Table of Contents

Chapter 1  Overview ........................................................................................................... 1

What is server clustering? ................................................................................................... 1
Extending the cluster for remote sites .................................................................................. 2
Supporting technologies ...................................................................................................... 3
  Automating failover ......................................................................................................... 4
  Providing ease of use ...................................................................................................... 5
Hardware and platform requirements .................................................................................. 6
User workflow ..................................................................................................................... 6

Chapter 2  Installation and Set Up .................................................................................. 9

Overview ........................................................................................................................... 9
Preparing for installation .................................................................................................... 11
Installing P4CMGR package ............................................................................................. 16
  CentOS and Red Hat ......................................................................................................... 17
  Ubuntu .............................................................................................................................. 17
Setting up the server cluster ............................................................................................. 18
  Add depot standby .......................................................................................................... 22
  Add workspace servers ..................................................................................................... 22
  Add the cluster router ........................................................................................................ 23
  Create a checkpoint .......................................................................................................... 23
Specifying workspaces for new users ................................................................................ 24
Setting the location and format of log files ......................................................................... 24
Creating a client ................................................................................................................ 25
Adding a license file ............................................................................................................ 25
Using a server cluster in a distributed environment ......................................................... 25
  Forwarding requests from edge to depot master ............................................................. 27
Securing the cluster and authenticating users .................................................................... 27
Migrating from an existing installation ............................................................................... 28
Troubleshooting installation issues ................................................................................... 28

Chapter 3  Configuring and Reconfiguring a Server Cluster ..................................... 31

Cluster-related configuration variables .............................................................................. 31
  Variable scope ................................................................................................................. 33
  Changing configuration values ......................................................................................... 34
  Configurables reference .................................................................................................. 34
Zookeeper configurables ..................................................................................................... 38
Changing server cluster topology ....................................................................................... 39

Chapter 4  Backup and Recovery .................................................................................... 41
<table>
<thead>
<tr>
<th>Chapter 5</th>
<th>Monitoring and managing server cluster</th>
<th>51</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitoring a server cluster</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>Getting status information</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>Evaluating replication status</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>Monitoring Zookeeper</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>Monitoring events</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>Managing a server cluster</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td>Improving performance</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td>Managing failovers</td>
<td>56</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 6</th>
<th>Command Reference</th>
<th>57</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using p4cmgr commands</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>Getting authorization to run p4cmgr commands</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>Restrictions on using p4cmgr and p4 commands</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>Getting help for p4cmgr commands</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>General guidelines</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>Specifying server id’s</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>Specifying host names</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>Using environment variables</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>Triggers</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>p4cmgr</td>
<td>61</td>
<td></td>
</tr>
<tr>
<td>p4cmgr add</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>p4cmgr backup</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>p4cmgr init</td>
<td>67</td>
<td></td>
</tr>
<tr>
<td>p4cmgr remove</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>p4cmgr restart</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>p4cmgr ssh-reset</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>p4cmgr start</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>p4cmgr status</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td>p4cmgr stop</td>
<td>79</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 1  
Overview

This chapter introduces Perforce’s server cluster architecture: it describes its advantages, it discusses the technologies used to support it, it explains how you can extend it, and it outlines the workflow used to install and use it.

What is server clustering?

Server clustering offers a multi-node solution for high availability and horizontal data center scaling. The following figure illustrates the topology of a Perforce server cluster.

Figure 1.1. Server Cluster

The elements of a Perforce server cluster include the following:

- A depot master that records changes to revision data.
- A depot standby that assumes the role of the depot master if the depot master fails.

The depot standby has a copy of the depot master’s database and journal files. It does not service workspace server requests.

The depot standby is automatically synchronized to the depot master. Should the depot master fail, the depot standby is promoted to master status. Any updates that the depot standby has not recorded from the depot master are rolled back. For example, a submit...
operation that has completed on the depot master but that the depot standby doesn’t know about will return with a failure, and the user will need to re-submit.

- A fault-tolerant *shared file system* that is accessible to all cluster nodes; it stores archive files and workspace shelves.

- An optional *workspace server* that manages client data. You can add more workspace servers to distribute the load of clients across machines.

The use of workspace servers is optional. If you are not concerned with offloading some of the work of the depot master or if you are not concerned with horizontal scaling, you can have clients work directly with the depot master.

- A pre-configured *cluster router* (Perforce Broker) that dispatches user requests to the workspace servers or to the depot master. It can also dispatch requests from external edge servers to the depot master.

The server cluster is accessed as a single server resource with the `P4PORT` of the cluster router as the entry point. The router directs incoming requests to the appropriate server; depending on how the cluster is defined, this might be a workspace server or a depot master.

The client connects to the cluster using the cluster router, sending requests, and syncing with the depot as if working with a standard Perforce server.

Server cluster elements are supported by additional technologies: Apache Zookeeper for automated failover, and SaltStack for administrative ease of use. These are described in “Supporting technologies” on page 3. You will note in Figure 1.1, “Server Cluster” on page 1, that each member of the cluster also runs a `p4zk` process that communicates with the Zookeeper service. Its function is to inform Zookeeper of the availability that cluster member.

If you are familiar with the commit-edge model, it might help to think of a server cluster as using a highly available, horizontally scalable commit-edge architecture. The depot master functions as a commit server; the depot standby, as a forwarding replica, and the workspace server, as an edge server. What distinguishes a cluster from the commit-edge architecture is the use of a shared file system, the low latency implied in the data center environment, the pre-configuration of the cluster elements that allows them to work together, and the supporting technologies that provide automated failover and ease of use.

**Extending the cluster for remote sites**

A server cluster is meant to run in a data center environment; it depends upon a shared file system and the low latency that characterizes such an environment. You can, if you need to, extend your cluster geographically, by adding a proxy or an edge server as shown in the next figure:
Figure 1.2. Extending the Cluster in a Distributed Environment

For more information on how to configure this deployment, see “Using a server cluster in a distributed environment” on page 25.

Supporting technologies

Two features of a cluster, failover and ease of use, are furnished by supporting technologies: Apache Zookeeper and SaltStack.

- Zookeeper tracks the state of cluster nodes and supports the automatic failover of the depot master to the depot standby.

- SaltStack provides the distributed configuration management and orchestration platform on which P4CMGR operates. P4CMGR, in turn, is the tool that allows administrators to easily initialize, monitor, and manage the cluster environment.

The following sections explain how these technologies support failover and ease of use.
Automating failover

A server cluster uses an Apache Zookeeper service to automate the process of failure detection and of role migration from the failing depot master to the depot standby. The fault-tolerant Zookeeper service maintains cluster state information and monitors the health of the cluster nodes. Zookeeper uses the \texttt{p4zk} process, which supports each server in the cluster, to register the existence of the server and to respond to cluster state changes. Three instances of the Zookeeper service must be deployed on three different nodes in the cluster environment to support the fault tolerance of Zookeeper itself.

The next figure illustrates the Zookeeper components that are part of your cluster installation. These include the \texttt{p4zk} processes, deployed on all depot nodes, and the three Zookeeper servers: one is found on a dedicated host (upper left corner); the two others are shown by the \texttt{Z} icon on the workspace server node and on the cluster router node.

Figure 1.3. Cluster with Supporting Technology

At startup, each cluster member, launches a new \texttt{p4zk} process that serves as a conduit from that member (\texttt{p4d} or \texttt{p4broker}) to Zookeeper. In the event of the failure of a node in the cluster, \texttt{p4zk} is responsible for the following:

- If a depot master fails, the \texttt{p4zk} process instructs the depot standby to restart as a depot master, and it updates the router’s configuration file and the configuration file of each workspace server with the new target — that of the promoted depot standby.

It also ensures that no data is lost in the transition to the new depot master.
• If a workspace server fails, **p4zk** updates the router's configuration file so that it no longer forwards client requests to that server.

• If a workspace node is stopped or started, it reports the change to the depot master and it updates the router's configuration file. (In the case of workspace addition or removal, it is P4CMGR that handles the change.)

For more detailed information about the failover process, see Chapter 4, “Backup and Recovery” on page 41.

**Providing ease of use**

Setting up and administering a cluster includes many complex tasks that are vulnerable to error: P4CMGR automates basic administration tasks that relate to cluster setup and management, ensuring that server configuration is consistent and that it supports the cluster's high availability and horizontal scaling features.

P4CMGR is a collection of Python scripts that extends Salt to manage administrative activities in a cluster.

Salt support is accessed using the **p4cmgr** interface, which defines the commands listed in the following table. For detailed information, see the description of each command in Chapter 6, “Command Reference” on page 57.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>p4cmgr add</strong></td>
<td>Adds a node to a cluster.</td>
</tr>
<tr>
<td><strong>p4cmgr backup</strong></td>
<td>Checkpoints the depot master using the replicated data on the depot standby.</td>
</tr>
<tr>
<td><strong>p4cmgr init</strong></td>
<td>Initializes a cluster.</td>
</tr>
<tr>
<td><strong>p4cmgr remove</strong></td>
<td>Removes a service from a cluster.</td>
</tr>
<tr>
<td><strong>p4cmgr restart</strong></td>
<td>Stops and then starts one or more servers in a cluster.</td>
</tr>
<tr>
<td><strong>p4cmgr start</strong></td>
<td>Starts one or more nodes in a cluster.</td>
</tr>
<tr>
<td><strong>p4cmgr status</strong></td>
<td>Returns a debug dump for all hosts and services in a cluster.</td>
</tr>
<tr>
<td><strong>p4cmgr stop</strong></td>
<td>Stops the specified node(s) in a cluster.</td>
</tr>
</tbody>
</table>

The administrator uses these commands to initialize and manage the server cluster.

Note that some **p4cmgr** functionality takes precedence over equivalent **p4** commands. For this reason, the router actively filters certain **p4** commands to avoid conflicts between the effect of these commands and the proper functioning of a cluster. Filtered commands include some variants of the **p4 admin** and **p4 configure** commands, the **p4 cluster** command, and the **p4 cachepurge** command. For detailed information about these commands see, “Restrictions on using p4cmgr and p4 commands” on page 57.
Hardware and platform requirements

The cluster environment must include the following:

- A tightly coupled data center environment with a low latency and stable network.
- A sufficient number of 64-bit servers or virtual machines to support high availability and if needed, horizontal scaling.
- Operating systems: Red Hat Linux 6.5, Ubuntu 12.04, or CentOS 6.4.

The operating systems can run on a physical or a virtual machine. High availability requires that the depot master and the depot standby be hosted on separate machines.

- A fault tolerant, POSIX-compliant shared file system; no specific requirements for locking semantics.
- Firewall protection. The cluster needs to be situated behind a firewall; only the administrator may access servers other than the router directly. Inside the firewall, the administrator can issue `p4cmgr` commands as well as standard `p4` commands.

For additional details, see Chapter 2, “Installation and Set Up” on page 9.

User workflow

The general workflow for installing, initializing, and working with a cluster includes the following steps, which must be performed by a system administrator:

1. Prepare the cluster environment.

2. Install the `perforce-cluster-manager` package.

   You can now execute `p4cmgr` commands.

3. Use the `p4cmgr init` command to initialize the cluster.

   This sets up a depot master on the target machine and sets cluster-related configurables on the depot master.

4. Use the `p4cmgr add zookeeper` command to add the nodes where Zookeeper is going to run.

   Before you add the depot standby, workspace, or router nodes, you should add the nodes that will host Zookeeper services. You’ll need a minimum of three or a maximum of five nodes. Either way, the number must be odd. You should provision the Zookeeper nodes first because every time you add a Zookeeper node, you must stop and restart the cluster. It is easiest to just do this once. You may not have more than one Zookeeper service on one host. If you have an existing Zookeeper service on a host, you need to move it somewhere else.

5. Use the `p4cmgr restart` command to restart the cluster.
6. Use the `p4cmgr add depot` command to add a depot standby.

   Only one depot standby is supported for the 2014.2 release; however, you are not
   prevented from adding more than one depot standby.

7. Use the `p4cmgr add workspace` command for each workspace server you want to create.
   (Optional)

8. Use the `p4cmgr add router` command to add a cluster router.

9. Create users, unless you are planning to have the users be auto-created.

   Service users are automatically created for each server to allow trust between cluster members.

   Clients can access the server cluster using any Perforce client, including the `p4` command line
   interface, P4V, P4VS, or Eclipse.

   The next chapter, Chapter 2, “Installation and Set Up” on page 9, provides detailed
   information on how you install a server cluster.
Chapter 2  
Installation and Set Up

Server cluster set up and administration are intended to be fairly straightforward. However, ease of use depends on the integrity of your initial setup, which requires close cooperation between IT staff and the Perforce administrator. Please follow the guidelines and procedures in this chapter closely to ensure the best outcome.

Overview

A full server cluster consists of the following nodes:

- One P4CMGR node that also hosts a Zookeeper service.
- Two additional nodes that host a Zookeeper service each.
- One depot master.
- One depot standby.
- One or more workspace servers (optional).
- One cluster router.
- A shared archive.

Zookeeper services may be hosted on nodes that host other server cluster elements, or they may be hosted on separate nodes. The following figure illustrates a sample deployment:
Figure 2.1. Deployed Cluster

As shown, this cluster contains one P4CMGR node, a depot master, a depot standby (master and standby are both servers of type depot), two workspace servers, a shared archive, and a router. A firewall is recommended in production.

Optimal performance is obtained by deploying cluster elements on different networks. The next figure illustrates the ideal deployment with respect to network topology. The four-network configuration is not required; but it is recommended for best performance.
The Perforce administrator is responsible for setting up the cluster and changing the default configuration if needed. However, before this can be done, the IT department must set up the data center environment to support a server cluster; this work is described in the next section.

Overall, setting up the cluster involves the following steps: the first two steps can be performed by the IT department; the last six steps can be done either by the IT or by a Perforce administrator. The Perforce administrator must have root privileges on all cluster nodes.

1. Prepare for installation.
2. Install the P4CMGR package.
3. Initialize the cluster.

   This creates the depot master, and starts a Zookeeper service on the P4CMGR host.
4. Add Zookeeper on two or four additional nodes.
5. Restart the cluster.
6. Add the depot standby.
7. Add the workspace servers (optional).
8. Add the cluster router.

The following sections describe each of these steps in detail.

**Preparing for installation**

The server cluster environment must meet the following requirements:

- It must be a tightly-coupled data center environment with a network that is stable with low latency.
• It must include a sufficient number of 64-bit servers or virtual machines to support high availability and if needed, horizontal scaling.

If you use virtual machines, observe the following guidelines:

• The depot master node and its depot standby node must not share the same physical resources other than the shared file system. This is to assure high availability in the case of underlying hardware failure.

• A VM farm should be configured so that VM processes are not restarted on different hosts if the underlying hardware fails.

Some VM installations offer high availability features. In this case, the cluster high availability process might conflict with the VM high availability process. To ensure the smooth operation of the cluster, VM migration for high availability should be disabled.

• Supported Linux operating systems must be installed: Ubuntu 12.04, CentOS 6.4, or Red Hat 6.5.

All machines in the cluster must run the same operating system; but all operating systems do not have to be the same version. The operating systems can run on physical hardware or in a virtual machine. High availability requires that the depot master and the depot standby be hosted on separate machines. The operating system for the fault tolerant, shared filesystem can be different from the operating system of other nodes in the cluster.

• Each depot and workspace server needs /p4/archives as the fault tolerant shared file system: archive data and shelves will be stored there. In addition, we recommend /p4/data as one local disk (hosting .db files) and /p4/journals on a separate local disk.

We recommend that these files be placed on different disks to improve performance and to support recovery. The P4CMGR host and router hosts do not need any special disk setup.

• The cluster must be protected by a firewall. Only the administrator may access servers in the cluster other than the router directly. Normal users can not use p4cmgr commands, and they must observe some restrictions on the use of p4 commands. For a discussion of these restrictions, see "Restrictions on using p4cmgr and p4 commands" on page 57.

To set up a server cluster environment, the IT department, in consultation with a Perforce administrator, must do the following:

1. Plan the topology of the deployment, identifying the physical and virtual machines that will host cluster components (depot master, depot standby, workspace servers, etc.)

2. Install supported operating systems on the selected physical or virtual machines.

3. Make sure there are no Zookeeper services already running on any of the nodes intended for the server cluster. If there are existing Zookeeper services supporting other technology, these must be moved elsewhere.

4. Make sure there are no SaltStack services already running on any of the nodes intended for the cluster. Move existing SaltStack services elsewhere.

5. Set up authentication for the administrator:
Chapter 2. Installation and Set Up

- Create the operating system user `perforce` and the operating system group `perforce` with the same user id and group id on every system in the cluster. Sharing user and group id will make ownership of files on the shared file system much easier.

Select a group id that is not currently in use on any of your systems. You can see the group id’s that are currently used in `/etc/group`.

Select a user id that is not currently in use on any of your systems. You can see the user id’s that are currently used in `/etc/passwd`.

We recommend that you set the Perforce operating system user’s default shell to be `/bin/bash`.

```
groupadd -g groupid perforce
useradd -s /bin/bash -u userid -g groupid -m perforce
```

- Make sure that every user who will run `p4cmgr` commands is a member of the operating system group `perforce` on the P4CMGR host. An operating system user other than `perforce` may run `p4cmgr` commands, but they must be members of the operating system group `perforce`. Any operating system user who wants to view logs on clustered nodes must also be a member of the operating system group `perforce`.

Whether you plan to run as root or non-root, you should do the following on all your cluster nodes:

- Use the `usermod` command to make sure that your operating system user is part of the `perforce` group. For example:

```
usermod -a -G perforce root
usermod -a -G perforce OS_user_with_ssh_and_sudo_privileges
```

- Log out and log in again after adding your user to the `perforce` group, or the current shell might not have the group associated with your user.

- The operating system user who will set up the server cluster or add systems to it by running the `p4cmgr_init` or `p4cmgr_add` commands, must have `sudo (NOPASSWD)` privileges on all systems in the cluster. This `sudo (NOPASSWD)` access can be removed after all setup commands have been run. For example:

```
echo 'perforce ALL=(ALL) NOPASSWD:ALL' > /tmp/perforce
sudo chmod 0440 /tmp/perforce
sudo chown root:root /tmp/perforce
sudo mv /tmp/perforce /etc/sudoers.d
```

**Warning**  Failure to have `sudo (NOPASSWD)` access on all systems when setting up the system will result in the `p4cmgr_init` or `p4cmgr_add` command.
becoming unresponsive. If this happens, you will need to stop the stalled commands using Ctrl+c, configure the sudo (NOPASSWD) permission where it is missing and then re-run the p4cmgr init or p4cmgr add command.

6. Make sure P4PASSWD is not set. For example:

```
unset P4PASSWD
```

7. Configure the network to support the server cluster:

- All hosts must be able to see (ping) one another on the network.

- In /etc/hosts ensure that localhost resolves correctly.

Remove any entry for 127.0.0.1 in /etc/hosts.

If you must have an entry for 127.0.0.1 in /etc/hosts, put it above the entries for the cluster hosts. Make sure that the host name comes before any reference to localhost. Make sure that the fully-qualified domain name is included before any localhost domain entry. This is to ensure that when any Perforce server on this host starts up and binds to the machine's host name, it does not bind to localhost/127.0.0.1 as this will allow connections from localhost only.

```
127.0.0.1   localhost localhost.localdomain
1.2.3.4  host1.example.com host1
1.2.3.5   host2.example.com host2
```

The fully-qualified host names of any depot master or depot standby will be required by Perforce licensing if a license is required for your server cluster.

- If you use /etc/hosts to manage IP addresses, define all the cluster hosts in /etc/hosts on every cluster host.

- If you use DNS, make sure all the cluster hosts are registered with DNS.

8. Disable installed firewalls on all server cluster hosts. If this is not possible or desirable, make sure that the following ports are open.

**Incoming ports:**

- SSH: 22

- SaltStack: 4505, 4506

- Perforce processes: 1666-1700
These ports are for depot master, depot standby, workspace, and router hosts. Not all ports will be used; once the cluster is set up, you can reduce this to active/configured ports only.

- **NFS**: 111, 2049
- **Zookeeper**: 2380, 3999-4999

Zookeeper hosts only, including the P4CMGR host.

**Outgoing ports** (access from hosts to outside the cluster firewall):

- **SMTP**: 25

  For the P4CMGR host only, to allow `crontab` to send emails as part of the regularly scheduled backup.

- **HTTP**: 80 (needed for `apt/yum`)
  
  **HTTPS**: 443 (needed for `apt/yum`)

  **FTP**: 21 (needed for `apt/yum`)

9. Mount a shared, writable, fault-tolerant filesystem on `/p4/archives` on the nodes that will serve as depot master, depot standby, and workspace servers.

   This directory must be owned by `perforce:perforce` and have **770** permissions

10. If you are using NFS to provide your fault tolerant, shared filesystem, set NFS client configuration options using the values described in the following table. For example, enter the following in `/etc/fstab`, all on one line:

   ```
   NFS-host:shared-dir /p4/archives nfs
   rw,hard,_netdev,sync,auto,nodev,noexec,nosuid,intr 0 0
   ```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rw</code></td>
<td>Read-write (default).</td>
</tr>
<tr>
<td><code>hard</code></td>
<td>NFS share must be available. This means that NFS requests are retried indefinitely.</td>
</tr>
<tr>
<td><code>_netdev</code></td>
<td>The file system resides on a device that requires network access; this prevents the system from attempting to mount filesystems until the network has been enabled on the system.</td>
</tr>
<tr>
<td><code>sync</code></td>
<td>To protect data integrity, do not use NFS caching on writes.</td>
</tr>
<tr>
<td><code>auto</code></td>
<td>Automatically mount at boot time (default).</td>
</tr>
<tr>
<td><code>nodev</code></td>
<td>Perforce does not actually create device nodes for devices.</td>
</tr>
</tbody>
</table>
noexec Executables should not be run directly from a Perforce archive directory. Triggers in the archive are copied to a temporary directory before they are executed.

nosuid SetUID executables should not be run directly from a Perforce archive directory.

intr Allow keyboard interrupts when the server is down.

The NFS server should export the archives share as read-write with sync enabled, and no_subtree_check. For example:

```
/archives *(rw,sync,no_subtree_check)
```

11. Optional: Enable SSH trust from the P4CMGR host to all other cluster hosts. This should be done for the user who will be running the `p4cmgr init` and `p4cmgr add` commands. If this is not done in advance, you will be prompted to accept SSH key fingerprint during the execution of the `p4cmgr init` and `p4cmgr add` commands.

The user for whom trust is established does not need to be root; however, that user does need to have `sudo (NOPASSWD)` permission on all nodes.

12. If you are installing on Red Hat, you must have the optional RPM repository enabled on all systems in the cluster. (This includes P4CMGR host, Zookeeper, depot master, depot standby, workspace, and router hosts if these are running Red Hat.) Use the following command:

```
sudo subscription-manager repos --enable=rhel-6-server-optional-rpms
```

13. Enclose the clustered servers within a firewall to prevent non-administrative direct access to the internal servers of the cluster. Standard Perforce users should only be able to access the cluster through the cluster router. (The firewall should be configured to allow access to the P4PORT of the router.)

After this preparatory work is completed, the following information should always be available to the administrator responsible for installing and monitoring the cluster:

- The name of the shared file system to be used for archives and shelves.
- The name of the hosts where cluster servers are to be installed and the type of server to be installed there.

**Important** You may not use an IP address in lieu of a host name.

### Installing P4CMGR package

To install P4CMGR, follow directions given for your operating system, in the next subsections.
p4cmgr administrative work can be done by logging in as root, or by logging in as a non-root user who is allowed to get a root shell (like sudo bash). The admin must be a member of the perforce group.

**CentOS and Red Hat**

Working on the host where you plan to install p4cmgr, follow these steps to install P4CMGR:

1. **Make sure the file /etc/yum.repos.d/epel.repo includes the following content.**
   
   This provides the dependencies required by SaltStack and ensures that Salt packages are installed from Perforce’s repository.

   ```
   [epel]
   name=epel
   baseurl=http://ftp.linux.ncsu.edu/pub/epel/6/$basearch
   failovermethod=priority
   enabled=1
   gpgcheck=1
   exclude=salt salt-*
   ```

2. **Make sure the file /etc/yum.repos.d/perforce.repo includes the following content.**
   
   This ensures that the packages for Perforce packages for the Perforce server, broker, and p4zk as well as our re-hosted Zookeeper and SaltStack packages can be installed.

   ```
   [perforce]
   name=Perforce
   baseurl=http://package.perforce.com/yum/rhel/6/x86_64/
   enabled=1
   gpgcheck=1
   ```

3. **Run the following commands to install P4CMGR:**

   ```
   sudo rpm --import http://package.perforce.com/perforce.pubkey
   sudo yum install perforce-cluster-manager
   ```

   You are now ready to set up your cluster.

**Ubuntu**

Working on the host where you plan to install P4CMGR, follow these steps:

1. **Make sure the file /etc/apt/sources.list.d/p4cmgr.list contains the following content.**

   ```
   deb http://package.perforce.com/apt/ubuntu precise release
   ```
2. Make sure the file `/etc/apt/preferences.d/salt-perforce-pin-1000` includes the following content.

This prefers Perforce Salt-related packages.

```
Package: *salt-*
Pin: release o=package.perforce.com
Pin-Priority: 1000
```

3. Run the following commands to install P4CMGR.

The first line ensures that no Perforce packages are installed: "rc" packages are okay; "ii" packages are not okay. The `sudo` command before `apt-key` is not needed if you have root privileges.

```
dpkg -l | egrep 'perforce'
wget http://package.perforce.com/perforce.pubkey -O - | sudo apt-key add -
sudo apt-get update
sudo apt-get install perforce-cluster-manager
```

You are now ready to set up your cluster.

### Setting up the server cluster

Using the information given to you by IT, you can now set up the server cluster. The following instructions assume that you are installing a system from scratch. Steps 2 and 3 ask you to validate your environment. If you are the agent who set up the environment, you can skip these steps.

If you are asked to supply a Perforce superuser password, the password you specify must comply with the Perforce Security level 3 password requirements (eight characters or more: two character cases, digits, and/or special characters). As a best practice, this should be different from your Unix root password.

**Important**
The same host name format (simple or fully-qualified domain name) must be used in all `p4cmgr` commands.

1. **Log in to the machine where you have installed P4CMGR.**

   You will need to have `sudo (NOPASSWD)` permissions on all systems in the cluster during setup.

2. **Check that all your cluster nodes, including the P4CMGR host, are adequately prepared. Log in to each of the cluster nodes as the user who will run the P4CMGR commands, and run the following commands:**

   - Check that your groups include the `perforce` group.
• Check `sudo` privileges on the P4CMGR host. Output should show that you ran `id` as root.

```
sudo id
```

• Check the archive share; it should show that `/p4/archives` is a shared filesystem.

```
df -h /p4/archives
```

• Check that the drive is readable and writable by the `perforce` OS user and group.

3. Check `ssh` access to all cluster nodes.

• Check `ssh` connectivity to the host where the depot master will be located.

```
ssh my-depot1
```

• Check ssh to other hosts in the cluster. For example:

```
for $host in my-depot1, my-depot2, my-wr1;
do ssh $host hostname;
done
```

4. Initialize your depot master. The `p4cmgr init` command creates a new depot master and takes a checkpoint. Use the option `-l` to specify the full path to the license file for the depot master if you anticipate having more than twenty users. For example:

```
p4cmgr init -s my-ds1 my-cluster my-depot1
```

The `p4cmgr init` shown in the example creates a `p4d` service on the host with the name `my-depot1`. The `p4d` service that you create will function as the depot master. Given that this service might fail over to the depot standby, it is recommended you do not include its role in the name; for example, do not name it something like `my-depot-master`.

Command processing might require an SSH access prompt for the host signature or for the password of the current operating system user on the remote machine.

If the `p4cmgr init` command initializes the depot master from a checkpoint, the user must also enter the user name and password for a valid Perforce super user in the checkpoint being imported. If not starting from a checkpoint, the user is prompted for a password.
and confirmation, and a new Perforce super user with the same user name as the current operating system user will be created.

- If you want to create the super user with a different user name than the current operating system user, set P4USER before running the p4cmgr init command.

- If you are running p4cmgr init as root, you will be prompted for a Perforce super user name, and then a password and confirmation that conforms to Perforce security level 3.

This command takes a while to complete because it does substantial setup work. While it is completing, you might be asked to enter a name for the initial Perforce super user; you can enter any name except root. If you are not prompted for a name, that means your current logged in Unix user name will be used.

5. **Log in with your Perforce super user account and password.**

(The password you specify must comply with the Perforce security level 3 password requirements (eight characters or more: two character cases, digits, and/or special characters). As a best practice, this should be different from your Unix root password.)

```
p4 -p my-depot:1667 login
```

6. **Confirm that your cluster has been properly initialized by using the p4cmgr status command.**

Furnish your Perforce super user username and password if prompted to do so.

```
p4cmgr status
```

If the cluster has failed to initialize, use the p4cmgr remove command to remove the depot master and try again. See “Troubleshooting installation issues” on page 28 for more information.

The p4cmgr status output should show that Zookeeper is running. You should see output like the following:

```
Processes: | - USER PID
%CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
<other_processes> - root 5447
  0.0  5.5 1079824 28080 ?  S 06:17 0:09 java
  -Dzookeeper.log.dir=/opt/perforce/log \output pruned for legibility
```

Check that cluster-related variables have been set as follows. The scope of environment variables and configurables is indicated just before the name and value pair.
### Configurables for depot1:
- any: security = 3
- any: unicode = 1
- depot1: P4JOURNAL = /p4/journals/journal-depot1
- depot1: P4PORT = 1667
- depot1: cluster.id = rca-cluster
- depot1: journalPrefix = /p4/journals/journal-depot1
- depot1: p4zk.log.file = /p4/logs/p4zk-log-depot1
- depot2: P4PORT = 1667
- depot2: cluster.id = rca-cluster
- depot2: db.replication = readonly
- depot2: journalPrefix = /p4/journals/journal-depot2
- depot2: lbr.replication = shared
- depot2: p4zk.log.file = /p4/logs/p4zk-log-depot2
- depot2: startup.1 = journalcopy -i 0 -b 1
- depot2: startup.2 = pull -l -i 1 -b 1
- rca-cluster: P4TARGET = rca-ds1:1667
- rca-cluster: lbr.autocompress = 1
- rca-cluster: monitor = 1
- rca-cluster: net.keepalive.count = 4
- rca-cluster: net.keepalive.idle = 10
- rca-cluster: net.keepalive.interval = 2
- rca-cluster: p4.utils.dir = /opt/perforce/libexec
- rca-cluster: rpl.journal.ack = 1
- rca-cluster: rpl.journal.ack.min = 0
- rca-cluster: server.depot.root = /p4/archives
- rca-cluster: serviceUser = cluster-replicator
- rca-cluster: zk.host.port.pairs = rca-p4cmgr.das.perforce.com:2380,rca-ws1:2380
- wsserver1: P4PORT = 1667
- wsserver1: cluster.id = rca-cluster
- wsserver1: db.replication = readonly
- wsserver1: journalPrefix = /p4/journals/journal-wsserver1
- wsserver1: lbr.replication = shared
- wsserver1: p4zk.log.file = /p4/logs/p4zk-log-wsserver1
- wsserver1: startup.1 = pull -i 0 -b 1

For detailed information about the use and meaning of configurables, see Chapter 3, “Configuring and Reconfiguring a Server Cluster” on page 31.

7. **Add two or four more Zookeeper nodes.**

   A Zookeeper service is automatically installed on the P4CMGR node as part of cluster initialization. You must explicitly add two or four more nodes to support Zookeeper fault tolerance. For example:

   ```
   p4cmgr add -s zk1 zookeeper router-host
   p4cmgr add -s zk2 zookeeper ws1_host
   ```

   It is probably safest to locate Zookeeper services on workspace and router nodes. You may not install more than one Zookeeper service on a single node.
8. Respond with **yes** when prompted to restart the cluster.

```
p4cmgr restart
```

You must restart the cluster after adding Zookeeper nodes because Perforce services are not aware of the new Zookeeper configuration until the services restart. The Zookeeper services must also be restarted to find out about one another. The `p4cmgr restart` command restarts Perforce services and also restarts the Zookeeper services in the correct order.

9. Check the cluster status to make sure that Zookeeper services are running on the right hosts.

```
p4cmgr status
```

The `zk.host.port.pairs` configurable for all `p4d` services should show the nodes running Zookeeper services.

```
myCluster: zk.host.port.pairs = myHost:2380,myZ2Host:2380...,myZ3Host:2380
```

### Add depot standby

1. Add the depot standby.

```
p4cmgr add -s my-svc depot my-DS2
```

Use the `-l` option if needed to specify the full path to the license file for the depot standby.

2. Check that the new service has been added correctly.

After adding a new service, you might want to check it has been added correctly. You can run `p4cmgr status` to do this:

```
p4cmgr status
```

If the depot standby has not been added, use the `p4cmgr remove` command to remove the depot standby and try again. See “Troubleshooting installation issues” on page 28 for information.

### Add workspace servers

You are not required to add workspace servers if your main reason for creating a cluster is getting a high availability configuration.
Chapter 2. Installation and Set Up

**Important**

If you do not add workspace servers, the router will assume an HA-only configuration. If you decide to add workspace servers later, you will need to remove the router and then add it again after adding the workspace servers.

1. **Use a command like the following to add a workspace server.**

   ```
p4cmgr add workspace -s my-ws1 my-ws1-host
   ```

   Repeat the command for each workspace server you want to add. Specify a unique service name for each workspace server.

2. **Check that the workspace server(s) have been added.**

   ```
p4cmgr status
   ```

   If the workspace server has not been added, use the `p4cmgr remove` command to remove the workspace server and try again. See “Troubleshooting installation issues” on page 28 for information.

For information on how you can assign workspace servers to a given set of users, see “Specifying workspaces for new users” on page 24.

**Add the cluster router**

If you add a router when there are no workspace servers, it is automatically configured to access the depot master. Optionally, multiple routers are supported.

Use a command like the following to add a router:

```
p4cmgr add router -s my-router my-router-host
   ```

The configuration file for the router can be found in `/p4/data/serverid`, where `serverid` is the server id you assigned the router when you added it to the cluster. You can include rules in this file that determine which users are assigned to which workspace servers; see “Specifying workspaces for new users” on page 24 for more information.

**Create a checkpoint**

After you have set up your cluster, you can use the `p4cmgr backup` command to create a checkpoint. The command takes a checkpoint of the depot standby and rotates the journal on the depot master.

```
p4cmgr backup
   ```
Chapter 2. Installation and Set Up

The \texttt{p4cmgr init} command schedules a regular weekly backup using the standard Linux \texttt{cron} facility; you can change the interval using standard Linux \texttt{crontab} commands. The \texttt{crontab} is owned by the operating system user \texttt{perforce}.

For more information about the backup and recovery process, see Chapter 4, "Backup and Recovery" on page 41.

**Specifying workspaces for new users**

You can use the router's configuration file to specify how new users can be assigned workspace servers. You might want to do this so that users are able to share shelves. The following example shows how you might select one of two workspace servers based on the first letter of the user's name. Such commands need to be added to all of your routers' configuration files.

```plaintext
command: .*
{  
  user = "^[a-mA-M]*";
  action = redirect;
  destination = wkspc1;
}
command: .*
{  
  user = "^[n-zN-Z]*";
  action = redirect;
  destination = wkspc2;
}
```

The next example illustrates how you might route all the members of the server dev team to the same workspace server:

```plaintext
command: .*
{  
  user = "^(woz|linus|ritchie|turing)$";
  action = redirect;
  destination = wkspc1;
}
```

Note the use of regular expressions to specify the users. Regular expressions are enclosed in double quotes to avoid problems stemming from the use of reserved words. Redirect clauses in the router configuration file override any conflicting routing information in the router's db.routing table.

**Setting the location and format of log files**

By default depot, workspace, router and \texttt{p4zk} logs are placed in the /p4/logs directory. File names for server logs are \texttt{log-serverid}; file names for \texttt{p4zk} logs are \texttt{p4zk-log-serverid}.

By default, cluster logging uses unstructured logs. For directions on how to enable and configure structured logging, see Logging and structured log files in the \texttt{Perforce Server}.
Structured logging is defined on a per-server basis. When specifying a log name, remember to specify its directory. For example:

```bash
p4 configure set myserverid#serverlog.file.1=/p4/logs/errors.csv
```

To help with debugging, it would be best if you stored your structured log files in a directory that refers to the server. This way, if there is debugging to be done, it would be clear which server a particular log file is related to. For example:

```bash
p4 configure set myserverid#serverlog.file.1=/p4/logs/my_serverID/errors.csv
```

To change the directory where the log file is saved, follow these steps:

- To change the location of the server logs, update the following file by setting the `P4LOG` entry in this file to the new logfile name (must be done per `p4d` service on the host where it is installed).
  
  ```bash
  /etc/perforce/p4dctl.conf.d/serverid
  ```

- To change the location of `p4zk` logs, set the service-scoped configurable `serverid#p4zk.log.file=new-p4zk-log-name-and-path`. For example:

  ```bash
  p4 configure set myServer#p4zk.log.file=/p4/logs/myServer/p4zk.log
  ```

### Creating a client

Cluster clients are just normal Perforce clients that connect to the cluster router, and then go on to map their workspaces in the normal way. For information about authenticating users, see “Securing the cluster and authenticating users” on page 27.

### Adding a license file

You can specify a license file when you first create the cluster. If you do not, you can add a license later by following these steps:

1. Place a license file in the root directory of the standby and in the root directory of the master.
2. Restart the standby.
3. Restart the master.

### Using a server cluster in a distributed environment

You can use the cluster in a distributed environment as follows:

- You can connect a client through a proxy to a cluster router. Client requests are then routed to a workspace server if one exists, or to the depot master otherwise.
The proxy’s **P4TARGET** should point to the cluster router.

- You can connect an edge server to a cluster router. Requests from the edge server are then routed directly to the depot master.

To implement this architecture, the edge server’s **P4TARGET** should point to the cluster router. You must also add a rule to the cluster router configuration file to forward requests from the edge server directly to the depot master. For information about configuring the cluster router to forward requests, see “Forwarding requests from edge to depot master” on page 27.

Both options allow you to connect a geographically distant client to a server cluster. The client could then benefit from the cluster’s automatic failover scheme. The following picture shows what a distributed environment that includes a server cluster would look like. It shows clients’ connection through router **Router**, to a workspace server; an edge server connection via the **Router-edge** router to a depot master; and a remote client connecting via a proxy to a workspace server via router **Router-proxy**.

**Figure 2.3. Distributed Environment with Perforce Clustering**
The topology shown in the preceding figure illustrates a number of points:

- Neither the edge server nor the proxy uses the cluster's shared file system. They each have their normal expected local storage: an archive cache for the proxy, an archive and database for the edge server.

- To improve performance it is best if the proxy and the edge server each use a separate router. Bottlenecks tend to happen at the router. However, requests from external servers and cluster clients can all go through a single router if you wish.

- If the depot master fails, the depot standby takes over and continues to handle requests originating from clients, the proxy, or the edge server.

**Forwarding requests from edge to depot master**

To have the cluster router forward requests from the edge server to the depot master, add a clause like the following to the router configuration:

```
command: .*
{
    user = my_edge_serviceuser;
    action = redirect;
    destination target;
}
```

You must specify the name of the edge's service user for `my_edge_serviceuser`. This might be different than the router's service user.

Note that the `destination` clause indicates that the request should be routed to the current default target, that is, the current depot master. No specific server id is given because the value of this id might change after failover.

**Securing the cluster and authenticating users**

The general guidelines for security in a server cluster are as follows:

- Only the admin may access internal servers directly. Even for the administrator, discretion is advised in the use of the restricted commands described in “Restrictions on using p4cmgr and p4 commands” on page 57.

- Access to servers within the cluster must be prevented by a firewall.

- Servers within the cluster do not use SSL connections. This improves performance and makes administration easier.

- Data servers (`p4d`) are set to security level 3; all user passwords must conform to security level 3 requirements (eight characters or more: two character cases, digits, and/or special characters).

- Servers within the cluster are automatically assigned service user names.
Router nodes are set up with SSL by default. You must explicitly specify the `--noss1` option to not use SSL.

Users are authenticated and granted access rights using normal means. See the *Perforce Server Administrator’s Guide: Fundamentals* for information. Auto-creation of users is on by default.

### Migrating from an existing installation

Migrating to a server cluster from an existing Perforce deployment — whether using a standard server or an edge-commit architecture — involves a host of issues that are best addressed by Perforce support. If you want to update your current Perforce installation, please contact Perforce support for directions.

**Important**

This is an early release of clustering, and requires Perforce consulting engagement for deployment.

### Troubleshooting installation issues

This section describes some common problems that might occur during installation and explains how you can address them.

- **I cannot run `p4cmgr init` after the `p4cmgr` package installation completes.**

  The P4CMGR installation process creates the `perforce` group. To run `p4cmgr init`, the user must be a member of the `perforce` group. Add the existing user to the group:

  ```
  usermod -a -G perforce myUser
  ```

- **The cluster fails to initialize properly or an add command fails to complete or is unable to add the specified service.**

  Use the `p4cmgr remove` command to remove the service that failed to be added, then retry the `p4cmgr init` or `p4cmgr add` command.

  ```
  p4cmgr remove -s myserverid --force depot myhost
  ```

- **Where does the action of `p4cmgr remove`, `p4cmgr stop`, or `p4cmgr start` get logged?**

  These actions are logged in the p4cmgr log, which is found at `/opt/perforce/p4cmgr/logs/p4cmgr.log`.

- **How are patches applied to clustered servers?**

  Run `yum update`, or `apt-get update` and `apt-get upgrade`, and get the latest patch builds. Run this command on each machine in the cluster. All services on an upgraded node must be restarted.
• How do I change the default unicode mode and have it apply?

You cannot change unicode mode once the cluster has been initialized. Unicode is always enabled for any cluster not initialized from a checkpoint. In order to get a non-unicode cluster, the cluster must be initialized from a non-unicode checkpoint.
The configuration of a server cluster involves two aspects:

- The use of configuration variables to define the behavior of the cluster and the location of its resources.
- Changing the original topology of the cluster by adding or removing nodes.

This chapter explores both these aspects.

**Cluster-related configuration variables**

This section describes the use of configuration variables to manage the performance of a cluster. These variables are either specific to this environment or they are existing configurables that must have certain values for servers that take part in a cluster.

You set cluster-related configurables as you set any configurables, using the `p4 configure set` command. You may issue the command to any server. If you send it to a workspace server, it will be automatically forwarded by the workspace server to the depot master. You may not send the command to the depot standby because that server is read-only and will refuse the command.

The following table is a summary of the cluster-related configuration variables. Existing configurables are marked with an asterisk (*). For additional detail about each configurable, see “Configurables reference” on page 34.

<table>
<thead>
<tr>
<th>cluster.id</th>
<th>The cluster name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>* journalPrefix</td>
<td>The prefix applied to the rotated journal file.</td>
</tr>
<tr>
<td>* monitor</td>
<td>Sets the level of monitoring activity.</td>
</tr>
<tr>
<td>* lbr.autocompress</td>
<td>The default value of 1 sets all text files to ctext.</td>
</tr>
<tr>
<td>* net.keepalive.count</td>
<td>Number of unacknowledged keepalives before failure.</td>
</tr>
<tr>
<td>* net.keepalive.idle</td>
<td>Idle time (in seconds) before starting to send keepalives.</td>
</tr>
<tr>
<td>* net.keepalive.interval</td>
<td>Interval (in seconds) between sending keepalive packets.</td>
</tr>
<tr>
<td>p4.utils.dir</td>
<td>The directory containing binaries and utilities used by Perforce servers. The p4zk binary is stored here.</td>
</tr>
<tr>
<td>* P4JOURNAL</td>
<td>The full path to the current journal file for the depot master. By default this is set to /p4/journals/journalfile.</td>
</tr>
<tr>
<td>* P4TARGET</td>
<td>The P4PORT of the starting depot master. This value is automatically updated during a failover.</td>
</tr>
</tbody>
</table>
### p4zk.log.file
The location of the p4zk log file.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p4zk.socket.path</td>
<td>The full path to the p4zk Unix domain socket.</td>
</tr>
<tr>
<td>rpl.counter.hook</td>
<td>Protects workspace servers against global system counters from going backwards. Must be set to 0.</td>
</tr>
<tr>
<td>rpl.journal.ack</td>
<td>Desired number of depot standby acknowledgements before releasing update results to the workspace server. This configuration only applies when there are at least that number of depot standbys available. Currently only one depot standby is supported.</td>
</tr>
<tr>
<td>rpl.journal.ack.min</td>
<td>When the minimum of available depot standby is less than the value of rpl.journal.ack, the cluster is operating in degraded mode. During degraded mode, this configurable indicates the minimum number of acknowledgments allowed to continue operation. We currently support zero or one.</td>
</tr>
<tr>
<td>serviceUser</td>
<td>A value used for authenticating members of a cluster for inter-server communication.</td>
</tr>
<tr>
<td>server.depot.root</td>
<td>Points to the shared file system.</td>
</tr>
<tr>
<td>zk.host.port.pairs</td>
<td>A comma-separated list of host:port pairs that reference the server nodes of the distributed Zookeeper service.</td>
</tr>
</tbody>
</table>

You can check the current setting of these variables using the `p4cmgr status` command; part of the output lists these variables.
The variables you are most likely to change are the `net.keepalive.*` variables; adjusting these is your way of minimizing unnecessary failover actions. For more information, see Chapter 5, “Monitoring and managing server cluster” on page 51. You might also need to change the `p4zk.log.file` variable if you need to relocate `p4zk` log files, although this is not recommended.

The scope of each variable is indicated before the variable name. The scope `any` refers to any cluster in the environment. See the next section for a discussion of variable scope.

### Variable scope

Configuration variables are scoped as follows:

- **global**: configuration options for the `P4NAME=all`
- **cluster**: configuration options for all members of a cluster.
• **server**: configuration options specified for a server designated by `P4NAME` (server ID)

• **command-line** overrides: configuration options that are overridden on the command line

The scoping of configuration variables governs the order in which they are applied: configuration values specified on the command line override those provided for a server, which override cluster-level settings, which override global settings.

Variable scope also governs when you may change the value of a variable: variables that affect the entire cluster may be changed after the `p4cmgr init` command; variables that affect a given server may be changed after you add the server to the cluster.

### Changing configuration values

Observe these guidelines when attempting to change values of cluster-related configurables:

• Never change the values of these configurables: `cluster.id`, `db.replication`, `lbr.replication`, `journalPrefix`, `P4TARGET`

• If you change the value of any configurables other than `rpl.journal.ack` or `rpl.journal.ack.min`, you will need to restart the cluster.

### Configurables reference

The following table describes configurables and environment variables used in a cluster environment: some are specific to this environment; others are existing configurables whose value must be set in a specific way for this environment. Existing configurables are marked with an asterisk (*). The scope and default value for each configurable is indicated. Also indicated is whether a cluster restart is required when the configurable value is changed.

<table>
<thead>
<tr>
<th>Configurable</th>
<th>Description</th>
<th>Scope</th>
<th>Default</th>
<th>Restarts Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster.id</td>
<td>A cluster name that scopes the configuration parameters that apply to each node in the cluster. When you add a node to a cluster, the set of configuration parameters referenced by the cluster id will be applied conditionally to each member of the cluster, depending on the member's role: depot standby, workspace, or router.</td>
<td>cluster</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>You create the cluster id when you use the <code>p4cmgr init</code> command to initialize and configure a cluster.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: <code>usaCluster</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scope: cluster</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Must never change.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>* journalPrefix</td>
<td>The prefix applied to the rotated journal file. Not required.</td>
<td>cluster</td>
<td>journal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: <code>usaJournal</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: <code>journal</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Setting</td>
<td>Description</td>
<td>Default</td>
<td>Restart Required</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------</td>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td><strong>monitor</strong></td>
<td>Sets monitoring activity. Monitoring is required to be on in a server cluster. You can set it to 1 or 2.</td>
<td>1</td>
<td>Must restart on change.</td>
<td></td>
</tr>
<tr>
<td><strong>lbr.autocompress</strong></td>
<td>The default value of 1 sets all text files to ctext.</td>
<td>1</td>
<td>Must restart on change.</td>
<td></td>
</tr>
<tr>
<td><strong>net.keepalive.count</strong></td>
<td>Number of unacknowledged keepalives before failure.</td>
<td>4</td>
<td>Must restart on change.</td>
<td></td>
</tr>
<tr>
<td><strong>net.keepalive.idle</strong></td>
<td>Idle time (in seconds) before starting to send keepalives.</td>
<td>10</td>
<td>Must restart on change.</td>
<td></td>
</tr>
<tr>
<td><strong>net.keepalive.interval</strong></td>
<td>Interval (in seconds) between sending keepalive packets.</td>
<td>2</td>
<td>Must restart on change.</td>
<td></td>
</tr>
<tr>
<td><strong>p4.utils.dir</strong></td>
<td>The directory containing binaries and utilities used by Perforce servers. The p4zk binary is stored here.</td>
<td>/opt/perforce/libexec</td>
<td>Must restart on change.</td>
<td></td>
</tr>
</tbody>
</table>
### Chapter 3. Configuring and Reconfiguring a Server Cluster

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P4JOURNAL</strong></td>
<td>The name of the current journal file for the depot master. By default this is set to <code>P4ROOT/journal</code>.</td>
</tr>
<tr>
<td></td>
<td>Scope: cluster or server</td>
</tr>
<tr>
<td></td>
<td>Must restart on change.</td>
</tr>
<tr>
<td><strong>P4TARGET</strong></td>
<td>The <code>P4PORT</code> of the starting depot master.</td>
</tr>
<tr>
<td></td>
<td>Scope: cluster</td>
</tr>
<tr>
<td></td>
<td>Default: <code>journal</code></td>
</tr>
<tr>
<td></td>
<td>Must restart on change.</td>
</tr>
<tr>
<td><strong>p4zk.log.file</strong></td>
<td>The location of the <code>p4zk</code> log file.</td>
</tr>
<tr>
<td></td>
<td>Scope: cluster</td>
</tr>
<tr>
<td></td>
<td>Default: <code>P4ROOT</code></td>
</tr>
<tr>
<td></td>
<td>Must restart on change.</td>
</tr>
<tr>
<td><strong>p4zk.socket.path</strong></td>
<td>The full path to the <code>p4zk</code> Unix domain socket. This configurable is only valid if the server’s service field is set to <code>depot-server</code> or <code>depot-standby</code>.</td>
</tr>
<tr>
<td></td>
<td>Example: <code>p4zk_socket</code></td>
</tr>
<tr>
<td></td>
<td>Scope: cluster or server</td>
</tr>
<tr>
<td><strong>rpl.counter.hook</strong></td>
<td>Protects workspace servers against global system counters from going backwards. Must be set to 0.</td>
</tr>
<tr>
<td><strong>rpl.journal.ack</strong></td>
<td>Controls behavior when update results are sent back to the requester.</td>
</tr>
<tr>
<td></td>
<td>• If zero, results are sent back immediately. If greater than 0, the value represents the number of depot standbys that must acknowledge that they have received and persisted the journal records for that transaction before the depot master will send back the results to the requester.</td>
</tr>
<tr>
<td></td>
<td>• If the number of available depot standbys falls below the configured value for <code>rpl.journal.ack</code>, the value of <code>rpl.journal.ack.min</code> is also taken into consideration when deciding if or when to return the results to the requester.</td>
</tr>
<tr>
<td></td>
<td>Currently only one depot standby is supported, so allowed values for this configuration item are 0 or 1.</td>
</tr>
</tbody>
</table>
Chapter 3. Configuring and Reconfiguring a Server Cluster

- A value of zero removes any guarantees that you will have a consistent standby replica for failover should the depot master stop functioning.

- A value of one assures that the depot standby will be consistent with any results seen by users of the cluster. It does this by delaying the return of the results to the requester until the depot master has received an acknowledgement from the depot standby.

Scope: cluster
Default: 1
No restart on change.

| rpl.journal.ack.min | Controls behavior when the cluster is operating in degraded mode. The cluster operates in degraded mode when the number of available depot standbys falls below the configured value of rpl.journal.ack. In degraded mode, the behavior is as follows:

- If rpl.journal.ack.min is greater than the number of remaining depot standbys, the depot master will stop sending update results to the requester.

- If rpl.journal.ack.min is equal or less than the number of remaining depot standbys, the depot master will send back update results to the requester after it receives acknowledgements from all remaining depot standbys.

Currently only one depot standby is supported. In this case:

- The default value of zero allows the cluster to continue processing updates even though there is not a consistent backup of the transaction. That is, high availability is not guaranteed.

- A value of one prevents updates from being returned when the depot standby is down.

Scope: cluster
Default: 0
No restart on change.

| serviceUser | A value used for authenticating members of a cluster for inter-server communication. Each workspace server and depot standby must have a defined serviceUser value. This value can be defined per server or it can be shared by all members of a cluster. |
Chapter 3. Configuring and Reconfiguring a Server Cluster

<table>
<thead>
<tr>
<th>Configurable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>p4service</code></td>
<td>Example: <code>p4service</code></td>
</tr>
<tr>
<td>Scope: cluster or per server.id</td>
<td>Default: <code>cluster-replicator</code></td>
</tr>
<tr>
<td>Must restart on change.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Configurable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>server.depot.root</code></td>
<td>Points to the shared file system.</td>
</tr>
<tr>
<td>Scope: cluster</td>
<td>Default: <code>/p4/archives</code></td>
</tr>
<tr>
<td>Must restart on change.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Configurable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>zk.host.port.pairs</code></td>
<td>A comma-separated list of <code>host:port</code> pairs that reference the server nodes of the distributed Zookeeper service. This configurable is only valid if the server’s service field is set to <code>depot-server</code> or <code>depot-standby</code>.</td>
</tr>
<tr>
<td>Scope: cluster</td>
<td>If you change this value, you must restart everything in the cluster, including the Zookeeper services.</td>
</tr>
<tr>
<td>Default: <code>cluster_manager_hostname:2887</code></td>
<td></td>
</tr>
<tr>
<td>Must restart on change.</td>
<td></td>
</tr>
</tbody>
</table>

### Zookeeper configurables

The following configurables are used to configure the performance of the Zookeeper servers, which determine the server’s responsiveness to a Perforce server failure. The default settings for these configurables are lax to prevent false positives. For additional information on Zookeeper administration, see the [Zookeeper Administrator’s Guide: Configuration Parameters](http://zookeeper.apache.org/doc/trunk/zookeeperAdmin.html#sc_configuration).

**tickTime**

The unit of measurement in ms that determines the granularity of time and affects the timing of heartbeat messages from Zookeeper clients to the servers.

Default: `2000`

Location: server config

**minSessionTimeout**

The minimum session timeout in ms that a server will wait for a client to negotiate.
### Chapter 3. Configuring and Reconfiguring a Server Cluster

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>maxSessionTimeout</code></td>
<td>The maximum session timeout in ms that a server will wait for a client to negotiate.</td>
<td>$2 \times \text{tickTime}$</td>
<td>server config</td>
</tr>
<tr>
<td><code>initLimit</code></td>
<td>Maximum time in ticks that the leader Zookeeper server allows a follower Zookeeper server to connect and sync.</td>
<td></td>
<td>server config</td>
</tr>
<tr>
<td><code>syncLimit</code></td>
<td>Maximum time in ticks to allow a follower Zookeeper to sync with other Zookeeper servers.</td>
<td></td>
<td>server config</td>
</tr>
<tr>
<td><code>cnxTimeout</code></td>
<td>Maximum time in ticks for opening connections for leader election notifications.</td>
<td></td>
<td>server config</td>
</tr>
</tbody>
</table>

### Changing server cluster topology

You may need to change the topology of a Perforce cluster. Possible use cases include the following:

- **You add or remove a workspace server.**

  To determine when to add a workspace server, you need to monitor CPU, RAM, disk, disk I/O, and response times. However, excessive load on a workspace server does not necessarily mean that you need another workspace server. It's possible that other workspace servers are under-utilized and that you can simply move some clients from the overloaded workspace server to the under-utilized server instead.

- **You add a second router.**

- **You add a depot standby after a failover has converted the current depot standby into a depot master.**

- **You add a Zookeeper service if an existing one or the node supporting it has died.** You must restart the cluster after adding the service.

  It is best also to use the `p4cmgr remove` command to explicitly remove the defunct Zookeeper service. This is because even a failed service continues to count in the automatic polling done in choosing a new depot master.

In all these cases you can use the `p4cmgr add` command to add services or the `p4cmgr remove` command to remove a service.
If you move clients around in your cluster using `p4 unload` and `p4 load` commands, your router cache (`db.routing`) will contain stale information, pointing to the wrong workspace server. It is best, in this case, to delete the `db.routing` file and let the system rebuild a new one, containing the updated location.
Chapter 4  Backup and Recovery

A Perforce server cluster is designed to provide automatic failover if the depot master fails. This chapter describes the sorts of failures that the cluster can handle and the ones it can’t; it describes how the depot master and depot standby are kept in sync, and it explains how you restore those cluster components that are not automatically recovered.

You should read this chapter to understand what events are handled automatically, what actions you might need to take even in the case of automatic recovery, and what actions you need to take for those events that are not handled by the cluster. This chapter also describes automated scheduled backups.

Automatic failover

A server cluster is highly available but does not support disaster recovery. This means that it can automatically handle the following kinds of failures:

- Mechanical failures causing processes on a machine to stop
- Power failure or operating system failure on a node

Automatic failover from depot master to a depot standby is intended to be transparent and is carried out in the background in such a way that the user should not be aware that anything is amiss. The failover process might take a couple of minutes.

By default the system waits for one second before it decides that a depot master has failed and proceeds to promote the depot standby to depot master. You can change that default value as shown below; the option value -b, which specifies the number of seconds to wait. (The default value of 1 second applies only for `p4 journalcopy` and `p4 pull` startup threads configured by `p4cmgr`; the `p4d` default is still 60 seconds.)

```
p4 -p depot_master_p4port
    configure set standby#startup.1 = journalcopy -i 0 -b 10
```

Although recovery from a failed depot master is automatic, you might want to use a monitoring tool to learn of such events in case additional action is needed. For example, if a depot master fails over to a depot standby, you no longer have a depot standby to assure high availability. In this case, you might want to be notified to add another depot standby. Likewise, if a node that fails has a Zookeeper service on it, you might want to be notified so that you can start another Zookeeper service.

It is possible that the cluster cannot be automatically restored to a valid state after failover if either a workspace or router server is stopped at the time that a failover occurs. See “Troubleshooting failover” on page 48 for information on how to handle this case.

In general, a node will drop out of the cluster rather than continuing to function in a degraded state when it encounters a serious error. Using monitoring software, you can detect and analyze the problem. You can then replace the node or fix the problem and have the node re-join the cluster.

The following types of failures are not automatically handled:
Chapter 4. Backup and Recovery

- Shared file system failure
- Cluster router failure
- Failure due to disaster — the data center is lost
- Network partitions that isolate servers from Zookeeper

The following table describes some of these cases and their effects, and suggests possible corrective action.

<table>
<thead>
<tr>
<th>Situation</th>
<th>Corrective Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared archive device dies or network failure disrupts access to shared archive.</td>
<td>Nodes drop out of cluster as they discover the failure. You must restore the shared archive.</td>
</tr>
<tr>
<td>Shared archive device runs out of disk space.</td>
<td>The depot master shuts down the cluster when it detects the problem.</td>
</tr>
<tr>
<td>Power failure causes the entire cluster to fail.</td>
<td>Cluster does not yet provide disaster recovery. You must restore power, restore the shared archive, and then must restart the cluster.</td>
</tr>
<tr>
<td>Power failure on machine hosting the cluster router.</td>
<td>The cluster is operational for administrators but no client requests can be delivered. You can add another cluster router, or you can determine what is wrong with the failed router, fix it, and restart the router. No other recovery work is needed. The administrator can also provision a redundant second workspace router on a separate node.</td>
</tr>
<tr>
<td>Network failure isolates the depot standby from the depot master, but not from Zookeeper.</td>
<td>The behavior depends on the setting of the <code>rpl.journal.ack</code> configurables and on the number of depot standbys. If the cluster is configured with more than one standby and the <code>rpl.journal.ack</code> configurable requires only one acknowledgment, everything should work as before. If failover occurs, the visible standby will always be chosen because it will have the longer journal. See “Journal management” on page 43 for more information on the use of configurables to control failover.</td>
</tr>
<tr>
<td>Network failure isolates the depot standby from the cluster router, but not from Zookeeper or the depot master.</td>
<td>Cluster will work until failover occurs and the depot standby becomes a depot master. There will then be no route for commands that go directly to the depot master.</td>
</tr>
<tr>
<td>Workspace server experiences problems.</td>
<td>Only the depot master is highly available. If a workspace server encounters a problem, clients whose requests are routed to that server are affected; work may continue on other workspace servers. The affected workspace server</td>
</tr>
</tbody>
</table>
needs to be recovered manually from backups, or by using a replica or standby of that workspace server.

**Important**

It is best to explicitly remove any cluster components that are no longer available, by using the `p4cmgr remove` command.

## Journal management

The process of backing up the depot master includes the steps outlined below. Two processes on the depot standby work together to make the transfer of records as fast and reliable as possible: one is a journalcopy thread on the standby that copies records from the master as quickly as possible; the other is a pull thread on the standby that writes the records to the standby’s database. Understanding this process can help you understand the work that is done to enable automatic recovery.

The following figure illustrates how this is done:

**Figure 4.1. Copying Records From Master to Standby**

The depot master maintains an acknowledgment table that tracks the extent to which the depot standby and the depot master are in sync. Output to the `p4 servers -J` command is generated from this table.

- The journal copy thread transfers records from the depot master to the depot standby’s local file system. The transfer is an exact, byte-for-byte copy of the depot master’s journal.

- The journal copy thread acknowledges having completed the copy after it writes journal records to disk and updates its state file.

Only after the acknowledgment, does the depot master release the updated data to workspace servers. This guarantees that if the depot master has processed a large transaction and fails during the replication of the data, the transaction does not complete until all the data has been read by the depot standby and the state file has been updated. If that doesn’t happen, the user’s update operation fails and must be retried on the newly promoted server.

Two configurables `rpl.journal.ack` and `rpl.journal.ack.min` control the behavior of the cluster with respect to the processing of transactions. The following table shows how configurable settings and available depot standbys affect behavior on update requests.
### Chapter 4. Backup and Recovery

#### Configurables reference

<table>
<thead>
<tr>
<th>ack</th>
<th>min</th>
<th>standby</th>
<th>eff ack req</th>
<th>Stall?</th>
<th>Hang?</th>
<th>Durable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

For more information, see “Configurables reference” on page 34.

- A local p4 pull -L thread applies the copied journal records to the depot standby’s database and updates its state file.

Output to the `p4 servers -J` command looks like this:

```
$ p4 -p 10.0.101.225:1111 servers -J
depot-master '2014/09/08 13:13:58' depot-master 5/258 5/258 wadl/1 1
depot-standby_1 '2014/09/08 13:14:58' depot-standby 5/258 5/258 WAdl/12 1
depot-standby_2 '2014/09/08 13:14:58' depot-standby 5/258 5/258 WAdl/12 1
workspace-server_1 '2014/09/08 13:14:58' workspace-server 5/258 5/258 WaDl/10 1
workspace-server_2 '2014/09/08 13:14:57' workspace-server 5/258 5/258 WaDl/10 1
workspace-server_3 '2014/09/08 13:14:58' workspace-server 5/258 5/258 WaDl/10 1
```

For a detailed explanation of this output, see the `p4 servers -J` command in the **P4 Command Reference**

### Backing up workspace servers

Manual backups are needed to recover from the failure of a workspace server. You have two options: backing up the workspace server while it’s offline, or checkpointing the server through the use of a replica. This section describes both options.

#### Backing up an offline workspace server

1. Wait for a maintenance interval.

2. Stop the router from forwarding new requests to the workspace server by adding this rule *before* all other rules in the router configuration file.
Chapter 4. Backup and Recovery

command: *
{
  action = reject;
  message = "The cluster is currently offline for maintenance."
}

You do not need to restart the router after adding the rule. The broker re-reads its configuration file when it has been changed.

3. **Issue a command like the following to each workspace server and to the depot standby to checkpoint on the next journal rotation:**

   ```
   p4 -p my_ws1:6321 admin checkpoint
   ```

4. **Instruct the depot master to perform a journal rotation by issuing a command like the following:**

   ```
   p4 -p my_ds1:1667 admin journal
   ```

   or by starting the depot master using the command line (on the depot master) with the following option.

   ```
   p4d -jj
   ```

5. **When all checkpoints are finished, remove the rule inserted in Step 2 on page 44**

### Backing up a workspace server using a replica

To produce checkpoints without disrupting the cluster's up-time, you should include read-only replicas in your deployment to mirror a workspace server's databases. This approach offers two advantages: each workspace server can continue to service commands while its replica takes the checkpoint, and if a workspace server fails, a backup database is available to seed a new workspace server.

This approach requires that the VM or machine hosting the workspace server be configured with sufficient resources to hold the metadata databases.

After setting up this configuration:

1. **Instruct each workspace replica and the depot standby to checkpoint on the next journal rotation by issuing a command like the following to each:**

   ```
   p4 -p my_ws1:6321 admin checkpoint
   ```

2. **Issue a command like the following to have the depot master perform a journal rotation:**

   ```
   ```
Chapter 4. Backup and Recovery

```
p4 -p my_ds1:1667 admin journal
```

See “Restoring a failed workspace server” on page 47 for more information.

### Recovering failed servers

The following sections describes the actions that are taken automatically or that must be performed manually in the case of server failure in a server cluster.

#### Restoring a failed depot master

If the depot master fails, the following steps are automatically taken:

1. The depot standby is informed.
2. The depot standby tells the journal copy thread to finish.
3. The journal copy thread persists any buffered records to the local copy of the journal. It then writes the last consistency point to the `statejcopy` file and then truncates the local journal to that consistency point.
4. Once the standby detects that the journalcopy thread has completed, it reports the last consistency point back to `p4zk`, which writes it to the Zookeeper server.
5. The depot standby updates its configuration to make itself the new depot master and it changes the old depot master’s type in its database to depot standby if possible.
6. The depot standby rotates the journal: it writes the epilogue records to the old journal and starts a new journal.
7. The newly-elected depot master restarts as the new depot master and lets `p4zk` know that it’s the new depot master.
8. Zookeeper then updates all the other servers in the cluster to redirect their requests to the new target — the new depot master.

If it happens that the failed depot master comes back to life, it will discover that it is no longer the depot master, it will log a fatal error, and it will exit. It is best to use the `p4cmgr remove` command to remove the now-defunct depot master.

Once restored, the new depot master will handle client or workspace server requests. However, the depot master is no longer highly available until you create and start another depot standby.

#### Restoring a failed depot standby

If the depot standby dies while the depot master is still running, the setting of the `rpl.journal.ack` and `rpl.journal.min` determine whether the cluster should continue running in degraded mode, which enables updates without the guarantee of high availability,
while you rebuild the depot standby. (Set `rpl.journal.min` to 0 to allow the cluster to run in degraded mode.) If you elect to disallow degraded mode, read-only requests will be satisfied, but update requests will hang.

Follow these steps to restore a failed depot standby:

1. Backup and remove the database, locks, log, and journal files.
2. If the `server.id` file is gone from `P4ROOT`, restore it.
3. Restore from the last `depot-server` checkpoint. If this checkpoint is from before the failover, use a command like the following:

   ```
   p4d -r ... -c "set clusterid#p4TARGET=myTarget"
   ```

4. Start up the depot standby.
5. Verify that this server's specification lists it correctly as a standby. Use `p4 configure show all servers` to verify.
6. Use the `p4 servers -J` command to determine when the depot standby gets up to date again. See the section Evaluating Replication Status in the P4 Command Reference for an explanation of how you make that determination.

### Restoring a failed workspace server

When a workspace server fails, clients whose requests are routed to that server are not redirected to an alternate server. You must restore the failed workspace server.

A failed workspace server does not automatically restart. If you have anticipated failure and provided a backup, you can restore a workspace server as follows.

1. Backup and remove the database, locks, log, and journal files.
2. If the `server.id` file is gone in `P4ROOT`, restore it.
3. Restore from the last workspace server checkpoint.
4. Replay the journal records after that checkpoint.
5. Start up the server.
6. Verify this server is configured with a startup thread for the `p4 pull` command.
7. Verify that this server's specification lists it correctly as a workspace server.

### Restoring a failed cluster router

In the absence of an alternate router, a failed cluster router will prevent client requests from reaching workspace servers or the depot master. Follow these steps to restore a failed cluster router.
Chapter 4. Backup and Recovery

1. Verify that the router configuration file still exists. If it does not, replace it.

2. Remove the `db.routing` file (which should be in the same directory as the router configuration file); this is the router's cache.

3. Start up the router.

4. Verify that the `p4zk` process associated with the router is up.

5. Verify the router configuration file reflects the current configuration of workspace servers and the target depot master.

You can shut down and restart a router without any problems. On startup, the p4zk process will fix up the router configuration with the current status of the cluster and it will reuse the router's client cache from the last session.

**Troubleshooting failover**

It is possible that a workspace server was not properly updated during the failover process because it was stopped when the failover occurs. When the workspace server tries to reconnect, it tries to reconnect to the old, failed depot master. You can determine that the workspace server is attempting to communicate with the wrong depot master by running the command `p4 configure show` for that workspace; it will report the old depot master as `P4TARGET`.

To resolve this problem do the following:

1. **Stop the workspace server that is unable to reconnect.**

   ```
   p4cmgr stop -s myServerId
   ```

2. **Run the following command to reset P4TARGET for the server that cannot connect.**

   ```
   p4d -r /p4/data/myServerId -cset clusterID="P4TARGET=new_master:port" -J off
   ```

   The `p4d` binary will be at `/opt/perforce/sbin/p4d`, and will probably not be on the system path. If you get a message telling you that the `p4d` command is not found, run the following command.

   ```
   /opt/perforce/sbin/p4d -r /p4/data/myServerId -cset "P4TARGET=new_master:port" -J off
   ```

3. **Start the updated server.**

   ```
   p4cmgr start -s myServerId
   ```
Scheduled backups

By default, backups of the depot master are performed every Sunday at 2 a.m. The backup and schedule is controlled by a cron task that runs on the P4CMGR cluster host as the perforce UNIX user.

You can view the current backup configuration by running the following command:

```
sudo crontab -u perforce -l
```

This will output information like the following:

```
# Lines below here are managed by Salt, do not edit
# SALT_CRON_IDENTIFIER:NO ID SET
0 2 * * 0 /opt/perforce/p4cmgr/cron/p4cmgr_backup.sh
```

If p4cmgr init is re-run, this configuration might get reset to the default value. Since this is unlikely to happen once a cluster has been configured, it is actually safe to edit the configuration. You can do so to change the frequency or the time the backup is done by running the following command:

```
sudo crontab -u perforce -e
```

You can find documentation on the crontab file format by entering the following:

```
man 5 crontab
```

If an error occurs during the backup, an email is sent to the local perforce user. It is recommended that you configure the local mail system to forward such emails to the appropriate administrator. You will need to configure the firewall to allow outgoing SMTP for this to work.

You can also use the p4cmgr backup command to perform a backup at any time.
Chapter 5  Monitoring and managing server cluster

This chapter describes how you monitor the status of the cluster and of some of the underlying technology that supports high availability. This chapter also includes sections on managing a server cluster.

Monitoring a server cluster

- “Getting status information” on page 51 explains how you use the `p4cmgr status` command to get information about cluster components.

- “Evaluating replication status” on page 52 describes the mechanism used to replicate the depot master and explains how you can interpret the output of the `p4 servers -J` command to determine whether replication is efficient.

- “Monitoring Zookeeper” on page 55 provides an alternate way of getting status information for Zookeeper.

- “Monitoring events” on page 55 explains how you can monitor events that affect the cluster using the `zkMonitor` script.

Getting status information

You can use the `p4cmgr status` command to get status information about a cluster. Output is voluminous because it reports, among other things, each server’s view of the cluster. Output is broken into the following parts:

- **Cluster configuration**: lists the role, host name, and port of each member of the cluster.

- **Ping**: lists all hosts and specifies whether they are up (true) or down (false).

- **Detailed status**: for each server in the cluster it lists the following:
  
  - Configurables set for the environment and for each member of the cluster as seen from this server (not applicable to `master.perforce.com`)
  
  - Information about other servers in the cluster as viewed from this server (not applicable to `master.perforce.com`). Information includes: Host name and port, description, serverid, role, and type.

  - `p4d` services: server status

  - Processes running on this server: this includes `p4zk` process, Zookeeper service if any, journal replication processes, and user-related processes.

Differences between servers’ view of the cluster are cause for concern. For example, each server should have the same value for `P4TARGET`, the port of the current depot master. Differences in this value indicate that a failover operation has not successfully updated all the workspace servers or the routers in the cluster.
In addition to using the `p4cmgr status` command, you can also use the `p4 configure show` command to display the status of configuration options derived from defaults, the environment, and configuration files.

**Evaluating replication status**

“Journal management” on page 43 explains the process by which the depot master is backed up. You must be familiar with this process before you can understand this section.

The high availability feature of the server cluster depends on how efficiently the depot standby is able to replicate the depot master’s records. You can use the `-J` or `--replication-status` option to the `p4 servers` command to check on this process. Given a server `A` and a replica `B`, output for this command gives you two basic pieces of information:

- The size and update time of `A`'s journal.
- For every server, `B`, that has sent a `p4 pull` or `p4 journalcopy` request, information is given as to when that request was sent and what is the persisted and applied state of `B`'s journal. (In the case of a simple master and replica, the persisted and applied numbers are always the same: `B`'s journal is updated by the `p4 pull` command.

When interpreting this information for cluster management, it is important to understand the difference between persisted and applied records. A depot standby replicates depot master records using two operations:

- It uses the `p4 journalcopy` command to copy (`persist`) the depot master's journal to the depot standby's journal.
- It uses the `p4 pull -L` command to apply the copied journal records to the standby's database.

You can look at the output of the `p4 servers -J` command to evaluate the load on various parts of your distributed system and to see how well your replicas are keeping up with the depot master. Growing lag times might be a reason for concern.

The untagged output of `p4 servers -J` looks like this:

```bash
depot-standby_1 '2014/09/08 13:14:58' depot-standby 5/258 5/258 WAdl/12 1
workspace-server_1 '2014/09/08 13:14:58' workspace-server 5/258 5/258 WadL/10 1
workspace-server_2 '2014/09/08 13:14:57' workspace-server 5/258 5/258 WadL/10 1
```

It is easier to interpret this output in tagged form:
... ServerID depot-master
... Updated 2014/09/08 13:13:58
... ServerType depot-master
... PersistedJournal 5
... PersistedSequence 258
... AppliedJournal 5
... AppliedSequence 258
... JAFlags wadL/1
... IsAlive 1

... ServerID depot-standby_1
... Updated 2014/09/08 13:14:58
... ServerType depot-standby
... PersistedJournal 5
... PersistedSequence 258
... AppliedJournal 5
... AppliedSequence 258
... JAFlags WAdl/12 1
... IsAlive 1

... ServerID workspace-server_1
... Updated 2014/09/08 13:14:58
... ServerType workspace-server
... PersistedJournal 5
... PersistedSequence 258
... AppliedJournal 5
... AppliedSequence 258
... JAFlags WaDl/10 1
... IsAlive 1

... ServerID workspace-server_2
... Updated 2014/09/08 13:14:57
... ServerType workspace-server
... PersistedJournal 5
... PersistedSequence 258
... AppliedJournal 5
... AppliedSequence 258
... JAFlags WaDl/10 1
... IsAlive 1

The meaning of the fields are described in the following table:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServerID</td>
<td>The server ID of the server.</td>
</tr>
<tr>
<td></td>
<td>The server ID should always match P4NAME if both are set. We recommend</td>
</tr>
<tr>
<td></td>
<td>setting the server ID, but support P4NAME for backward compatibility.</td>
</tr>
<tr>
<td>Updated</td>
<td>The date and time the requesting server last requested journal records from</td>
</tr>
<tr>
<td></td>
<td>this server (normally the depot master).</td>
</tr>
<tr>
<td>ServerType</td>
<td>The server type. Normally either depot-master, depot-standby,</td>
</tr>
<tr>
<td></td>
<td>or workspace-server. But it could also be standard, replica,</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
forwarding-replica, build-server, edge-server, commit-server, standby, or forwarding-standby.

<table>
<thead>
<tr>
<th><strong>PersistedJournal</strong></th>
<th>The rotation number of the journal to which records are being persisted.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PersistedSequence</strong></td>
<td>The persisted journal position.</td>
</tr>
<tr>
<td>For depot masters, replicas, and workspace servers, the persisted and applied positions are always the same. They differ only for all types of depot standbys.</td>
<td></td>
</tr>
<tr>
<td><strong>AppliedJournal</strong></td>
<td>The rotation number of the applied journal.</td>
</tr>
<tr>
<td><strong>AppliedSequence</strong></td>
<td>The applied journal position.</td>
</tr>
<tr>
<td>For depot masters, replicas, and workspace servers, the persisted and applied positions are always the same. A depot standby server’s <strong>AppliedSequence</strong> value might lag behind its <strong>PersistedSequence</strong> value while its pull -L thread applies new journal records to its database.</td>
<td></td>
</tr>
<tr>
<td><strong>JAFlags</strong></td>
<td>Set of fields printed in upper-case if set or lower-case if not. The numeric value of the flags is displayed after the alphabetic display.</td>
</tr>
<tr>
<td>Common field displays with their associated p4 pull or p4 journalcopy commands are as follows:</td>
<td></td>
</tr>
<tr>
<td>• <strong>WaDl/12</strong>: journalcopy -i 0</td>
<td></td>
</tr>
<tr>
<td>• <strong>WaDl/10</strong>: pull -i 0</td>
<td></td>
</tr>
<tr>
<td>• <strong>wAdl/4</strong>: journalcopy -i 1</td>
<td></td>
</tr>
<tr>
<td>• <strong>waDl/2</strong>: pull -i 1</td>
<td></td>
</tr>
<tr>
<td>• <strong>wadl/1</strong>: synthesized record for master status. You can compare the journal positions of each replica with that of this server to see if any replica is falling behind.</td>
<td></td>
</tr>
<tr>
<td>Symbols are to be interpreted as follows:</td>
<td></td>
</tr>
<tr>
<td>• <strong>W/8</strong>: wait, long-poll request</td>
<td></td>
</tr>
<tr>
<td>• <strong>W</strong>: no wait</td>
<td></td>
</tr>
<tr>
<td>• <strong>A/4</strong>: Acknowledging</td>
<td></td>
</tr>
<tr>
<td>• <strong>a</strong>: non-acknowledging</td>
<td></td>
</tr>
<tr>
<td>• <strong>D/2</strong>: durable</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 5. Monitoring and managing server cluster

<table>
<thead>
<tr>
<th>d</th>
<th>non-durable</th>
</tr>
</thead>
<tbody>
<tr>
<td>L/1</td>
<td>data about the local journal; that is, the journal of the server that is the target of the <code>p4 servers</code> command.</td>
</tr>
<tr>
<td>1</td>
<td>request from a replica (shows progress in copying master's journal)</td>
</tr>
</tbody>
</table>

IsAlive 1 if the server is up; 0 if it is down.

Pull or journal-copy requests are recorded in the `db.jnlack` table only when made from a replica that has either a server ID or a `P4NAME`. Any replica that makes such a request but does not have a server ID or `P4NAME` is not recorded in the table.

**Monitoring Zookeeper**

The status of Zookeeper nodes is returned as part of the `p4cmgr status` command. Or, you can use `telnet` or `netcat` to send the command `stat` to port `2380` of a Zookeeper node. For example, for a Zookeeper service on a node called `myznode`:

```bash
echo stat | nc myznode 2380
```

This returns a status page that reports whether Zookeeper is serving requests.

**Monitoring events**

You can monitor events that affect the cluster using the Python script `zkMonitor`. To invoke the script, use a command like the following. Currently `zkMonitor` must be run with `sudo`.

```bash
sudo zkMonitor --cluster cluster_id --zkport zk_host_ports [--debug]
```

You must specify the id of the cluster of interest and the host and port values where Zookeeper modules have been deployed. For example:

```bash
zkMonitor --cluster myCluster --zkport "host1:2380,host2:2380,host3:2380" --debug
```

This command will monitor events in Zookeeper and print out information when the status of any state information stored in Zookeeper has changed. Specifying the `--debug` option causes a full dump of every event structure when `p4zk` callbacks occur.

If you press `Return`, the monitoring program allows you to choose one of three options: `q` to quit, `s` to display a cluster status summary, or any other value to resume monitoring mode. The cluster status summary displays a more concise view of the cluster than the initial view displayed by `zkMonitor`. 

---

Perforce Server Administrator’s Guide: Cluster Management 55
Managing a server cluster

The following sections describe some of the issues you need to handle in managing a server cluster.

**Improving performance**

The default setting of `lbr.autocompress` to 1 stores all text files as `ctext` (the file type remains text). This improves performance in a server cluster for which a shared archive has been configured.

**Managing failovers**

The timeout Zookeeper configurables are used to control how Zookeeper evaluates the need to generate a failover. These are described in “Zookeeper configurables” on page 38. In setting timeouts, you want to weigh the interests of failover responsiveness versus the danger of false failovers. You can set values in the Zookeeper configuration files to address these issues. For information, please contact Perforce support.
Using p4cmgr commands

This chapter describes the p4cmgr commands you use to set up, monitor, and manage a server cluster. In addition, it explains the authorization you need to run different commands, and it provides some additional guidelines for using these commands.

Getting authorization to run p4cmgr commands

This section explains the authorizations you need to run p4cmgr commands.

To run p4cmgr init, you must:

• Be a member of the OS group perforce (created at install time).

• Have sudo (NOPASSWD) permission to run /bin/sh on the P4CMGR host and on the depot master.

• Be able to ssh to the target depot master as this OS user (the OS user must exist on all hosts in the cluster).

To run p4cmgr add, you must:

• For the first add on a given host, you must be able to ssh to the target host and have sudo (NOPASSWD) permission to run /bin/sh on that host.

• Have a Perforce super user name and a password for that user unless an existing ticket grants the permission required.

• Be a member of the OS group perforce.

To run any of the other p4cmgr commands, you must:

• Be a member of the OS group perforce.

• For the p4cmgr status command you may need a Perforce super user name and a password for that user, unless an existing ticket grants the permission required.

Restrictions on using p4cmgr and p4 commands

A Perforce server cluster needs to be situated behind a firewall; only the administrator may access servers other than the router directly. Users may call p4 commands; however, the following commands and command options are automatically blocked when issued from outside the firewall:

• You may not use the p4 configure command to change any cluster properties.

• You may not use the checkpoint, journal, stop, restart, dump, or import arguments to the p4 admin command.

• You may not use the p4 cluster command.
• You may not use the `p4 cache purge` command.

• You may not use the `p4 protect` command.

Inside the firewall, the administrator can issue `p4cmgr` commands as well as any `p4` commands directly to any server, but should exercise great discretion when attempting to use the commands and command options described above.

### Getting help for p4cmgr commands

Each `p4cmgr` command has an `-h` option that displays help for that command. For example, the following command displays help for the `p4cmgr init` command.

```
p4cmgr init -h
```

### General guidelines

Follow these guidelines in using `p4` and `p4cmgr` commands.

#### Specifying server id's:

Server id's are used to identify the members of a cluster: they identify an instance of a service. A server id must be unique for a cluster.

We recommend that you include the server role in the serverid. For example, `WkSpc1`, `WkSpc2` and `Router1`, `Router2`. But do not specify the role for the depot master and depot standby in the server id because after failover, these roles will change. It is best to name the depot master and depot standby with names like `Depot1` and `Depot2`.

#### Specifying host names

The simple or fully-qualified name format you use when you first specify a host name (with the `p4cmgr init` or `p4cmgr add` commands) must be the format you use any time you refer to that same host. If you use different commands to refer to different hosts, you may use different formats; that is, you can use a simple name for `host1` and a fully-qualified domain name for `host2`. Here are the different formats:

- Fully qualified domain name: `myhost.perforce.com`
- Simple name: `myhost`
- IP address: `10.0.0.123`

You may not use an IP address for a host name.

#### Using environment variables

The following environment variables are automatically set for server cluster nodes:

- `P4ROOT`
• P4JOURNAL
• P4PORT
• P4LOG
• PATH

**Triggers**

The `p4 submit` command on a workspace server will run the `edge-submit` and `edge-content` triggers, if defined, prior to transferring the changelist to the depot master for final submission. The `p4 submit` command on the depot master will run the standard trigger types; however, if the change was submitted from a workspace server, the `p4 change` and `p4 describe` commands must not be used in the trigger. The `%serverid%` variable should be used in triggers that need to detect which server is running the trigger.
p4cmgr

Displays a help summary for p4cmgr commands and displays version information.

**Syntax**

p4cmgr [-h | --help][-V | --version ]

**Usage Notes**

<table>
<thead>
<tr>
<th>Can File Arguments Use Revision Specifier?</th>
<th>Can File Arguments Use Revision Range?</th>
<th>Minimal Access Level Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>NA</td>
<td>super</td>
</tr>
</tbody>
</table>

**Examples**

p4cmgr -h  

Displays a summary of all p4cmgr commands.
**p4cmgr add**

Add a node to a cluster of the given type.

**Syntax**

```
p4cmgr add -h | --help
p4cmgr add depot -s server_id [-l license_path] [host_name]
p4cmgr add workspace -s server_id [-l license_path] [host_name]
p4cmgr add zookeeper -s server_id [--force] [--no-restart] [host_name]
p4cmgr add router -s server_id [--noss] [--port port] [host_name]
```

**Description**

Use `p4cmgr add` to add a node of the given type to a cluster. You can add only nodes of type **depot**, **workspace**, **router**, or **zookeeper**; only one depot standby can be active at once.

A node may be added only during a maintenance interval; at this time, the router should refuse to accept all external requests, but cluster services can still be running. (The router is not automatically configured to refuse requests; the administrator must explicitly configure it to do so.) Adding a node creates a `server.id` file in the new server’s root directory.

If you try to add a Zookeeper service when a cluster server is down, the `p4cmgr add zookeeper` command will fail. You must remove or restart the problematic server, add the Zookeeper service, and then restart the cluster. There are some cases where you might need to force the Zookeeper service to be added: for example, you want to add a Zookeeper service when a depot master has failed over and it hasn’t yet been removed from the cluster. In this case, use the `--force` option to execute the `p4cmgr add zookeeper` command.

To use this command you must meet the following requirements:

- For the first add for a given host, you must be able to ssh to the target host and you must have `sudo (NOPASSWD)` permission on that host to run `/bin/sh`.

- Have a Perforce super user username and a password for that user unless an existing ticket grants the required permission.

- Be a member of the OS group `perforce`.

**Important**

You must not place more than one Zookeeper service on the same node. It is better to place Zookeeper services on workspace and router nodes, rather than on depot master or depot standby nodes, because Zookeeper controls failover of depot masters to depot standbys. It is best to place Zookeeper services on their own nodes.
Options

- `license_path` The full path to a license file for the depot standby. The license file must be situated on the P4CMGR host. `p4cmgr` distributes the license file to the target. Workspace servers do not need license files, but you can specify one if needed for historic reasons.

You can also use this syntax: `--license-path path`

- `h` Display help for the command, then exit.

You can also use this syntax: `--help`.

- `server_id` The server id of the server you want to add.

You can also use this syntax: `--service-name server_id`

- `force` Forces a command without issuing confirmation.

- `no-restart` Prevents a cluster from restarting. Set to `false` by default. This is useful when adding multiple Zookeeper services to prevent restarting until you add the last one.

- `noss` Configure the added router in non-SSL mode. Set to `false` by default.

- `port number` Configure the added router to be started on the specified port. An error occurs if the port is in use. You can either specify a different port when you run the `p4cmgr add` command again, or you can specify no port when you run the command and have the system choose the next available port.

- `host_name` The name of the host where you want to run the service.

The name format you use (simple or fully-qualified) must be used in all subsequent references to this host. You may not use an IP address for a host name.

Usage Notes

<table>
<thead>
<tr>
<th>Can File Arguments Use RevisionSpecifier?</th>
<th>Can File Arguments Use Revision Range?</th>
<th>Minimal Access Level Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>NA</td>
<td>super</td>
</tr>
</tbody>
</table>

Examples

```
p4cmgr add -s myDepot depot myHost.com
```

Adds the depot server `myDepot` on `myHost.com`. 
### Related Commands

<table>
<thead>
<tr>
<th>To start the service you have added</th>
<th><code>p4cmgr start</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>To get status to make sure the service has been added</td>
<td><code>p4cmgr status</code></td>
</tr>
</tbody>
</table>

### p4cmgr backup

Checkpoints the current server cluster. There must be a depot standby running to use this command.

#### Syntax

`p4cmgr backup [-h | --help]`

#### Description

Use `p4cmgr backup` to backup the cluster. This is done by taking a checkpoint of the depot standby and rotating the journal on the depot master.

The checkpoint is placed in `/p4/archives/master_checkpoint.gz` and is available on any node that has mounted the shared file system.

A weekly automatic backup is done at 2 a.m. every Sunday, which simply runs the `p4cmgr backup`. There is currently no mechanism in place to inform you if the automatic backup fails. You must set up your own email notifications or other monitoring or these jobs. For more information about automatic backups, see “Scheduled backups” on page 49.

#### Options

- `-h` Display help for the command, then exit.

You can also use this syntax: `--help`.

#### Usage Notes

<table>
<thead>
<tr>
<th>Can File Arguments Use RevisionSpecifier?</th>
<th>Can File Arguments Use Revision Range?</th>
<th>Minimal Access Level Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>NA</td>
<td>super</td>
</tr>
</tbody>
</table>

#### Examples

```
p4cmgr backup
```

Checkpoints the current cluster.
p4cmgr init

Initializes a cluster by setting up a depot master on the target machine and setting cluster-related configurables on the installed depot master. For information on the configurables set, see Chapter 3, “Configuring and Reconfiguring a Server Cluster” on page 31.

Syntax

p4cmgr init -h | --help
p4cmgr init [--case-insensitive] [-l path] [-s server_id] cluster_name host_name

Description

Use p4cmgr init to set up and configure a cluster and its depot master.

This command also starts a Zookeeper service on the p4cmgr node, that is the local node where you are executing the p4cmgr init command.

After you initialize the cluster, you can use the p4cmgr add command to add other nodes to the cluster.

Options

--case-insensitive Enables case insensitivity, which might be useful if the cluster is accessed by Windows clients.

-l path The full path to a license file for the depot master. The license file must be situated on the P4CMGR host. The license files are distributed to their target hosts by p4cmgr.

You only need to specify a license if you have more than twenty users and 1,000 files. If you don’t specify a license and content grows to require one, it needs to be installed manually.

You can also use this syntax: --license-path path.

-s server_id The name for the new depot master. By default, this is set to depot-server. You should change the default value to a name that does not include a role designation. This will avoid confusion if the depot standby is promoted to depot master. The name must be unique across the cluster.

The server id of the Zookeeper that is automatically installed on the P4CMGR host is zookeeper.

You can also use this syntax: --service-name server_id

cluster_name The name for the cluster you want to initialize.
host_name

The name of the host for the depot master. Do not specify an IP address; but you can specify a fully-qualified domain name.

This command also creates a specification for the depot master.

**Usage Notes**

<table>
<thead>
<tr>
<th>Can File Arguments Use Revision Specifier?</th>
<th>Can File Arguments Use Revision Range?</th>
<th>Minimal Access Level Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>NA</td>
<td>super</td>
</tr>
</tbody>
</table>

**Examples**

```bash
p4cmgr init -l /path/to/myMasterLicense -s myDepot1 myCluster myHost.com
```

Initialize the cluster myCluster and set up a depot master myDepot1 on the host myHost.com, using the license file /path/to/myMasterLicense.

**Related Commands**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>To add more nodes to the cluster</td>
<td>p4cmgr add</td>
</tr>
<tr>
<td>To check that the cluster has been initialized</td>
<td>p4cmgr status</td>
</tr>
</tbody>
</table>
p4cmgr remove

Removes one or more services from a cluster.

**Syntax**

```
p4cmgr remove [-h | --help]
p4cmgr remove depot -s server_id [--force] [hostname]
p4cmgr remove workspace -s server_id [--force] [hostname]
p4cmgr remove zookeeper -s server_id [--force] [hostname]
p4cmgr remove router -s server_id [--force] [hostname]
```

**Description**

The `p4cmgr remove` command removes the specified services whether they are stopped or not. You are prompted to confirm the removal unless you force the operation. The depot master must be the last service to be removed. Normally, this command is used to remove services that have failed to be added to a cluster.

Use the `p4cmgr status` command to check that the specified services have been removed.

There are two cases that require explicit removal:

- A `p4cmgr init` or `p4cmgr add` command has failed to complete. You need to remove the service that was not completely added before you retry the command. You must use the `--force` option in this case.

- A failed Zookeeper service that is replaced by another service.

   It is best to explicitly remove the defunct Zookeeper service. This is because even a failed service continues to be counted in the automatic polling done in choosing a new depot master. You must restart the cluster after removing the failed Zookeeper service.

If a depot master has failed and the system has failed over to a depot standby, it is safe to remove the old depot master. Removing the depot master at any time other than following a failover will cause the cluster to break completely and have to be started again from scratch.

If you remove a workspace server, you will probably delete user workspace data, which isn't automatically backed up or replicated anywhere else, so this not safe to do on a cluster that is in use.

When removing a Zookeeper service, make sure that all current cluster nodes are up and running. This is because every service has to hold a reference to the host-port pairs for all configured Zookeeper services. If a Zookeeper service is removed when a service is down, the Zookeeper configuration for the downed service will not be updated, which will cause problems.
Options

- `-s server_id`  The server id of the `p4d` instance you want to remove.

  You can also use `--service-name` to specify this option.

<table>
<thead>
<tr>
<th><code>host_name</code></th>
<th>The name of the host where the specified service resides.</th>
</tr>
</thead>
</table>

| `--force` | Remove the specified service without prompting for confirmation. |

  This flag is required in any case where a partially set up service needs to be removed. It ignores all errors returned by the `p4cmgr remove` command, allowing it to ignore errors in removing things that were not properly setup in the first place.

Usage Notes

<table>
<thead>
<tr>
<th>Can File Arguments Use Revision Specifier?</th>
<th>Can File Arguments Use Revision Range?</th>
<th>Minimal Access Level Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>NA</td>
<td>super</td>
</tr>
</tbody>
</table>

Examples

```
p4cmgr remove -s myDepot2 myHost.com  
Removes the service `myDepot2` on `myHost.com`.
```

```
p4cmgr remove -s zookeeper1 zookeeper  
Removes the Zookeeper service from its host.
```

Related Commands

To check that a service has been removed  
`p4cmgr status`
**p4cmgr restart**

Restarts one or more p4d instances in a cluster. Use this command to restart one or more
cluster nodes after stopping the nodes.

This command basically combines the `p4cmgr stop` and `p4cmgr start` commands.

**Syntax**

```
p4cmgr restart [-h | --help]
p4cmgr restart [-s server_id] [--force] [host_name]
```

**Description**

Use `p4cmgr restart` to do one of the following:

- Restart the p4d instance specified by the `server_id` parameter.
- Restart all of the p4d instances configured on the specified host (by just passing the
`host_name` parameter).
- Restart all the p4d instances on the entire cluster (by omitting all parameters).

Restarting all instances on the cluster restarts the depot master, depot standby, and all
workspace servers and Zookeeper services. It also restarts the routers and verifies that all
nodes and the routers are available.

You must restart the cluster after adding Zookeeper nodes because Perforce services are not
aware of the new Zookeeper configuration until the services restart. The Zookeeper services
must also be restarted to find out about one another. The `p4cmgr restart` command also
restarts the Zookeeper services in the correct order.

The `p4cmgr restart` command does not verify that services are actually started. Use the `p4cmgr
status` command to determine what services are running.

Use the `--force` option to force a command without prompting for confirmation.

**Options**

- `-s server_id` The name of the p4d instance you want to restart.
  You can also use `--service-name` to specify this option.

| `host_name` | The name of the host that you want to restart. |
Usage Notes

<table>
<thead>
<tr>
<th>Can File Arguments Use Revision Specifier?</th>
<th>Can File Arguments Use Revision Range?</th>
<th>Minimal Access Level Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>NA</td>
<td>super</td>
</tr>
</tbody>
</table>

Examples

`p4cmgr restart myHost.com`  
Restarts all p4d instances on myHost.com.

Related Commands

To stop a service in the cluster  
`p4cmgr stop`
p4cmgr ssh-reset

Regenerates SSH certificates for the specified host. Use this command only with guidance from Perforce support.

**Syntax**

```
p4cmgr ssh-reset [-h | --help] [--force] hostname
```

**Description**

Use `p4cmgr ssh-reset` to clear the keys for the server(s) on the specified host.

**Options**

- `-h` Display help for the command, then exit.
- `--force` Force command execution without prompting for confirmation and continuing when errors occur.
- `hostname` The name of the managed host for which certificates are cleared.

**Usage Notes**

<table>
<thead>
<tr>
<th>Can File Arguments Use Revision Specifier?</th>
<th>Can File Arguments Use Revision Range?</th>
<th>Minimal Access Level Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>NA</td>
<td>super</td>
</tr>
</tbody>
</table>

**Examples**

```
p4cmgr ssh-reset myHost
```

Resets SSH certificates on `myHost`.
**p4cmgr start**

Starts one or more p4d instances in a cluster.

**Syntax**

```
p4cmgr start -s server_id [host_name]
p4cmgr start -s server_id
p4cmgr start host_name
p4cmgr start
```

**Description**

Use `p4cmgr start` to do one of the following:

- Start the p4d instance specified by the `server_id` parameter.
- Start all the p4d instances configured on the specified host.
- Start all the p4d instances on the entire cluster.

Starting all instances on the cluster starts the depot master, depot standby, all workspace servers, and Zookeeper services. It also starts the routers and verifies that all nodes and the routers are available.

When you use the `p4cmgr init` or `p4cmgr add` commands, the system automatically starts the depot master or the server that was added. You only need to use the `p4cmgr start` command after you have explicitly stopped a server or after the server has lost power or failed for some other reason.

No validation is done to make sure all the specified services have started. You must use the `p4cmgr status` command to determine which services are running in your cluster.

**Options**

- `-s server_id` The server id of the p4d or Zookeeper instance you want to start.
- `host_name` The name of the host where you want to specified service to run.

**Usage Notes**

<table>
<thead>
<tr>
<th>Can File Arguments Use Revision Specifier?</th>
<th>Can File Arguments Use Revision Range?</th>
<th>Minimal Access Level Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>NA</td>
<td>super</td>
</tr>
</tbody>
</table>
Examples

```
p4cmgr start myHost.com
```
Starts all `p4d` instances on `myHost.com`.

Related Commands

To stop a service in the cluster

```
p4cmgr stop
```
**p4cmgr status**

Reports on the health of the specified host.

**Syntax**

```
p4cmgr status [host_name]
```

**Description**

See “Getting status information” on page 51 for a discussion of the information that is displayed with the `p4cmgr status` command.

**Options**

`host_name`  The name of the host where the specified service is running. If the host name is not specified, information is returned about all services in the cluster.

**Usage Notes**

<table>
<thead>
<tr>
<th>Can File Arguments Use RevisionSpecifier?</th>
<th>Can File Arguments Use RevisionRange?</th>
<th>Minimal Access Level Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>NA</td>
<td>super</td>
</tr>
</tbody>
</table>

**Examples**

```
p4cmgr status myHost.com
```

Display information about the node `myHost.com`. 
**p4cmgr stop**

Stop one or more p4d instances in a cluster. This command stops all the Zookeeper services in the cluster as well.

**Syntax**

```
p4cmgr stop [-h | --help]
p4cmgr stop -s server_id [host_name]
p4cmgr stop hostname
p4cmgr stop
```

**Description**

Use `p4cmgr stop` to do one of the following:

- Stop the specified p4d instance specified by the `service` parameter.
- Stop all the p4d instances on the specified host.
- Stop all the p4d instances on the entire cluster.

Stopping all the servers in a cluster includes the depot master, depot standby, workspace servers, and routers. The command does not validate that all instances have stopped.

This command ensures that all servers are stopped in the correct order: routers are stopped first, then workspace servers, then the depot standby, the the depot master, and finally the Zookeeper services.

**Options**

- `-s server_id` The server id of the p4d instance you want to stop.
- `host_name` The name of the host where the service or services are running.

**Usage Notes**

<table>
<thead>
<tr>
<th>Can File Arguments Use RevisionSpecifier?</th>
<th>Can File Arguments Use Revision Range?</th>
<th>Minimal Access Level Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>NA</td>
<td>super</td>
</tr>
</tbody>
</table>

You can use the `p4cmgr start` command to start one or more p4d instances that you have stopped.

**Examples**

```
p4cmgr stop myHost.com
```

Stops all p4d instances on `myHost.com`. 

Related Commands

To start one or more p4d instances

```
p4cmgr start
```
Appendix

License Statements

Perforce software includes software developed by the University of California, Berkeley and its contributors. This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/).
Index

A
authentication
   administrator, of, 12
   guidelines for, 27
   perforce group, 12, 28
   sudo privileges, 13

B
backup
   automatic, 49, 65
   journal management, and, 43
   workspace servers, 44
binary files, 35

C
case-sensitivity, 67
checkpoint
   creating, 23
   file, 65
   p4cmgr backup command, 65
clients, 40
   creating, 25
   defined, 2
cluster
  access to, 2
  architecture of, 1
  backing up, 65
  case sensitivity, 67
  changing topology of, 39
  checkpointing, 65
  components of, 9
  configurables (see configurables)
  deployed, picture of, 9
  deployment (see deployment)
  distributed environment, in, 25
  ease of use (see P4CMGR)
  environment variables for, 58
  environment, requirements for, 11
  environment, set up of, 12
  extending, 2
  firewall, 12
  hardware requirements for, 6
  initializing, 19, 67
  license file for, 25, 64, 67
  log files (see log files)
  migrating to, 28
  monitoring (see monitoring)
  moving clients in, 40
  name, 67

network configuration for, 14
network topology of, 10
operating system support, 12
p4zk processes in, 4
patches to, 28
platform requirements for, 6
removing nodes from, 39
removing services from, 69
restarting, 69, 71
scope, 33
security (see security)
setting up, 18
shared file system (see shared file system)
SSL connections, use of, 27, 64
standby server, 22
starting servers in, 75
status, 20, 51, 77
stopping services in, 79
virtual machines in (see virtual machines)
cluster architecture and commit-edge, 2
cluster router
   adding, 23
   cache, stale, 40
   configuration file, 23, 24, 27
   defined, 2
   failure of, 42
   recovering, 47
   removing, 69
   SSL connections, use of, 28
cluster.id configurable, 34
configurables
   current setting of, 32
   displayed by p4cmgr status command, 20
   guidelines for using, 34
   scope of, 33
   summary of, 31

D
db.replication configurable, 34
db.routing file, 40
deployment
   cluster nodes, 9
network topology, 10
operating systems, 12
preparing for, 11
workflow, 11
depot master
backing up, 43
backup error, 49
defined, 1
journal file, 36
P4PORT of, 2
port of, 36
recovering, 46
removing, 69
replication of records, 52
scheduled backups of, 49
depot standby
defined, 1
recovering, 46
disaster recovery, 42
E
et edge server, 26, 27
environment variables, 58
F
failover, 46
(see also recovery)
automatic, 41
enabled by Zookeeper, 4
failure handling, 41
net.keepalive.count, 35
net.keepalive.idle, 35
net.keepalive.interval, 35
rpl.journal.ack configurable, 36
rpl.journal.min configurable, 37
summary of, 4
unneeded, 56
file compression, 35
firewall, 16
cluster, in, 12
H
high availability, 4
(see also recovery)
host names, 58
I
installation
preparing for, 11
troubleshooting, 28
J
journal file
depot master, for, 36
file prefix, 34
use of in recovery, 43
journalPrefix configurable, 34
L
lbr.autocompress configurable, 35, 56
lbr.replication configurable, 34
license file, 25, 64
log files
format of, 24
setting location for, 24
M
master_checkpoint.gz file, 65
monitor configurable, 35
monitoring
events, 55
setting level of, 35
status information, 51
Zookeeper, 55
N
net.keepalive.count configurable, 35
net.keepalive.idle configurable, 35
net.keepalive.interval configurable, 35
network failure, 42
P
p4 admin command, 57
p4 cachepurge command, 58
p4 cluster command, 57
p4 configure command, 57
p4 journalcopy command, 41, 52
p4 protect command, 58
p4 pull command, 41, 44, 52
p4 servers command, 43, 44, 52
p4.utils.dir configurable, 35
P4CMGR
command interface to, 5
defined, 5
installing, 16
p4cmgr add command, 22, 63
p4cmgr backup command, 65
p4cmgr command, 61
p4cmgr commands
authorization to run, 57
help for, 58
  host names, specifying in, 58
  restrictions on using, 57
  server id’s, specifying in, 58
p4cmgr commands vs p4 commands, 5
p4cmgr init command, 19, 67
p4cmgr remove command, 69
p4cmgr restart command, 22, 71
p4cmgr ssh-reset command, 73
p4cmgr start command, 75
p4cmgr status command, 51, 77
p4cmgr stop command, 79
P4JOURNAL environment variable, 36
P4LOG environment variable, 25
P4TARGET environment variable, 26, 34, 36
p4zk processes, 4
p4zk Unix domain socket, 36
p4zk.log.file configurable, 36
p4zk.socket.path configurable, 36
perforce group, 28
performance
  improving, 56
  network topology, and, 10
proxy server, 25

R
recovery
  cluster router, 47
  depot master, 46
  depot standby, 46
  workspace server, 47
router
  adding, 63
  port, 64
rpl.counter.hook configurable, 36
rpl.journal.ack configurable, 36, 43
rpl.journal.min configurable, 37, 43

S
SaltStack, 3, 5
  (see also P4CMGR)
security
  guidelines for, 27
  service user, defining, 37
  SSH trust, enabling, 16
server cluster (see cluster)
server.depot.root configurable, 38
serviceUser configurable, 37
shared file system
  client configuration options, 15
defined, 2
  defining location of, 38
failure of, 42
use of, 12
shared filesystem
  mounting, 15
SSH certificates, 73

T
triggers, 59

U
unicode mode, 29
utility files, 35

V
version information, 61
virtual machines
  guidelines for use, 12
  high availability, and, 12

W
workspace server
  adding, 22
  backing up, 44
  defined, 2
  failure of, 5
  problems with, 42
  recovering, 47
  removing, 69
  specifying for new users, 24
  troubleshooting recovery of, 48
  updating problems, 48

Z
zk.host.port.pairs configurable, 22, 38
zkMonitor script, 55
Zookeeper
  adding a service, 63
  adding nodes, 21
  configurables, 38
  defined, 3
  failover, and, 41
  heartbeat messages, 38
  location of servers, 38
  log file for, 25
  log file, configurable for, 36
  monitoring, 55
  p4zk processes, and, 4
  removing, 69
starting, 67