Server Deployment Package (SDP) for Perforce Helix

SDP Failover Guide (for Unix and Windows)

Perforce Professional Services

Version v2019.3, 2020-08-05

Table of Contents

Preface

The Server Deployment Package (SDP) is the implementation of Perforce's recommendations for operating and managing a production Perforce Helix Core Version Control System. It is intended to provide the Helix Core administration team with tools to help:

- High Availability (HA)
- Disaster Recovery (DR)

This guide is intended to provide instructions for failover in an SDP environment using built in Helix Core features.

For more details see:

- Sysadmin Guide - Failover

Please Give Us Feedback

Perforce welcomes feedback from our users. Please send any suggestions for improving this document or the SDP to consulting@perforce.com.

Chapter 1. Overview

We need to consider planned vs unplanned failover. Planned may be due to upgrading the core Operating System or some other dependency in your infrastructure, or a similar activity.

Unplanned covers risks you are seeking to mitigate with failover:

- loss of a machine, or some machine related hardware failure
- loss of a VM cluster
- failure of storage
- loss of a data center or machine room
- etc...

See also p4 failover in Command Reference Guide

1.1. Planning

HA failover should not require a P4PORT change for end users. Depending on your topology, you can avoid changing P4PORT by having users set P4PORT to an alias, e.g. `perforce.p4demo.com` (or just `perforce`), or `perforce_syd.p4demo.com` (or just `perforce_syd`). During failover, that would be targeted by doing something like:

- Changing a DNS alias so `perforce.p4demo.com` points to the backup machine.
- Changing the Perforce broker configuration to target the backup machine.
- Changing the Virtual IP configuration.
- Changing the anycast routing configuration.

Chapter 2. Planned Failover

In this instance you can run `p4 failover` with the active participation of its upstream server.

We are going to provide examples with the following assumptions:

- serverid `master_1` is current commit, running on machine `bos-helix-01`
- serverid `p4d_ha_bos` is HA server
- DNS alias `perforce` is set to `bos-helix-01`

2.1. Prerequisites

You need to ensure:

1. you are running p4d 2018.2 or later for your commit and all replica instances, preferably 2020.1

   ```
   source /p4/common/bin/p4_vars 1
   p4 info | grep version
   ```

2. your failover target server instance is of type `standby` or `forwarding-standby`

   On HA machine:

   ```
   p4 info
   :
   ServerID: p4d_ha_bos
   Server services: standby
   Replica of: perforce:1999
   :
   ```

3. it has Options mandatory set in its server spec

   ```
   p4 server -o p4d_ha_bos | grep Options
   Options:  mandatory
   ```

4. you have a valid `license` installed in `/p4/1/root` (<instance> root)

   On HA machine:

   ```
   cat /p4/1/license
   ```

5. Monitoring is enabled - so the following works:

```
p4 monitor show -al
```

6. DNS changes are possible so that downstream replicas can seamlessly connect to HA server
7. Current `pull` status is valid

```
p4 pull -lj
```

2.2. Failing over

The actions are:

1. Run `p4 failover` in reporting mode on HA machine:

```
p4 failover
```

Successful output looks like:

```
Checking if failover might be possible ...
Checking for archive file content not transferred ...
Verifying content of recently update archive files ...
After addressing any reported issues that might prevent failover, use --yes or -y
to execute the failover.
```

2. Perform failover:

```
p4 failover --yes
```

Output should be something like:

```
Starting failover process ...
Refusing new commands on server from which failover is occurring ...
Giving commands already running time to complete ...
Stalling commands on server from which failover is occurring ...
Waiting for 'journalcopy' to complete its work ...
Waiting for 'pull -L' to complete its work ...
Waiting for 'pull -u' to complete its work ...
Checking for archive file content not transferred ...
Verifying content of recently updated archive files ...
Stopping server from which failover is occurring ...
Moving latest journalcopy'd journal into place as the active journal ...
Updating configuration of the failed-over server ...
Restarting this server ...
```


During this time, if you run commands against the master, you may see:

```
Server currently in failover mode, try again after failover has completed
```

3. Change the DNS entries so downstream replicas (and users) will connect to the new master (that was previously HA)
4. Validate that your downstream replicas are communicating with your new master

   On each replica machine:

   ```
   p4 pull -lj
   ```

   Or against the new master:

   ```
   p4 servers -J
   ```

   Check output of `p4 info`:

   ```
   :
   Server address: box-helix-02
   :
   ServerID: master_1
   Server services: commit-server
   :
   ```

5. Make sure the old server spec (`p4d_ha_bos`) has correctly had its `Options:` field set to `nomandatory` (otherwise all replication would stop!)

Chapter 3. Unplanned Failover

In this case there is no active participation of upstream server, so there is an increase risk of lost data.

We assume we are still failing over to the HA machine, so:

- Failover target is `standby` or `forwarding-standby`
- Server spec still has `Options:` set to `mandatory`
- Original master is not running

The output of `p4 failover` on the DR machine might be:

```
Checking if failover might be possible ...
Server ID must be specified in the '-s' or --serverid' argument for a failover without
the participation of the server from which failover is occurring.
Checking for archive file content not transferred ...
Verifying content of recently update archive files ...
After addressing any reported issues that might prevent failover, use --yes or -y to
execute the failover.
```

1. Execute `p4 failover` with the extra parameter to specify server we are failing over from:

   ```
   p4 failover --serverid master_1 --yes
   ```

Expected output is somewhat shorter than for planned failover:

```
Starting failover process ...
Waiting for 'pull -L' to complete its work ...
Checking for archive file content not transferred ...
Verifying content of recently updated archive files ...
Moving latest journalcopy'd journal into place as the active journal ...
Updating configuration of the failed-over server ...
Restarting this server ...
```

Chapter 4. Old style failover

This does not use the `p4 failover` command (so is valid for pre-2018.2 p4d versions)

See: KB - Failing over to a Replica