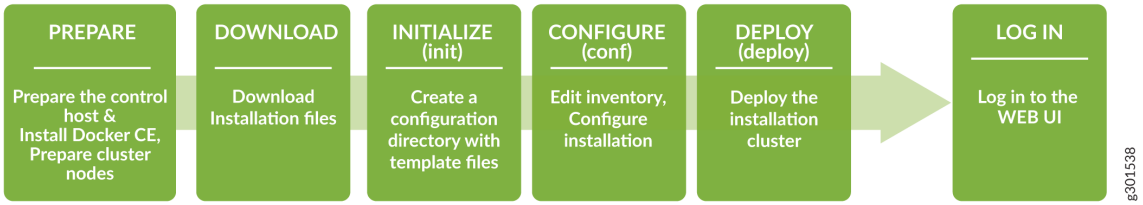
Uninstall Paragon Automation | 37

Install Paragon Automation

IN THIS SECTION

This topic describes the installation of the Paragon Automation cluster. The order of installation tasks is shown at a high level in Figure 4 on page 20. Ensure that you have completed all the preconfiguration and preparation steps described in "Installation Prerequisites" on page 11 before you begin installation.

Figure 4: High-Level Process Flow for Installing Paragon Automation



Download the Software

Prerequisite

- You need a Juniper account to download the Paragon Automation software.

1. Log in to the control host.
2. Create a directory in which you download the software.

   This directory is referred to as ***pa-download-dir*** in this guide.
3. From the Version drop-down list on the Paragon Automation software download page at https://support.juniper.net/support/downloads/?p=pa, select the version number.

4. Download the **Paragon Automation Setup** installation files to the download folder using the `wget "http://cdn.juniper.net/software/file-download-url"` command.

   The Paragon Automation Setup installation bundle consists of the following scripts and tar files to install each of the component modules:

   - `davinci.tar.gz`, which is the primary installer file.
   - `infra.tar`, which installs the Kubernetes infrastructure components including Docker and Helm.
   - `ems.tar`, which installs the base platform component.
   - `northstar.tar`, which installs the Paragon Pathfinder and Paragon Planner components.
   - `healthbot.tar`, which installs the Paragon Insights and the UI components.
   - `run` script, which executes the installer image. You might need to make the run script executable by using the `chmod +x run` command.

Install Paragon Automation

1. Use the run script to create and initialize a configuration directory with the configuration template files.

   ```
   # ./run -c config-dir init
   ```

   config-dir is a user-defined directory on the control host that contains configuration information for a particular installation. The `init` command automatically creates the directory if it does not exist. Alternatively, you can create the directory before you execute the `init` command.

   If you are using the same control host to manage multiple installations of Paragon Automation, you can differentiate between installations by using differently named configuration directories.

2. Use a text editor to customize the inventory file, created under the *config-dir* directory, with the IP addresses or hostnames of the cluster nodes, as well as the usernames and authentication information that are required to connect to the nodes.

   The inventory file describes the cluster nodes on which Paragon Automation will be installed.

   ```
   # vi config-dir/inventory
   ```

   Edit the following groups in the **inventory** file.

   a. Add the IP addresses of the Kubernetes primary and worker nodes of the cluster.

The `master` group identifies the primary nodes, and the `node` group identifies the worker nodes. The same IP address cannot be in both `master` and `node` groups. If you have configured hostnames that can be resolved to the required IP addresses, you can also add hostnames in the inventory file.

For example:

```
[master]
10.12.xx.x3
10.12.xx.x4
10.12.xx.x5
# hostname of kubernetes master

[node]
10.12.xx.x6
10.12.xx.x7
10.12.xx.x8
# hostname(s) of kubernetes nodes
```

b. Define the nodes that run Elasticsearch in the `elasticsearch_cluster` group.

These nodes store the Elasticsearch hosts for log collection, and use **/var/lib/elasticsearch** to store logging data. These nodes can be the same as the worker nodes, or can be a different set of nodes. Elasticsearch uses a lot of disk space. If you use the worker nodes, you must ensure that the nodes have sufficient space for log collection.

For example:

```
[elasticsearch_cluster]
10.12.xx.x7
10.12.xx.x8
```

c. Define the nodes that have disk space available for applications under the `local_storage_nodes:children` group.

Services such as Postgres, Zookeeper, and Kafka use local storage or disk space partitioned inside **export/local-volumes**. By default, worker nodes have local storage available. If you require the primary nodes to run applications as well, add `master` to this group. If you do not add the `master` here, you can run only applications that do not require local storage on the primary nodes.

For example:

```
[local_storage_nodes:children]
node
master
```

d. Configure the user account and authentication methods to authenticate the installer with the cluster nodes under the `[all:vars]` group. Set the `ansible_user` variable to the user account to log in to the cluster. The user account must be root or in case of non-root users, the account must have superuser (`sudo`) privileges. Use any *one* of the following methods to specify user account passwords.

- Use an `ssh-key` for authentication. Configure the `ansible_ssh_private_key_file` variable in the inventory file.

  ```
  [all:vars]
  ansible_user=root
  ansible_ssh_private_key_file=config/id_rsa
  ```

  If you use an SSH key, you must perform the following steps on the control host.

  i. Generate an SSH key.

  ```
  # ssh-keygen -t rsa
  Generating public/private rsa key pair.
  Enter file in which to save the key (/root/.ssh/id_rsa):     <= ENTER (use
  default)
  Enter passphrase (empty for no passphrase):               <= ENTER (no
  passphrase)
  Enter same passphrase again:                                 <= ENTER (no
  passphrase)
  Your identification has been saved in /root/.ssh/id_rsa.
  Your public key has been saved in /root/.ssh/id_rsa.pub.
  The key fingerprint is:
  SHA256:YS8cWopND9RFnpHGqaI1Q8e5ca2fxP/yMVzZtIDINbg root@Control1
  The key's randomart image is:
  +---[RSA 2048]----+
  |     ..o *=+     |
  |    ..= *o*oo    |
  |   . .o==*+. .  .|
  |    =+oO.Eo   ..+|
  ```

```
|   o.++ So.o   oo|
|  .      .o .. . |
|             .+  |
|            . .o |
|             o.  |
+----[SHA256]-----+
```

ii. Copy the private key to the *config-dir* directory, where the inventory file is saved.

```
# cd config-dir
# cp ~/.ssh/id_rsa .
```

iii. To allow authentication using the SSH key, copy **id_rsa.pub** to the cluster nodes. Repeat this step for all cluster nodes.

```
# ssh-copy-id -i ~/.ssh/id_rsa.pub cluster-node-IP-or-hostname
```

iv. (Optional) If you want to protect the SSH key with a passphrase, you must run ssh-agent and use the `ssh-add` *config-dir*`/id_rsa` command to add the key to the agent.

- Enter the `ansible_user` name and `password` in the master and node groups of the inventory file.

```
[master]
10.12.xx.x3 ansible_user=root ansible_ssh_pass=Password
10.12.xx.x4 ansible_user=root ansible_ssh_pass=Password
10.12.xx.x5 ansible_user=root ansible_ssh_pass=Password
# hostname of kubernetes master

[node]
10.12.xx.x6 ansible_user=root ansible_ssh_pass=Password
10.12.xx.x7 ansible_user=root ansible_ssh_pass=Password
10.12.xx.x8 ansible_user=root ansible_ssh_pass=Password
# hostname(s) of kubernetes nodes
```

- Use the `ansible-vault encrypt_string` command supported by Ansible to encrypt passwords.

i. Run the `./run -c` *config-dir* `ansible-vault encrypt_string` command.

ii. Enter a vault password and confirm the password when prompted.

iii. Copy and paste the encrypted password into the inventory file.

For example:

```
# ./run -c config-dir ansible-vault encrypt_string
====================
PO-Runtime installer
====================

Supported command:
  deploy [-t tags]  deploy runtime
  destroy [-t tags] destroy runtime
  init              init configuration skeleton
  conf              basic configuration editor
  info [-mc]        cluster installation info

Starting now: ansible-vault encrypt_string
New Vault password:
Confirm New Vault password:
Reading plaintext input from stdin. (ctrl-d to end input)
V3rySecret!
!vault |
          $ANSIBLE_VAULT;1.1;AES256

623833643638336334623535343837313738383165393030303435656235373436616561373164 61

383132393133336664613164353339633432663564643963 0a333630623035306464346139643437

376433343236393638363438356539323833623362346234643737366335636561643866623162623 3

306436376161646235 0a393664383433333638343433633931326166663066646165663764663032
          6265
Encryption successful
```

In this example, the encrypted password is the text starting from "!vault |" up to and including "6265". If you are encrypting multiple passwords, enter the same password for all.

For more information, see https://docs.ansible.com/ansible/latest/user_guide/vault.html.

NOTE: The default **inventory** file is in the INI format. If you choose to encrypt passwords using the Vault method, you must convert the inventory file to the YAML format. For information about inventory files, see https://docs.ansible.com/ansible/latest/network/getting_started/first_inventory.html.

- Enter the authentication password directly in the file for `ansible_password`. We do not recommend using this option to specify the password.

```
[all:vars]
ansible_user=root
ansible_password=root-password
```

If `ansible_user` is not root, the configured user must be able to use `sudo` to execute privileged commands. If `sudo` requires a password, also add `ansible_become_password=password` to the **inventory** file. For more information about how to configure Ansible inventory, see https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html.

e. (Optional) Specify a name for your Kubernetes cluster in the `kubernetes_cluster_name` group.

3. Use the `conf` script to configure the installer.

```
# ./run -c config-dir conf
```

The `conf` script runs an interactive installation wizard that allows you to choose the components to be installed and configure a basic Paragon Automation setup. The script populates the **config.yml** file with your input configuration. For advanced configuration, you must edit the **config.yml** file manually.

Enter the information as prompted by the wizard. Use the cursor keys to move the cursor, use the space key to select an option, and use `a` or `i` to toggle selecting or clearing all options. Press Enter to move to the next configuration option. You can skip configuration options by entering a period (.). You can re-enter all your choices by exiting the wizard and restarting from the beginning. The installer allows you to exit the wizard after you save the choices that you already made or to restart from the beginning. You cannot go back and redo the choices that you already made in the current workflow without exiting the wizard altogether.

The configuration options that the `conf` script prompts for are listed in Table 4 on page 26:

Table 4: conf Script Options

`conf` Script Prompts	Description/Options
Select components	You can install one or more of the Infrastructure, Pathfinder, Insights, and base platform components. By default, all components are selected. **NOTE**: In Paragon Automation Release 21.2, you must install the base platform and Paragon Insights components. The installation of the other components is optional and based on your requirement.

Table 4: conf Script Options *(Continued)*

`conf` Script Prompts	Description/Options
Infrastructure Options	These options are displayed only if you selected to install the Infrastructure component in the previous prompt. • Install Kubernetes Cluster—Installs the required Kubernetes cluster. If you are installing Paragon Automation on an existing cluster, you can clear this selection. • Install MetalLB LoadBalancer—Installs an internal load balancer for the Kubernetes cluster. By default, this option is already selected. If you are installing Paragon Automation on an existing cluster with preconfigured load balancing, you can clear this selection. • Install Chrony NTP Client—NTP is required to synchronize the clocks of the cluster nodes. If NTP is already installed and configured, you need not install Chrony. All nodes must run NTP or some other time-synchronization at all times. • Allow Master Scheduling—Master scheduling determines how the primary nodes are used. If you select this option, the master nodes are used as both the control plane and worker nodes, which means that you can run application workloads on the primary nodes as well. Master scheduling allows for better resource allocation and management in the cluster. However, you also run the risk that a misbehaving workload can exhaust resources and affect the stability of the whole cluster. If you do not allow master scheduling, the primary nodes are used only as the control plane. You can accommodate primary nodes with lesser resources, because they do not run any application workloads. If you have multiple primary nodes or nodes with high capacity and disk space, you risk wasting their resources by not utilizing them completely. If you allow master scheduling, the primary nodes can also provide storage.
Kubernetes Master Virtual IP address	Configure a virtual IP address (VIP) for the primary nodes in a multi-primary node setup. The VIP must be in the same broadcast domain as the primary and worker nodes. **NOTE**: This option is displayed only when the inventory file is updated with more than one primary node.

Table 4: conf Script Options *(Continued)*

`conf` Script Prompts	Description/Options
Install Loadbalancer for Master Virtual IP address	Install a load balancer for clusters with multiple primary nodes. The load balancer is responsible for the primary node's VIP and is not used for externally accessible services. By default, the load balancer is internal, but you can also use external load balancers. For more information, see https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/high-availability/#create-load-balancer-for-kube-apiserver.
List of NTP servers	Enter a comma-separated list of NTP servers.
LoadBalancer IP address ranges	Enter a comma-separated list of IP addresses or address ranges that are reserved for the load balancer. The externally accessible services are handled through MetalLB, which needs one or more IP address ranges that are accessible from outside the cluster. VIPs for the different servers are selected from these ranges of addresses. The address ranges can be (but need not be) in the same broadcast domain as the cluster nodes. For ease of management, because the network topologies need access to Insights services and the PCE server clients, we recommend that the VIPs for these be selected from the same range. For more information, see "VIP Considerations" on page 18. **NOTE**: Do not include the Kubernetes Master Virtual IP address. Addresses can be entered as comma separated values, or as a range, or as a combination of both. For example: • 10.x.x.1, 10.x.x.2, 10.x.x.3 • 10.x.x.1-10.x.x.3 • 10.x.x.1, 10.x.x.3-10.x.x.5 • 10.x.x.1-3 is not a valid format
Virtual IP address for ingress controller	Enter a VIP to be used for Web access of the Kubernetes cluster or the Paragon Automation user interface. This must be an unused IP address that is managed by the load balancer.

Table 4: conf Script Options *(Continued)*

`conf` Script Prompts	Description/Options
Virtual IP address for Insights services	Enter a VIP for Paragon Insights services. This must be an unused IP address that is managed by the load balancer.
Virtual IP address for Pathfinder PCE server	Enter a VIP to be used for Paragon Pathfinder PCE server access. This must be an unused IP address that is managed by the load balancer. **NOTE**: The addresses for ingress controller, Insights services, and PCE server must be unique. You cannot use the same address for all three VIPs. Ensure that these three addresses are listed in the LoadBalancer IP address ranges option.
Hostname of Main web application	Enter a hostname or an IP address. If you enter an IP address, it must be the same as the VIP that you entered for the ingress controller. If you enter a hostname, it should resolve to the VIP for the ingress controller and must be preconfigured in the DNS server. This address or hostname is used to access the Paragon Automation Web UI from your browser. For example, https:// *hostname* or https:// *IP-address*.
BGP autonomous number of CRPD peer	Set up the Containerized Routing Protocol Daemon (cRPD) autonomous systems and the nodes with which cRPD creates its BGP sessions. If Paragon Pathfinder is installed, you must configure a cRPD to peer with a BGP-LS router in the network to import the network topology. For a single autonomous system, you must configure the ASN of the network. **NOTE**: While you can configure the correct ASN at the time of installation, you can also modify the cRPD configuration later.

Table 4: conf Script Options *(Continued)*

`conf` Script Prompts	Description/Options

Comma separated list of CRPD peers	List of CRPD peers. The CRPD instance running as part of a cluster opens a BGP connection to the specified peer routers and imports topology data using BGP-LS.
	
	NOTE: While you can configure the correct peer IP addresses at the time of installation, you can also modify the cRPD configuration later.
	
	If more than one peer is required, these can be added as comma separated values or as a range or as a combination of both, similar to LoadBalancer IP addresses.
	
	The following example shows the configuration of the BGP peer in the connected network topology. The example shows the configuration of a Juniper device active as BGP-LS peer to provide topology information to Paragon Pathfinder.
	
	`[edit groups northstar]`
	`root@system# show protocols bgp group northstar`
	`type internal;`
	`family traffic-engineering {`
	`  unicast;`
	`}`
	`export TE;`
	`allow 10.xx.43.0/24;`
	
	`[edit groups northstar]`
	`root@system# show policy-options policy-statement TE`
	`from family traffic-engineering;`
	`then accept;`
	
	In this example, the cluster hosts are in the 10.xx.43.0/24 network, and the router will accept BGP sessions from any host in this network.
	
	You can also configure the specific IP addresses of the worker nodes combined with the passive option. For example:
	
	`[edit protocols bgp group BGP-LS]`
	`root@vmx101# show | display set`
	`set protocols bgp group BGP-LS family traffic-engineering unicast`
	`set protocols bgp group BGP-LS peer-as 11`
	`set protocols bgp group BGP-LS allow 10.x.43.1`
	`set protocols bgp group BGP-LS allow 10.x.43.2`

Table 4: conf Script Options *(Continued)*

conf Script Prompts	Description/Options

	`set protocols bgp group BGP-LS allow 10.x.43.3`
	`set protocols bgp group BGP-LS passive`
	`set protocols bgp group BGP-LS export TE`
	
	The BGP session is initiated by cRPD. Only one session is established at a time and is initiated using the address of the worker node currently running cRPD. If you choose to configure the specific IP addresses instead of using the allow option, configure the addresses of all the workers nodes for redundancy.
	
	You will also need to enable OSPF/ISIS and MPLS traffic engineering as shown:
	
	`set protocols isis traffic-engineering igp-topology`
	`Or`
	`set protocols ospf traffic-engineering igp-topology`
	
	`set protocols mpls traffic-engineering database import igp-topology`
	
	For more information, see https://www.juniper.net/documentation/us/en/software/junos/mpls/topics/topic-map/mpls-traffic-engineering-configuration.html.

4. Click `Yes` to save the configuration information.

   This configures a basic setup, and the information is saved in the **config.yml** file in the *config-dir* directory.

5. (Optional) For more advanced configuration of the cluster, use a text editor to manually edit the **config.yml** file.

   The **config.yml** file consists of an essential section at the beginning of the file that corresponds to the configuration options that the installation wizard prompts you to enter. The file also has an extensive list of sections under the essential section that allows you to enter complex configuration values directly in the file.

   Consider editing the following configuration options:

   - By default, the `install_opendistro_es` is set to true to replace the Elasticsearch version with Open Distro. Open Distro provides authentication to log in to the Kibana application. The username is preconfigured as admin in `#opendistro_es_admin_user: admin`.

     You must configure the authentication password, `# opendistro_es_admin_password: "`*password*`"`. Use admin as username and this password to log in to Kibana.

- Set the `iam_skip_mail_verification` configuration option to true for user management without SMTP by Identity Access Management (IAM). For user management with SMTP, set the option to false.

Save and exit the file after you finish editing it.

6. (Optional) If you want to deploy custom SSL certificates signed by a recognized certificate authority (CA), store the private key and certificate in the *config-dir* directory. Save the private key as **ambassador.key.pem** and the certificate as **ambassador.cert.pem**.

   By default, ambassador uses a locally generated certificate signed by the Kubernetes cluster-internal CA.

   NOTE: If the certificate is about to expire, save the new certificate as **ambassador.cert.pem** in the same directory, and execute the `./run -c config-dir deploy -t ambassador` command.

7. Install the Paragon Automation cluster based on the information that you configured in the **config.yml** and **inventory** files.

   ```
   # ./run -c config-dir deploy
   ```

   The time taken to install the configured cluster depends on the complexity of the cluster. A basic setup installation takes at least 30 minutes to complete.

   If you are installing Paragon Automation on an existing Kubernetes cluster, the `deploy` script upgrades the currently deployed cluster to the latest Kubernetes version. The script also upgrades the Docker CE version, if required. If Docker EE is already installed on the nodes, the deploy script will not overwrite it with Docker CE. When upgrading the Kubernetes version or the Docker version, the script performs the upgrade sequentially on one node at a time. Each node is cordoned off and removed from scheduling and upgrades are performed, Kubernetes is restarted on the node, and the node is finally uncordoned and brought back into scheduling.

8. Log in to the worker nodes.

   Use a text editor to configure the following recommended information for Paragon Insights in the **limits.conf** and **sysctl.conf** files.

   a.

   ```
   # vi /etc/security/limits.conf
                                # End of file
                                *         hard    nofile      1048576
                                *         soft    nofile      1048576
                                root      hard    nofile      1048576
                                root      soft    nofile      1048576
   ```

```
                                influxdb  hard    nofile      1048576
                                influxdb  soft    nofile      1048576
```

b.

```
# vi /etc/sysctl.conf
                                fs.file-max = 2097152
                                vm.max_map_count=262144
                                fs.inotify.max_user_watches=524288
                                fs.inotify.max_user_instances=512
```

Repeat this step for all worker nodes.

Log in to the Paragon Automation UI

After you install Paragon Automation, log in to the Paragon Automation UI.

1. Open a browser, and enter either the hostname of the main Web application or the VIP of the ingress controller that you entered in the URL field of the installation wizard.

   For example, https://vip-of-ingress-controller-or-hostname-of-main-web-application. The Paragon Automation login page is displayed.

2. For first-time access, enter **admin** as username and **Admin123!** as the password to log in. You must change the password immediately.

   The **Set Password** page is displayed. To access the Paragon Automation setup, you must set a new password.

3. Set a new password that meets the password requirements.

   The password should be between 6 to 20 characters and must be a combination of uppercase letters, lowercase letters, numbers, and special characters. Confirm the new password, and click **OK**.

   The **Dashboard** page is displayed. You have successfully installed and logged in to the Paragon Automation UI.

4. Update the URL to access the Paragon Automation UI in **Administration > Authentication > Portal Settings** to ensure that the activation e-mail sent to users for activating their account contains the correct link to access the GUI. For more information, see *Configure Portal Settings*.

RELATED DOCUMENTATION

Installation Prerequisites | 11

Paragon Automation GUI Overview

Backup and Restore | 40

Troubleshoot Paragon Automation Installation | 48

Migrate Data from Northstar to Paragon Automation | 61

Update Paragon Automation

To reinstall Paragon Automation, run the `deploy` script again on the control host.

To update an existing instance of Paragon Automation, edit the **inventory** and **config.yml** files, and run the `deploy` script again on the control host.

```
# ./run -c config-dir deploy
```

If the `deploy` script fails for a particular component, you can run the `destroy` command to uninstall the component, and then reinstall it with the `deploy` script.

```
# ./run -c config-dir destroy -t tags
# ./run -c config-dir deploy -t tags
```

The following optional parameters are supported for the `deploy` script:

- `--list-tags`—View a list of available tags.
- `-t tag1,tag2`—Deploy or redeploy a subset of the installation tasks or components of the cluster selectively. For example, to install or update only the infrastructure component, use `# ./run -c config-dir deploy -t infra`.
- `--skip-tags tag1,tag2`—Skip over some installation tasks. For example, to deploy the cluster without installing the Paragon Insights component, use `# ./run -c config-dir deploy --skip-tags healthbot`.
- `--ask-vault-pass`—Prompt for the password to decrypt authentication passwords, if Ansible vault was previously configured.

RELATED DOCUMENTATION

Installation Prerequisites | 11

Install Paragon Automation | 20

Uninstall Paragon Automation | 37

Edit Cluster Nodes

IN THIS SECTION

Use the information provided in this topic to edit cluster nodes.

Add a Node

You can edit an operational Paragon Automation cluster and add additional nodes to the cluster. The node can be either a primary or worker node.

However, note that if your existing cluster is configured with a single primary node, you cannot add an additional primary node using this procedure. To add an additional primary node to a single primary node cluster, you must reinstall the whole cluster.

To add new nodes:

1. Prepare the node and ensure it meets all the cluster node prerequisites as described in "Prepare Cluster Nodes" on page 14.
2. Log in to the control host.
3. Edit the inventory file with the IP address or hostname of the new node. Add the node to the required cluster node group (primary or worker).
4. Execute the `./run -c` *`config-dir`* `deploy` command.

   You can also limit the execution of the command to deploy only the new node by using the `./run -c` *`config-dir`* `deploy -l` *`node-address`* command.

Remove a Node

You can edit an operational Paragon Automation cluster to remove one or more nodes from the cluster.

To remove nodes:

1. Access kubectl.

   The main interface in the Kubernetes cluster is kubectl which is installed on a primary node. However, you can also access the Kubernetes API from any node that has access to the cluster, including the control host. To use a node other than the primary node, you must ensure that you copy the **admin.conf** file and set the `kubeconfig` environment variable, or you can use the `export KUBECONFIG=config-dir/admin.conf` command.

   For more information on `kubectl` commands, see https://kubernetes.io/docs/reference/kubectl/overview/.

2. Execute the following `kubectl` commands to remove a node.

   ```
   root@primary-node:~# kubectl cordon node-address
   root@primary-node:~# kubectl drain node-address
   root@primary-node:~# kubectl delete node-address
   ```

3. Log in to the control host. Update the inventory file to delete the node so that the inventory file also reflects the change. However, do *not* execute the `./run -c config-dir deploy` command again.

Replace a Node

You can replace a node with another node in an existing Paragon Automation cluster.

To replace a node with another node:

1. Prepare the node and ensure it meets all the cluster node prerequisites as described in "Prepare Cluster Nodes" on page 14.
2. Access `kubectl`.

   The main interface in the Kubernetes cluster is `kubectl` which is installed on a primary node. However, you can also access the Kubernetes API from any node that has access to the cluster, including the control host. To use a node other than the primary node, you must ensure that you copy the **admin.conf** file and set the `kubeconfig` environment variable, or you can use the `export KUBECONFIG=config-dir/admin.conf` command.

   For more information on `kubectl` commands, see https://kubernetes.io/docs/reference/kubectl/overview/.

3. Run the following `kubectl` commands to remove a node.

```
root@primary-node:~# kubectl cordon node-address
root@primary-node:~# kubectl drain node-address
root@primary-node:~# kubectl delete node-address
```

4. Log in to the control host.
5. Edit the inventory file with the IP address or hostname of the new node. Add the node to the required cluster node group (primary or worker). Also, delete the node you replaced.
6. Run the `./run -c config-dir deploy` command.

   You can also limit the execution of the command to deploy only the new node by using the `./run -c config-dir deploy -l node-address` command.

In some cases, if a node fails, you can rebuild a replacement node using the same IP address as the failed node.

To redeploy the same node as a replacement, perform the following steps:

1. Prepare the node and ensure it meets all the cluster node prerequisites as described in "Prepare Cluster Nodes" on page 14.
2. Log in to the control host.
3. Run the `./run -c config-dir deploy` command again.

   You can also limit the execution of the command to only the redeployed node by using the `./run -c config-dir deploy -l node-address` command.

RELATED DOCUMENTATION

Installation Prerequisites | 11

Install Paragon Automation | 20

Update Paragon Automation | 34

Uninstall Paragon Automation

To uninstall Paragon Automation:

1. Log in to the control host.

2. Uninstall individual components or component groups:

```
# ./run -c config-dir destroy -t tags
```

To view a list of available tags, use `# ./run -c config-dir deploy --list-tags`.

If you uninstall Paragon Automation completely, you must also ensure that the **/var/lib/rook** directory is removed on all nodes, and all Ceph block devices are wiped. For information on clearing Ceph block devices, see "Reformat a Disk" on page 55.

> **NOTE**: To completely uninstall the whole cluster, we recommend that you re-image all the cluster nodes. Re-imaging is a faster and more complete option.

RELATED DOCUMENTATION

Install Paragon Automation | 20

Update Paragon Automation | 34

Edit Cluster Nodes | 35

4

CHAPTER

Backup and Restore

Backup and Restore

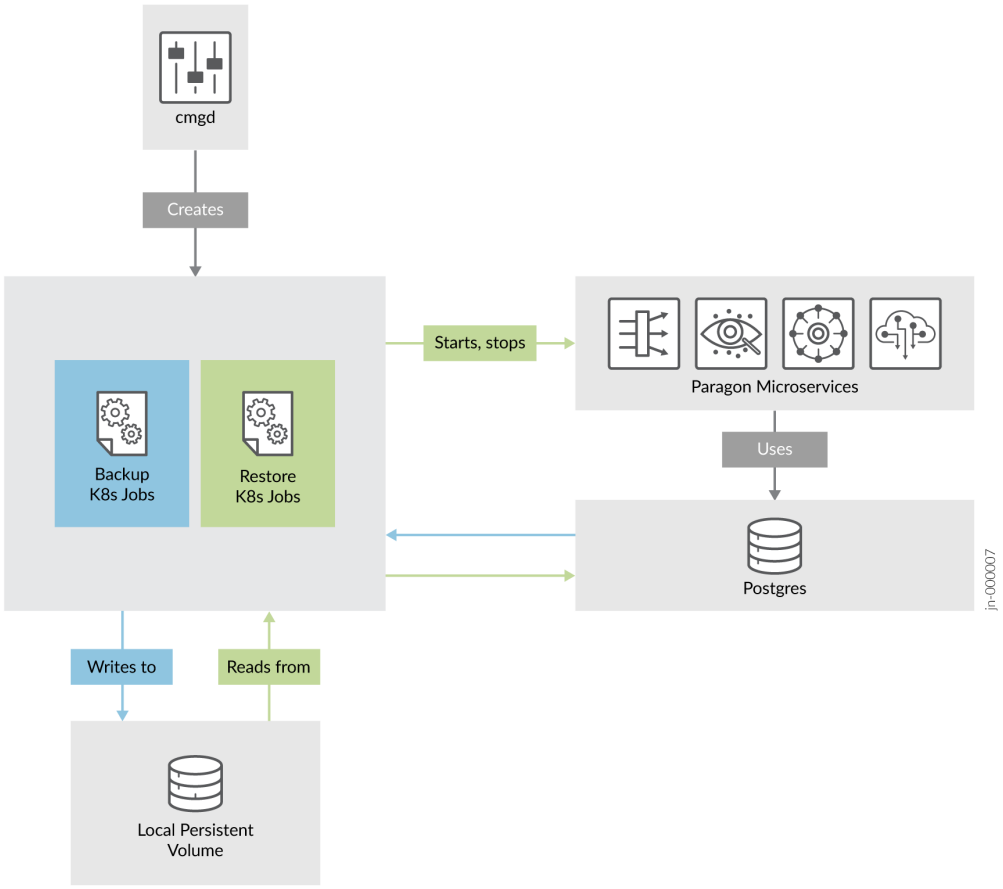
IN THIS SECTION

This topic describes the backup and restore capabilities available in Paragon Automation. Although Paragon Automation is a GUI-based application, the backup and restore operations are managed from the Paragon Insights cMGD CLI. Postgres is the primary persistent storage database for microservices. Backup files are saved in a local persistent volume on the cluster nodes. The backup procedure can be performed while microservices are running and does not affect the operation of the cluster. However, for restore procedures, microservices are stopped and the cluster is not functional until the databases are restored.

Currently, you cannot schedule a backup or restore procedure. Also, you cannot select applications to be backed-up and restored. A preconfigured and fixed set of applications are backed-up and restored for each component.

The backup and restore procedures are implemented by containerized scripts that are invoked through Kubernetes jobs.

Figure 5: Backup and Restore Process



Back up the Configuration

Data across most Paragon Automation applications is primarily stored in Postgres. When you back up a configuration, system determined predefined data is backed-up. When you perform a backup, the operational system and microservices are not affected. You can continue to use Paragon Automation while a backup is running.

To back up the current Paragon Automation configuration:

1. Determine and log in to the cMGD CLI managed by Paragon Insights (formerly Healthbot).

For example:

```
root@primary-node:~# kubectl get -n healthbot pods -l app=mgd
NAME                   READY   STATUS    RESTARTS   AGE
mgd-57b5754b7f-26mlm   1/1     Running   0          10d
root@primary-node:~# kubectl exec -it -n healthbot mgd-57b5754b7f-26mlm -- bash
root@primary-node:~# cli
```

NOTE: The main interface in the Kubernetes cluster is kubectl which is installed on a primary node. However, you can also access the Kubernetes API from any node that has access to the cluster, including the control host. To use a node other than the primary node, you must ensure that you copy the **admin.conf** file and set the kubeconfig environment variable, or you can use the export KUBECONFIG=config-dir/admin.conf command.

2. Enter the `request system backup path` *`path-to-backup-folder`* command to start a backup job which backs up all databases up until the moment you run the command.

   For example:

   ```
   root@mgd-57b5754b7f-26mlm> request system backup path /hello/world/
   ```

   A corresponding Kubernetes `db-backup-hello-world` job is created. The Kubernetes job creates a backup of the predefined data. The files are stored in a local persistent volume.

3. After backup is complete, you must explicitly and manually back up the base platform resources using kubectl.

   a. Back up **jobmanager-identitysrvcreds** and **devicemodel-connector-default-scope-id**:

   ```
   root@primary-node:~# kubectl get secrets -n ems jobmanager-identitysrvcreds devicemodel-
   connector-default-scope-id -o yaml > ems-scope-bkup.yaml
   ```

   b. (Optional) If SMTP is configured on the Paragon Automation cluster, then back up the available `iam-smtp-config` secret:

   ```
   root@primary-node:~# kubectl get secrets -n common iam-smtp-config -o yaml > iam-smtp-
   bkup.yaml
   ```

   If this command fails, then SMTP is not configured in the cluster and you can ignore the error.

Frequently Used kubectl Commands to View Backup Details

To view the status of your backup or the location of your backup files, or to view more information on the backup files, use the following commands.

- Backup jobs exist in the common namespace and use the `common=db-backup` label. To view all backup jobs:

```
root@primary-node:~# kubectl get -n common jobs -l common=db-backup
NAME            COMPLETIONS   DURATION   AGE
db-backup-hello-world   1/1           3m11s      2d20h
```

- To view more details of a specific Kubernetes job:

```
root@primary-node:~# kubectl describe -n common jobs/db-backup-hello-world
```

- To view the logs of a specific Kubernetes job:

```
root@primary-node:~# kubectl logs -n common --tail 50 jobs/db-backup-hello-world
```

- To determine the location of the backup files:

```
root@primary-node:~# kubectl get -n common pvc db-backup-pvc
NAME            STATUS   VOLUME              CAPACITY   ACCESS MODES   STORAGECLASS    AGE
db-backup-pvc   Bound    local-pv-cb20f386   145Gi      RWO            local-storage   3d3h
```

  The output points you to the local persistent volume. Use that persistent volume to determine the node on which the backup files are stored.

```
root@primary-node:~# kubectl describe -n common pv local-pv-cb20f386
Node Affinity:
 Required Terms:
    Term 0:        kubernetes.io/hostname in [10.49.xxx.x2]
Message:
Source:
   Type:  LocalVolume (a persistent volume backed by local storage on a node)
   Path:  /export/local-volumes/pv*
```

To view all the backup files, log in to the node and navigate to the location of the backup folder.

```
root@primary-node:~# ssh root@10.49.xxx.x2
root@10.49.xxx.x2:~# ls -l /export/local-volumes/pv*
```

To view commonly seen backup and restore failure scenarios, see "Common Backup and Restore Issues" on page 49.

Restore the Configuration

You can restore a Paragon Automation configuration from a previously backed-up configuration folder. A restore operation rewrites the databases with all the backed-up configuration information. You cannot selectively restore databases. When you perform a restore, a Kubernetes job is spawned, which will stop the affected microservices. The job restores the backed-up configuration, and restarts the microservices. Paragon Automation is not functional until the restoration procedure is complete.

You cannot run multiple restore jobs at the same time, since microservices are stopped during the restoration process. Also, you cannot run both backup and restore processes concurrently.

NOTE: We strongly recommend that you restore a configuration during a maintenance window, else the system can go into an inconsistent state.

To restore the Paragon Automation configuration to a previously backed-up configuration:

1. Log in to the cMGD CLI managed by Paragon Insights (formerly Healthbot).

   For example:

   ```
   root@primary-node:~# kubectl get -n healthbot pods -l app=mgd
   NAME                   READY   STATUS    RESTARTS   AGE
   mgd-57b5754b7f-26mlm   1/1     Running   0          10d
   root@primary-node:~# kubectl exec -it -n healthbot mgd-57b5754b7f-26mlm -- bash
   root@primary-node:~# cli
   ```

2. Enter the `request system restore path` *`path-to-backup-folder`* command to restore the configuration with the files in the specified backup folder on the persistent volume.

For example:

```
root@mgd-57b5754b7f-26mlm> request system restore path /hello/world/
```

A corresponding Kubernetes `db-restore-hello-world` job is created. The restore process takes longer than a backup process since microservices are stopped and restarted. When the restoration is complete, the Paragon Automation system is not operational immediately. You must wait around ten minutes for the system to stabilize and become fully functional.

NOTE: If you are logged in during the restore process, you must log out and log back in after the restore process is complete.

3. After restore is complete, you must explicitly restore the base platform resources with the previously manually backed-up base-platform backup files.

   a. Delete the **jobmanager-identitysrvcreds** and **devicemodel-connector-default-scope-id** base platform secrets resources.

   ```
   root@primary-node:~# kubectl delete secrets -n ems jobmanager-identitysrvcreds devicemodel-connector-default-scope-id
   ```

   b. Restore the previously backed-up base platform resources.

   ```
   root@primary-node:~# kubectl apply -f ems-scope-bkup.yaml
   ```

   c. Restart the `jobmanager` and `devicemodel-connector` base platform services.

   ```
   root@primary-node:~# kubectl rollout restart deploy jobmanager devicemodel-connector -n ems
   ```

   d. (Optional) If SMTP is configured on the Paragon Automation cluster, delete the current SMTP secrets file and restore from the previously backed-up file.

   ```
   root@primary-node:~# kubectl delete secret -n common iam-smtp-config
   root@primary-node:~# kubectl apply -f iam-smtp-bkup.yaml
   ```

e. (Optional) Delete the manually backed-up files.

```
root@primary-node:~# rm ems-scope-bkup.yaml iam-smtp-bkup.yaml
```

Frequently Used kubectl Commands to View Restore Details

To view more information and the status of your restore process, use the following commands.

- Restore jobs exist in the common namespace and use the `common=db-restore` label. To view all restore jobs:

```
root@primary-node:~# kubectl get -n common jobs -l common=db-restore
NAME                     COMPLETIONS   DURATION   AGE
db-restore-hello-world   0/1           20s        21s
```

- To view more details of a specific Kubernetes job:

```
root@primary-node:~# kubectl describe -n common jobs/db-restore-hello-world
```

- To view the logs of a particular Kubernetes job:

```
root@primary-node:~# kubectl logs -n common --tail 50 jobs/db-restore-hello-world
```

To view commonly seen backup and restore failure scenarios, see "Common Backup and Restore Issues" on page 49.

RELATED DOCUMENTATION

Troubleshoot Paragon Automation Installation | 48

Update Paragon Automation | 34

Uninstall Paragon Automation | 37

5
CHAPTER

Troubleshooting

Troubleshoot Paragon Automation Installation

SUMMARY

This topic provides a general guide to troubleshooting some typical problems you might encounter during and after installation.

IN THIS SECTION

Resolve Merge Conflicts of the Configuration File

The `init` script creates the template configuration files. If you update an existing installation using the same *config-dir* directory that was used for the installation, the template files that the `init` script creates are merged with the existing configuration files. Sometimes, this merging action creates a merge conflict that you must resolve. The script prompts you about how to resolve the conflict. When prompted, select one of the following options:

- C—You can retain the existing configuration file and discard the new template file. This is the default option.
- n—You can discard the existing configuration file and reinitialize the template file.
- m—You can merge the files manually. Conflicting sections are marked with lines starting with "<<<<<<<<", "||||||||", "========", and ">>>>>>>>". You must edit the file and remove the merge markers before you proceed with the update.
- d—You can view the differences between the files before you decide how to resolve the conflict.

Common Backup and Restore Issues

In a scenario when you destroy an existing cluster and redeploy a software image on the same cluster nodes, if you try to restore a configuration from a previously backed up configuration folder, the restore operation might fail. Restore fails because the mount path for the backed up configuration is now changed. When you destroy an existing cluster, the persistent volume is deleted. When you redeploy the new image, the persistent volume gets recreated in one of the cluster nodes wherever space is available, but not necessarily in the same node as it was present in previously. Hence, the restore operation fails.

As a workaround:

1. Determine the mount path of the new persistent volume.
2. Copy the contents of the previous persistent volume's mount path to the new path.
3. Retry the restore operation.

View Installation Log Files

If the `deploy` script fails, you must check the installation log files in the *config-dir* directory. By default, the *config-dir* directory stores six zipped log files. The current log file is saved as **log**, and the previous log files are saved as **log.1** through **log.5** files. Every time you run the `deploy` script, the current log is saved, and the oldest one is discarded.

Error messages are typically found at the end of a log file. View the error message, and fix the configuration.

View Log Files in Kibana

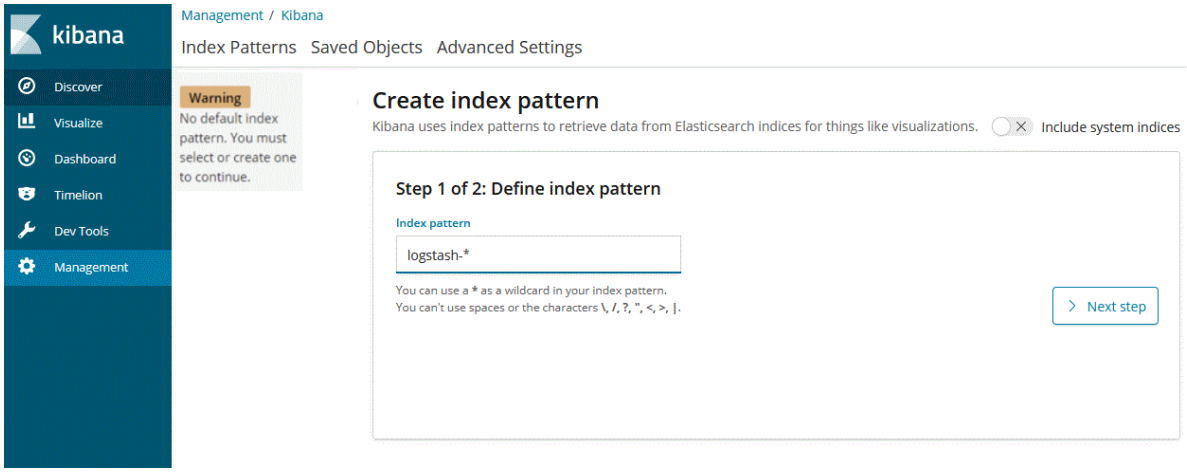
System logs are stored in Elasticsearch, and can be accessed through the Kibana application. To view logs in Kibana:

1. To access the Kibana application use the VIP of the ingress controller. Open a browser, and enter https:// *vip-of-ingress-controller-or-hostname-of-main-web-application*/kibana, in the URL field.
2. Enter the `opendistro_es_admin_user` username and the `opendistro_es_admin_password` password that you configured in the **config.yml** file during installation. The default username is **admin**.

   You cannot log in to Kibana if you have not configured a password before installation.

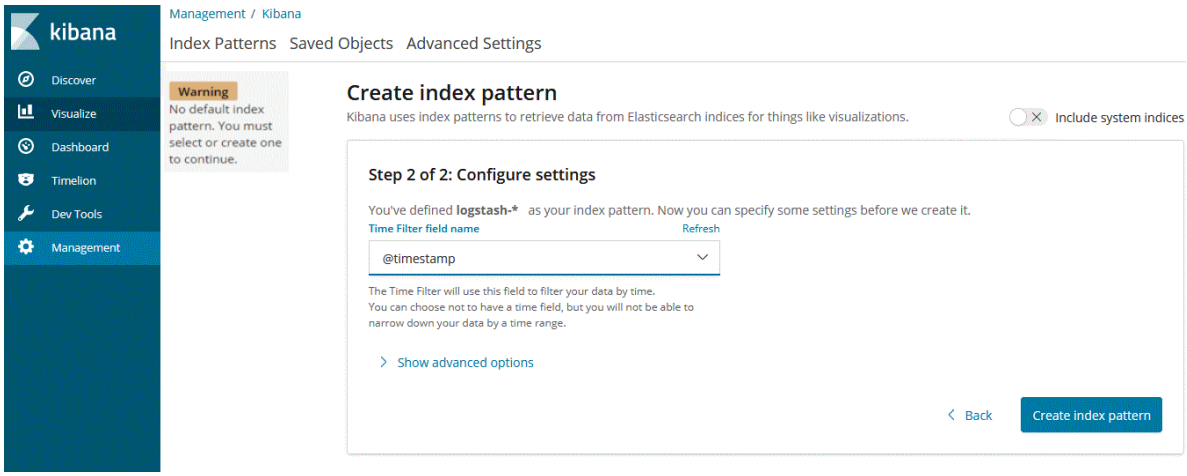
3. If you are logging in for the first time, create an index pattern by navigating to **Management > Index Pattern**.
4. Enter **logstash-*** in the **Index pattern** field and then click **> Next Step**.

Figure 6: Kibana - Define Index Pattern



5. Select **@timestamp** from the **Time Filter field name** list, and then click **Create index pattern** to create an index pattern.

Figure 7: Kibana - Configure Settings



6. Use **Discover** to browse the log files, and to add or remove filters as required.

Troubleshooting Using the kubectl Interface

IN THIS SECTION

The main interface in the Kubernetes cluster is kubectl, which is installed on a primary node. You can log in to the primary node and use the kubectl interface to access the Kubernetes API, view node details, and perform basic troubleshooting actions. The **admin.conf** file is copied to the *config-dir* directory on the control host as part of the installation process.

You can also access the Kubernetes API from any other node that has access to the cluster. To use a node other than the primary node, you must copy the **admin.conf** file and set the `kubeconfig` environment variable. Another option is to use the `export KUBECONFIG=`*`config-dir`*`/admin.conf` command.

Use the following sections to troubleshoot and view installation details using the kubectl interface.

View Node Status

Use the `kubectl get no` command to view the status of the cluster nodes. The status of the nodes must be Ready, and the roles should be either control-plane or none. For example:

```
root@primary-node:~# kubectl get no
NAME           STATUS   ROLES                  AGE    VERSION
10.49.xx.x1    Ready    control-plane,master   5d5h   v1.20.4
10.49.xx.x6    Ready    <none>                 5d5h   v1.20.4
10.49.xx.x7    Ready    <none>                 5d5h   v1.20.4
10.49.xx.x8    Ready    <none>                 5d5h   v1.20.4
```

If a node is not Ready, verify whether the kubelet process is running. You can also use the system log of the node to investigate the issue.

View Pod Status

Use the `kubectl get po -n namespace | -A` command to view the status of a pod. You can specify an individual namespace (such as healthbot, northstar, and common) or you can use the `-A` parameter to view the status of all namespaces. For example:

```
root@primary-node:~# kubectl get po -n northstar
NAME                                READY   STATUS      RESTARTS   AGE
bmp-854f8d4b58-4hwx4                3/3     Running     1          30h
dcscheduler-55d69d9645-m9ncf        1/1     Running     1          7h13m
```

The status of healthy pods must be displayed as Running or Completed, and the number of ready containers should match the total. If the status of a pod is not Running or if the number of containers does not match, use the `kubectl describe po` command to troubleshoot the issue further.

View Detailed Information about a Pod

Use the `kubectl describe po -n namespace pod-name` command to view detailed information about a specific pod. For example:

```
root@primary-node:~# kubectl describe po -n northstar bmp-854f8d4b58-4hwx4
Name:         bmp-854f8d4b58-4hwx4
Namespace:    northstar
Priority:     0
Node:         10.49.xx.x1/10.49.xx.x1
Start Time:   Mon, 10 May 2021 07:11:17 -0700
Labels:       app=bmp
              northstar=bmp
              pod-template-hash=854f8d4b58
…
```

View the Logs for a Container in a Pod

Use the `kubectl logs -n namespace pod-name [-c container-name]` command to view the logs for a particular pod. If a pod has multiple containers, you must specify the container for which you want to view the logs. For example:

```
root@primary-node:~# kubectl logs -n common atom-db-0 | tail -3
2021-05-31 17:39:21.708 36 LOG {ticks: 0, maint: 0, retry: 0}
```

```
2021-05-31 17:39:26,292 INFO: Lock owner: atom-db-0; I am atom-db-0
2021-05-31 17:39:26,350 INFO: no action.  i am the leader with the lock
```

Run a Command on a Container in a Pod

Use the `kubectl exec -ti -n` *`namespace pod-name`* `[-c` *`container-name`*`] --` *`command-line`* command to run commands on a container inside a pod. For example:

```
root@primary-node:~# kubectl exec -ti -n common atom-db-0 -- bash


 ____        _ _
/ ___| _ __ (_) | ___
\___ \| '_ \| | |/ _ \
 ___) | |_) | | | (_) |
|____/| .__/|_|_|\___/
      |_|

This container is managed by runit, when stopping/starting services use sv

Examples:

sv stop cron
sv restart patroni

Current status: (sv status /etc/service/*)

run: /etc/service/cron: (pid 29) 26948s
run: /etc/service/patroni: (pid 27) 26948s
run: /etc/service/pgqd: (pid 28) 26948s
root@atom-db-0:/home/postgres#
```

After you run `exec` the command, you get a bash shell into the Postgres database server. You can access the bash shell inside the container, and run commands to connect to the database. Not all containers provide a bash shell. Some containers provide only SSH, and some containers do not have any shells.

View Services

Use the `kubectl get svc namespace | -A` command to view the cluster services. You can specify an individual namespace (such as healthbot, northstar, and common), or you can use `-A` parameter to view the services for all namespaces. For example:

```
root@primary-node:~# kubectl get svc -A --sort-by spec.type
NAMESPACE              NAME                              TYPE           EXTERNAL-IP    PORT(S)
…
healthbot              tsdb-shim                         LoadBalancer   10.54.xxx.x3
8086:32081/TCP
healthbot              ingest-snmp-proxy-udp             LoadBalancer   10.54.xxx.x3   162:32685/
UDP
healthbot              hb-proxy-syslog-udp               LoadBalancer   10.54.xxx.x3   514:31535/
UDP
ems                    ztpservicedhcp                    LoadBalancer   10.54.xxx.x3   67:30336/
UDP
ambassador             ambassador                        LoadBalancer   10.54.xxx.x2   80:32214/
TCP,443:31315/TCP,7804:32529/TCP,7000:30571/TCP
northstar              ns-pceserver                      LoadBalancer   10.54.xxx.x4
4189:32629/TCP
…
```

In this example, the services are sorted by type, and only services of type LoadBalancer are displayed. You can view the services that are provided by the cluster and the external IP addresses that are selected by the load balancer to access those services.

Frequently Used kubectl Commands

- List the replication controllers:

```
# kubectl get -n namespace deploy
```

```
# kubectl get -n namespace statefulset
```

- Restart a component:

```
kubectl rollout restart -n namespace deploy deployment-name
```

- Edit a Kubernetes resource: You can edit a deployment or any Kubernetes API object, and these changes are saved to the cluster. However, if you reinstall the cluster, these changes are not preserved.

```
# kubectl edit -ti -n namespace deploy deployment-name
```

Troubleshooting Ceph and Rook

Ceph requires relatively newer Kernel versions. If your Linux kernel is very old, consider upgrading or reinstalling a new one.

Use this section to troubleshoot issues with Ceph and Rook.

Insufficient Disk Space

A common reason for installation failure is that the object storage daemons (OSDs) are not created. An OSD configures the storage on cluster node. OSDs might not be created because of non-availability of disk resources, in the form of either insufficient resources or incorrectly partitioned disk space. Ensure that the nodes have sufficient unpartitioned disk space available.

Reformat a Disk

Examine the logs of the "rook-ceph-osd-prepare-hostname-*" jobs. The logs are descriptive and if you need to reformat the disk or partition, and restart Rook, perform the following steps.

1. Use one of the following methods to reformat an existing disk or partition.
   - If you have a block storage device that should have been used for Ceph, but wasn't used because it was in an unusable state, you can reformat the disk completely.

```
$ sgdisk -zap /dev/disk
$ dd if=/dev/zero of=/dev/disk bs=1M count=100
```

- If you have a disk partition that should have been used for Ceph, you can clear the data on the partition completely.

```
$ wipefs -a -f /dev/partition
$ dd if=/dev/zero of=/dev/partition bs=1M count=100
```

NOTE: These commands completely reformat the disk or partitions that you are using and you will lose all data on them.

2. Restart Rook to save the changes and reattempt the OSD creation process.

```
$ kubectl rollout restart deploy -n rook-ceph rook-ceph-operator
```

View Pod Status

To check the status of Rook and Ceph pods installed in the rook-ceph namespace use the `# kubectl get po -n rook-ceph` command. The following pods must be in the `running` state.

- `rook-ceph-mon-*`—Typically three monitor pods are created
- `rook-ceph-mgr-*`—One manager pod
- `rook-ceph-osd-*`—Three or more OSD pods
- `rook-ceph-mds-cephfs-*`—Metadata servers
- `rook-ceph-rgw-object-store-*`—ObjectStore gateway
- `rook-ceph-tools*`—For additional debugging options.
  To connect to the toolbox, use the command:

  ```
  $ kubectl exec -ti -n rook-ceph $(kubectl get po -n rook-ceph -l app=rook-ceph-tools \ -o jsonpath={..metadata.name}) -- bash
  ```

  Some of the common commands you can use in the toolbox are:

  ```
  # ceph status # ceph osd status, # ceph osd df, # ceph osd utilization, # ceph osd pool stats, # ceph osd tree, and # ceph pg stat
  ```

Repair a Failed Disk

Check the status of pods installed in the rook-ceph namespace.

```
# kubectl get po -n rook-ceph
```

If a `rook-ceph-osd-*` pod is in the `Error` or `CrashLoopBackoff` state, then you must repair the disk.

1. Stop the `rook-ceph-operator`.

   ```
   # kubectl scale deploy -n rook-ceph rook-ceph-operator --replicas=0
   ```

2. Remove the failing OSD processes.

   ```
   # kubectl delete deploy -n rook-ceph rook-ceph-osd-number
   ```

3. Connect to the toolbox.

   ```
   $ kubectl exec -ti -n rook-ceph $(kubectl get po -n rook-ceph -l app=rook-ceph-tools \ -o
   jsonpath={..metadata.name}) -- bash
   ```

4. Identify the failing OSD.

   ```
   # ceph osd status
   ```

5. Mark out the failed OSD.

   ```
   [root@rook-ceph-tools-/]# ceph osd out 5
   marked out osd.5.
   [root@rook-ceph-tools-/]# ceph osd status
   ID  HOST           USED  AVAIL  WR OPS  WR DATA  RD OPS  RD DATA  STATE
    0  10.xx.xx.210  4856M  75.2G      0        0       0        0   exists,up
    1  10.xx.xx.215  2986M  77.0G      0        0       1       89   exists,up
    2  10.xx.xx.98   3243M  76.8G      0        0       1       15   exists,up
    3  10.xx.xx.195  4945M  75.1G      0        0       0        0   exists,up
    4  10.xx.xx.170  5053M  75.0G      0        0       0        0   exists,up
    5  10.xx.xx.197     0      0       0        0       0        0   exists
   ```

6. Remove the failed OSD.

   ```
   # ceph osd purge number --yes-i-really-mean-it
   ```

7. Connect to the node that hosted the failed OSD and do one of the following:

   - Replace the hard disk in case of a hardware failure.

- Reformat the disk completely.

```
$ sgdisk -zap /dev/disk
$ dd if=/dev/zero of=/dev/disk bs=1M count=100
```

- Reformat the partition completely.

```
$ wipefs -a -f /dev/partition
$ dd if=/dev/zero of=/dev/partition bs=1M count=100
```

8. Restart `rook-ceph-operator`.

```
# kubectl scale deploy -n rook-ceph rook-ceph-operator --replicas=1
```

9. Monitor the OSD pods.

```
# kubectl get po -n rook-ceph
```

If the OSD does not recover, use the same procedure to remove the OSD, and then remove the disk or delete the partition before restarting `rook-ceph-operator`.

For more information on Rook and Ceph, see https://github.com/rook/rook/blob/master/Documentation/ceph-common-issues.md.

Common Utility Commands

You can use the following utility commands installed in **/usr/local/bin** to connect to pods running in the system.

Command	Description
`paragon-db`	Start the Postgres SQL shell in the atom-db instance for database management.
`pf-cmgd`	Start the shell connected to the Paragon Pathfinder CMGD instance.
`pf-crpd`	Start the shell connected to the CRPD instance.
`pf-redis`	Start (authenticated) redis-cli connected to Paragon Pathfinder Redis.
`pf-debugutils`	Start the shell connected to Paragon Pathfinder debugutils pod which must be installed with `install_northstar_debugutils: true`.

RELATED DOCUMENTATION

Installation Prerequisites | 11

Install Paragon Automation | 20

Uninstall Paragon Automation | 37

CHAPTER 6

Migrate Data

Migrate Data from Northstar to Paragon Automation

IN THIS SECTION

You can migrate DeviceProfile, Cassandra DB, and Analytics (ES DB) data from an existing Northstar Release 6.*x* setup to a Paragon Automation Release 21.2 setup.

Use the steps described in this topic to migrate date from Northstar to Paragon Automation.

Prerequisites

- Ensure that both the Northstar and Paragon Automation setups are up and running.
- Cassandra must be accessible from Paragon Automation. Set the `rpc_address` parameter in the **/opt/northstar/data/apache-cassandra/conf/cassandra.yaml** to an address to which the Paragon Automation setup can connect. Ater setting the address, restart Cassandra for the configuration changes to take effect:

  ```
  root@ns1: # supervisorctl restart infra:cassandra
  ```

- (Optional) We recommend that you back up Cassandra DB prior to starting the data migration procedure. The `userDataBackupRestore` script in Northstar facilitates backing up and restoring relevant data in Cassandra. The script exports and imports the data using the CSV format.

For example, to export data to **/opt/northstar/utils/backupRestoreData/** in the CSV format:

```
root@ns1: # /opt/northstar/utils/userDataBackupRestore.js -t db -m backup allset
```

Large database tables might cause backup issues. Specifically, tables with sizes exceeding ~1M rows might cause the backup to fail. The `userDataBackupRestore.js` script indicates any issues with error messages, similar to the following:

```
“Cannot catch: keyspace.table_name Reason: File too big/No such file.”
```

Or

```
“The entries of keyspace.table_name is larger than 1000,000: row count”
```

If you see these errors, back up those specific tables manually with the `-force` flag:

```
root@ns1: # /opt/northstar/utils/userDataBackupRestore.js -t db -m backup keyspace.table_name
-force
```

Follow this procedure to migrate data from Northstar to Paragon Automation.

Create the nsmigration Task Pod

1. Log in to the Paragon Automation primary node.
2. Create the nsmigration task pod.

```
root@pa-primary: # kubectl apply  -f /etc/kubernetes/po/nsmigration/kube-cfg.yml
 job.batch/nsmigration created
```

3. Log in to the nsmigration task pod.

```
root@pa-primary:~# kubectl exec -it $(kubectl get po -n northstar -l app=nsmigration -o
jsonpath={..metadata.name}) -c nsmigration -n northstar -- bash
root@nsmigration-fcvl6:/# cd /opt/northstar/util/db/nsmigration
```

Migrate DeviceProfile and Cassandra DB

1. Run the `ns_data_migration.py -a -sp -dp` script from the nsmigration task pod. The complete command syntax is `./ns_data_migration.py -a` *`ns-app-server-ip`* `-su root -sp` *`ns-app-user-ssh-password`* `-dh` *`cassandra-db-host`* `-du cassandra -dp` *`cassandra-password`* `-pu` *`postgres-user`* `-pp` *`postgres-password`* `-ph` *`postgres-host`* `-po` *`postgres-port`* `-pah` *`vip-of-ingress-controller-or-hostname-of-main-web-application`* `-pau` *`paragon-web-ui-login`* `-pap` *`paragon-web-ui-password`* `-dr 1`.

   For example:

```
root@nsmigration-7xbbz:/opt/northstar/util/db/nsmigration# ./ns_data_migration.py -a
10.xx.xx.200 -su root -sp password -dh 10.xx.xx.200 -dp password -pu northstar -pp
BB91qaDCfjpGWPbjEZBV -ph atom-db.common -po 5432 -pah 10.xx.xx.11 -pau admin -pap password1 -
dr 1
Logs stored at /opt/northstar/util/db/nsmigration/logs/nsdatamigration.log
Cassandra connection established...connection attempt: 1
Testing cassandra connectivity
Connected to cluster Test Cluster
Testing EMS connectivity
scope_id: d3ae39f7-35c6-49dd-a1bd-c509a38bd4ea, auth_token length: 1160
scoped token length: 1303
jwt_token length: 40974
All connection ok starting mirgation
Starting device profile migration...
Found 2 devices in Northstar device profile

...

2021-09-28 13:59:15,559:INFO:Copying cassandra table anycastgroup: anycastgroupIndex
2021-09-28 13:59:15,942:INFO:Copying cassandra table pcs_restconf: admin_group_names
2021-09-28 13:59:15,946:INFO:Skipping database db_meta
The NS data migaration completed
```

   The following parameters must be specified while running the `ns_data_migration.py` script.

   - `-a APP, --app APP`—IP address or hostname of the application server
   - `-su SSHUSER, --sshuser SSHUSER`—SSH username (default is root)
   - `-sp SSHPASS, --sshpass SSHPASS`—SSH password
   - `-so SSHPORT, --sshport SSHPORT`—SSH port (default is 22)
   - `-du DBUSER, --dbuser DBUSER`—Cassandra DB username (default is cassandra)

- `-dp DBPASS, --dbpass DBPASS`—Cassandra DB password
- `-do DBPORT, --dbport DBPORT`—Cassandra DB port (default is 9042)
- `-dh DBHOST, --dbhost DBHOST`—Comma separated Host IP addresses of Cassandra DB
- `-pu PGUSER, --pguser PGUSER`—Postgres DB username (default is northstar)
- `-pp PGPASS, --pgpass PGPASS`—Postgres DB password
- `-ph PGHOST, --pghost PGHOST`—Postgres DB host (default is atom-db.common)
- `-po PGPORT, --pgport PGPORT`—Postgres DB port (default is 5432)
- `-pah PARAGONHOST, --paragonHost PARAGONHOST`—IP address (VIP) of Paragon Automation Web UI
- `-pau PARAGONUSER, --paragonUser PARAGONUSER`—Paragon Automation Web UI username
- `-pap PARAGONPASSWORD, --paragonPassword PARAGONPASSWORD`—Paragon Automation Web UI user password
- `-dr DISCOVERYRETRIES, --discoveryRetries DISCOVERYRETRIES`—Device discovery retries (default is 2).

  The `dr DISCOVERYRETRIES` option is used for device-profile migration when device discovery by Paragon Automation fails in the first attempt. There are multiple reasons for discovery failure, such as devices not being reachable or device credentials being incorrect. Despite discovery failure for devices with incorrect information, devices with correct information are discovered. Partial failure for a subset of devices while discovering multiple devices at a time is possible. To determine the exact reason of failure, see the **Monitoring > Jobs** page in the Paragon Automation Web UI.

  If the `dr` option is set to more than 1, on getting a discovery failure, the `ns_data_migration.py` script retries the discovery for all the devices. This does not impact the devices that are already discovered. However, the chances of successfully discovering devices in subsequent attempts for any failed device discovery is minimal. We recommended that the maximum value for the `dr` option be set to 2, which is the default value. Use a value of 1 if there are too many devices in the network avoid unnecessary retries.

2. Verify the DeviceProfile data.

   Log in to Paragon Automation Web UI and navigate to **Configuration > Device**. Verify that all the devices are discovered and present. Also, verify if all the configuration information is the same as that in the Northstar device profile.

   To view the device discovery result, see **Monitoring > Jobs** page in the Paragon Automation Web UI.

3. Verify Cassandra DB data.

   The log output of the `ns_data_migration.py` script indicates if there were any problems migrating data from Cassandra. You can also check individual tables for content or row count. For example:p

On Northstar:

```
root@ns1 # /opt/northstar/thirdparty/apache-cassandra/bin/cqlsh -u cassandra -p <password> -
ssl
cassandra@cqlsh> select count(*) from pcs.messages;
 count
-------
  2173
```

On Paragon Automation:

```
root@pa-primary:~# /usr/local/bin/paragon-db
postgres=# \c ns_pcs
ns_pcs=# select count(*) from messages;
 count
-------
  2268
```

(Optional) Migrate Analytics Data

If you have installed Analytics, perform the following steps to migrate analytics data from Northstar ES DB to Paragon Automation Influx DB:

1. Log in to the nsmigration task pod, and run the `import_es_data.py -a` script.

```
root@nsmigration-p7tcd:/# cd /opt/northstar/util/db/nsmigration
root@nsmigration-p7tcd:/opt/northstar/util/db/nsmigration# ./import_es_data.py -a 10.xx.xx.95
Logs stored at /opt/northstar/util/db/nsmigration/logs/es_data_migration.log
Certs are missing, fetching them from Northstar app server
Please enter SSH password:
Testing Elasticsearch connectivity
Elasticsearch DB connection ok
Testing Influx DB connectivity
Influx DB connection ok
Starting data extraction for type= interface

<OUTPUT SNIPPED>

  "migartion_rate_sec": 1471.1758360302051,
```

```
  "timetaken_min": 0.7725,
  "total_points": 68189
}
ETLWorker-2 completed, total points=68189 in 0.7725 minutes with
migartion_rate=1471.1758360302051
```

The `import_es_data.py` script options are described here.

- Statistics type—By default, supports interface, LSP, and link-latency stats data. You can select a specific type by using the `--type` option.
- Rollups type—By default, supports daily and hourly time periods. You can select a specific type by using the `--rollup_type` option.
- Migration Schema: The ES DB to Influx DB schema mapping is defined in the **/opt/northstar/util/db/nsmigration/es_influx_mapping.json** file.
- Rollup ages—By default, fetches hourly and daily data for the last 180 days and 1000 days respectively. You can change the ages by using `--hourly_age` and `--daily_age` options.
- ETL parallel worker process: By default, uses 4 ETL parallel worker processes. You can change the worker process by using the `--wp` option.
- Execution time—The script execution time varies based on data volume, the number of days, and the number of ETL parallel worker processes. For example, if 4 ETL workers take migration_rate=1500, then:

  25K LSP stats of 180 days hourly can take 5 hours and

  50K LSP stats of 180 days hourly can take 10 hours

For more information on script arguments, see `help 'import_es_data.py -h'`.

2. Verify Influx DB data using the following commands.

   - To query all tunnel traffic data for the last 30 days in influxdb, run the `/opt/pcs/bin/getTrafficFiles.py` script inside the dcscheduler pod:

```
root@pa-primary:~# kubectl exec -it $(kubectl get po -n northstar -l app=dcscheduler -o
jsonpath={..metadata.name}) -c dcscheduler  -n northstar -- /opt/pcs/bin/
getTrafficFiles.py -t tunnel_traffic -i 1d -b 30
#Starting Time : 08/17/21 12:00:00 AM
#Interval : 24 hour
# UNIT = 1

# Aggregation:
#   - Series: time series
```

```
#   - Statistic: 95th percentile
#   - Interval: 1 day
# Report Date= 2021-09-16 (Thu) 08:56

vmx101:Silver-101-102 A2Z 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 -1 -1 0 -1
-1
vmx101:Silver-101-104 A2Z 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 -1
vmx102:Silver-102-101 A2Z 0 1071913 1072869 1073082 1073378 1073436 1073378 1073620
1073378 1073388 1073484 1073896 1074086 1073974 1073795 1073378 1073590 1073790 1074498
1074595 1074498 1074092 1076565 1076565 1076919 1075502 1075857 1075325 1075148 -1 -1
vmx102:Silver-102-103 A2Z 0 2118101 2120705 2121258 2120438 2120773 2119652 2121258
2120296 2120190 2120962 2121364 2121867 2121817 2122209 2120167 2120323 2121665 2122733
2122685 2122321 2121511 2121855 2119546 2119700 2109572 2102489 2101604 2121258 2109749
2110280
vmx102:Silver-102-104 A2Z 0 3442749 3449550 3450757 3448983 3448603 3446081 3453525
3451513 3448142 3449008 3450874 3452721 3451650 3450733 3447297 3447147 3449132 3451747
3450887 3450727 3448429 3452310 3448132 3447328 3200657 3200480 3197646 3445363 3215530
3215884
vmx103:Silver-103-101 A2Z 0 2149705 2151625 2158319 2170251 2170980 2171171 2169252
2167757 2168518 2172730 2168582 2166350 2161904 2161460 2167162 2158050 2160413 2166131
2167033 2166226 2165632 2171717 2178973 2178102 2158015 2158015 2157661 2157306 -1 -1
vmx103:Silver-103-102 A2Z 0 2122922 2125508 2131074 2141411 2142899 2141840 2139937
2138338 2139743 2144156 2139602 2138745 2134561 2132725 2137973 2129397 2132755 2138203
2138653 2136713 2135444 2144637 2150006 2147677 2108332 2107801 2107270 2124800 2112228
2113113
vmx103:Silver-103-104 A2Z 0 3426540 3437589 3447876 3461550 3464308 3461249 3460710
3453848 3458821 3463446 3456119 3456969 3450036 3446943 3451602 3439059 3445325 3455444
3455491 3454308 3449833 3468558 3472376 3470223 3185429 3187731 3183304 3430135 3198001
3202781
vmx104:Silver-104-102 A2Z 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
vmx104:Silver-104-103 A2Z 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
vmx105:rsvp-105-106 A2Z 0 114 114 121 116 122 125 125 114 214 224 215 223 213 223 222 226
222 217 213 214 216 219 218 219 202 202 202 211 204 202
```

- To query all egress interface traffic data for the last 30 days in influxdb, run the `/opt/pcs/bin/getTrafficFiles.py` script inside the dcscheduler pod:

```
root@pa-primary:~# kubectl exec -it $(kubectl get po -n northstar -l app=dcscheduler -o
jsonpath={..metadata.name}) -c dcscheduler  -n northstar -- /opt/pcs/bin/
getTrafficFiles.py -t interface_out -i 1d -b 30
#Starting Time : 08/17/21 12:00:00 AM
#Interval : 24 hour
```

```
# UNIT = 1

# Aggregation:
#   - Series: time series
#   - Statistic: 95th percentile
#   - Interval: 1 day
# Report Date= 2021-09-16 (Thu) 08:49


vmx101 ge-0/0/8.0 A2Z 0 2620 2620 2621 2621 2621 2621 2622 2622 2623 2624 2626 2627 2627
2627 2627 2627 2627 2627 2628 2631 2631 2632 2632 0 0 0 2632 -1 -1
vmx101 ge-0/0/5 A2Z 0 843 846 848 860 843 858 863 866 1001 1012 1012 1018 1011 1048 1018
1048 1027 1013 1025 1017 1010 1046 1046 1048 1053 1055 1073 1045 -1 -1
...
...
...
vmx107 ge-0/0/8.0 A2Z 0 2620 2621 2622 2622 2623 2624 2626 2626 2630 2631 2632 2632 2632
2632 2632 2632 2633 2633 2635 2635 2635 2635 2636 2636 0 0 0 2636 0 0
vmx107 ge-0/1/9.0 A2Z 0 6888955 6907022 6907653 6902645 6899706 6892876 6905804 6902894
6899395 6897851 6897322 6896863 6900351 6898745 6890080 6889337 6896781 6902034 6899116
6898749 6898630 6903136 6889662 6890800 6401393 6410976 6400867 6885900 6431500 6436156
vmx107 ge-0/0/5 A2Z 0 4290428 4296767 4297691 4295393 4292480 4290593 4293842 4293149
4295279 4294504 4294045 4294905 4294996 4294921 4292093 4292703 4295408 4297494 4296424
4295983 4295972 4296808 4294929 4299425 4285605 4286205 4285146 4288390 2126258 2127510
vmx107 ge-0/0/6 A2Z 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 122 122 122 122 122 122
122 122 122
vmx107 ge-0/0/7 A2Z 0 878 874 915 898 886 879 897 889 1028 1021 1022 1023 1055 1079 1077
1097 1094 1092 1044 1007 1028 1062 1065 1057 1094 1075 1071 1102 1082 1054
vmx107 ge-0/0/8 A2Z 0 2921 2925 2925 2924 2925 2926 2928 2928 2930 2934 2934 2936 2934
2935 2935 2934 2935 2936 2939 2938 2938 19892 20581 20965 20582 21076 20376 21578 23252
21312
vmx107 ge-0/1/8 A2Z 0 2127443 2130145 2130846 2128792 2128138 2127177 2128628 2128331
2128820 2128716 2128916 2129022 2129380 2128995 2127648 2127240 2128885 2130132 2130474
2130345 2129410 2129376 2125952 2125957 2117061 2114139 2119148 2126518 2122808 2121792
vmx107 ge-0/1/9 A2Z 0 6889737 6907821 6908350 6903585 6900486 6893779 6906516 6903747
6900412 6898908 6898427 6897892 6901325 6899809 6891078 6890377 6897822 6903099 6900152
6899763 6899726 6904248 6890745 6891782 6402507 6412083 6401924 6884168 6432556 6437280
```

(Optional) Migrate Northstar Planner Data

If you want to use saved Northstar Planner models on the Northstar application server file system in Paragon Automation, copy the models using the following steps:

1. Log in to the Northstar server.
2. Use scp and copy the directory (**/opt/northstar/data/specs**) where your Planner models are saved to the Paragon Automation primary node (**/root/ns_specs**). For example:

```
[root@ns1-site1-q-pod21 specs]# ls -l /opt/northstar/data/specs
total 8
drwx------ 2 root root 4096 Sep 16 08:18 network1
drwx------ 2 root root 4096 Sep 16 08:18 sample_fish


[root@ns1-site1-q-pod21 data]# [root@ns1-site1-q-pod21 ~]# scp -r /opt/northstar/data/specs
root@10.xx.xx.153:/root/ns_specs
The authenticity of host '10.xx.xx.153 (10.xx.xx.153)' can't be established.
ECDSA key fingerprint is SHA256:haylHqFfEuIEm8xThKbHJhG2uuTpT2xBpC2GZdzfZss.
ECDSA key fingerprint is MD5:15:71:76:c7:d2:2b:0d:fe:ff:0d:5f:62:7f:52:80:fe.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.xx.xx.153' (ECDSA) to the list of known hosts.
bblink.x
               100% 3893     2.2MB/s   00:00
bgplink.x
               100%  140     9.6KB/s   00:00
bgpnode.x
               100%  120    56.5KB/s   00:00
bgpobj.x
               100% 4888     1.8MB/s   00:00
cosalias.x
               100%  385   180.4KB/s   00:00
custrate.x
               100% 1062   184.0KB/s   00:00
demand.x
               100%  104KB   2.1MB/s   00:00
dparam.x
               100%   11KB   2.5MB/s   00:00
...
```

3. Log in to the Paragon Automation primary node.

4. Copy the **/root/ns_specs** folder to the Northstar Planner pod at **/opt/northstar/data/specs** using the `kubectl` command. For example

```
root@pa-primary:~# ls -l /root/ns_specs
total 8
drwx------ 4 root root 4096 Sep 16 01:41 network1
drwx------ 4 root root 4096 Sep 16 01:41 sample_fish


root@pa-primary:~# kubectl cp /root/ns_specs  northstar/$(kubectl get po -n northstar -l
app=ns-web-planner -o jsonpath={..metadata.name}):/opt/northstar/data/specs -c ns-web-planner
```

5. Verify that the Northstar Planner models are copied inside the Northstar Planner pod at **/opt/northstar/data/specs/ns_specs**.

```
root@pa-primary:~/ns_specs# kubectl exec -it $(kubectl get po -n northstar -l app=ns-web-
planner -o jsonpath={..metadata.name}) -c ns-web-planner -n northstar -- ls -l /opt/northstar/
data/specs/ns_specs
total 8
drwx------ 2 root root 4096 Sep 16 08:18 network1
drwx------ 2 root root 4096 Sep 16 08:18 sample_fish
```

RELATED DOCUMENTATION

Paragon Automation System Requirements | 6

Install Paragon Automation | 20

Uninstall Paragon Automation | 37