Server Deployment Package (SDP) for Perforce Helix

SDP Failover Guide (for Unix and Windows)

Perforce Professional Services

Version v2024.2, 2024-12-20

Table of Contents

Preface . . . 1
1. Overview . . . 2
1.1. Planning . . . 2
2. Planned Failover . . . 3
2.1. Prerequisites . . . 3
2.1.1. Pre Failover Checklist . . . 4
2.1.1.1. Create end user/admin test script . . . 4
2.1.1.2. Confirm Trust Established with Replicas . . . 4
2.1.1.3. Checks for edge server failover to its HA . . . 5
2.1.1.4. Swarm Triggers or Extensions . . . 5
2.1.1.5. Other Triggers . . . 6
2.1.1.6. Other Replica's P4TARGET . . . 6
2.1.1.7. Proxies . . . 6
2.1.1.8. HA Server OS Configuration . . . 7
2.2. Failing over . . . 7
2.3. Post Failover . . . 8
2.3.1. Validation of SDP Config . . . 8
2.3.2. Validating the server - user testing . . . 9
2.3.3. Moving of Checkpoints . . . 9
2.3.4. Check daily_checkpoint.sh runs successfully . . . 10
2.3.5. Check on Replication . . . 10
2.4. Failing over on Windows . . . 11
2.4.1. Post Failover on Windows . . . 12
2.5. Preparation for Failing Back - Setup original commit server as new Failover replica . . . 12
3. Unplanned Failover . . . 14
3.1. Post Unplanned Failover . . . 14
3.1.1. Resetting Downstream Replicas . . . 14
3.2. Unplanned Failover on Windows . . . 15

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.

Please Give Us Feedback

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

© 2007-2024 Perforce Software, Inc.

Chapter 1. Overview

Relevant Docs:

- SDP Guide for Unix - Planning for HA and DR
- SDP Guide for Unix: Pre-requisites for Failover
- SDP Guide for Windows
- Sysadmin Guide - Failover

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 their P4PORT to a DNS name that is centrally managed.

For example:

- `bos-helix-01`, a master/commit-server in Boston, pointed to by a DNS name like `perforce` or `perforce.p4demo.com`.
- `bos-helix-02`, a standby replica in Boston, not pointed to by a DNS until failover. In event of failover, it gets pointed to by `perforce`/`perforce.p4demo.com` via a DNS change.

See SDP Guide: Server Host Naming Conventions

Other advanced networking options might be possible if you talk to your local networking gurus (virtual IPs etc).

© 2007-2024 Perforce Software, Inc.

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 `p4d-bos-01`
- ServerID `p4d_ha_bos` is HA server
- DNS alias `perforce` is set to `p4d-bos-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/root/license
   ```

5. Monitoring is enabled - so the following works:

© 2007-2024 Perforce Software, Inc.

```
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
```

8. You have a valid `offline_db` for the HA instance

   Check that the sizes of the `db.*` are similar - compare output:

```
ls -lhSr /p4/1/offline_db/db.* | tail
ls -lhSr /p4/1/root/db.* | tail
```

   Check the current journal counter and compare against live journal counter:

```
/p4/1/bin/p4d_1 -r /p4/1/offline_db -jd - db.counters | grep journal
p4 counters | grep journal
```

9. Check all defined triggers will work (see next section) - including Swarm triggers
10. Check authentication will work (e.g. LDAP configuration)
11. Check firewall for HA host - ensure that other replicas will be able to connect on the appropriate port to the instance (using the DNS alias)

2.1.1. Pre Failover Checklist

It is important to perform these checks before any planned failover, and also to make sure they are considered prior any unplanned failover.

2.1.1.1. Create end user/admin test script

See Section 2.3.2, “Validating the server - user testing”.

It is important to understand the key tests that will need to be run after the failover that will test all appropriate workflows for your Helix Core related processes!

Please note that you may need to ensure that appropriate teams and personnel are available to perform these tests as part of your failover planning.

2.1.1.2. Confirm Trust Established with Replicas

If using `ssl` on the commit server, then downstream replicas with have already set up `P4TRUST` entries with that server and its fingerprint.

After failover, you need to check that all replicas have appropriate `p4 trust` setup with the HA

© 2007-2024 Perforce Software, Inc.

server (which is now the commit server).

If you are using ***background submit*** between edge and commit servers, then you need to confirm the following:

- `p4 trust` is setup on the HA (new commit) with all edge servers
- the commit service user is correctly logged on to all edge servers (make sure the configurable `auth.id` is appropriately set to avoid IP-only tickets) on the HA server before failover

To check on background submit configuration:

```
p4 configure show allservers | grep bg
```

2.1.1.3. Checks for edge server failover to its HA

Similar to the above, if ***background submit*** is turned on then you need to confirm the following:

- `p4 trust` is setup on the commit with the **new** edge servers (which was edge HA)
- the commit service user is correctly logged on to the new edge server

2.1.1.4. Swarm Triggers or Extensions

If Swarm is not installed go to next section!

If it is installed and you are using extensions then all should be OK - since extensions are replicated and their environment is the same on a replica.

```
p4 extension --list --type extensions
```

The above should show `Perforce::helix-swarm` if swarm extension is in use - in which case go to next section!

Otherwise, check for triggers as below.

1. Swarm trigger is installed on HA machine (could be executed from a checked in depot file)

   Typically installed (via package) to `/opt/perforce/swarm-triggers/bin/swarm-trigger.pl`

   But can be installed anywhere on the filesystem

   Execute the trigger to ensure that any required Perl modules are installed:

   ```
   perl swarm-trigger.pl
   ```

   Note that things like `JSON.pm` can often be installed with:

© 2007-2024 Perforce Software, Inc.

```
sudo apt-get install libjson-perl
```

or

```
sudo yum install perl-JSON
```

2. Swarm trigger configuration file has been copied over from commit server to appropriate place.

   This defaults to one of:

```
/etc/perforce/swarm-trigger.conf
/opt/perforce/etc/swarm-trigger.conf
swarm-trigger.conf (in the same directory as trigger script)
```

2.1.1.5. Other Triggers

Checklist:

- Make sure the actual referenced script has been copied to the HA server prior to failover!
  - Note that executing triggers from the depot can make this easier - but following issue still applies
- Make sure that the appropriate version of `perl`, `python`, `ruby` etc are installed in locations as referenced by `p4 triggers` entries.
- Make sure that any relevant modules and dependencies have also been installed (e.g. `P4Python` or `P4Perl`)

2.1.1.6. Other Replica's P4TARGET

Review the settings for other replicas and also to check live replicas on the source server of the failover (`p4 servers -J`)

```
p4 configure show allservers | grep P4TARGET
```

Make sure the above settings are using correct DNS alias (which will be redirected).

2.1.1.7. Proxies

These are typically configured via `/p4/common/bin/p4_1.vars` settings:

```
export PROXY_TARGET=
```

Ensure the target is the correct DNS alias (which will be redirected).

 © 2007-2024 Perforce Software, Inc.

2.1.1.8. HA Server OS Configuration

Check to make sure any performance configuration such as turning off THP (transparent huge pages), and putting serverlocks.dir into a RAM filesystem have also been made to your HA Failover system. See SDP Guide: Maximizing Server Performance

2.2. Failing over

The basic actions are the same for Unix and Windows, but there are extra steps required as noted in Section 2.4, “Failing over on Windows”

1. Rotate the active journal on the master:

```
p4 admin journal
p4 servers -J       # Check all is up-to-date
```

2. 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.
```

3. 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 ...
```

© 2007-2024 Perforce Software, Inc.

```
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
```

4. Change the DNS entries so downstream replicas (and users) will connect to the new master (that was previously HA)
5. 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: p4d-bos-02
   :
   ServerID: master_1
   Server services: commit-server
   :
   ```

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

2.3. Post Failover

2.3.1. Validation of SDP Config

1. Run the script:

   ```
   /p4/common/bin/verify_sdp.sh 1
   ```

(or specify other instance ID).

Ensure that all output is valid and that there are no errors.

© 2007-2024 Perforce Software, Inc.

Ensure that the following returns 'yes':

```
/p4/common/bin/run_if_master.sh 1 echo yes
```

If it does not, then run: `bash -xv /p4/common/bin/run_if_master.sh 1 echo yes` and check the output. The most likely problems are things like the settings of the following variables in `/p4/common/config/p4_1.vars`:

```
P4MASTERHOST   <== Set to DNS name of new master host
P4MASTER_ID    <== Make sure this is same as value in /p4/1/root/server.id on the
master server machine.
```

2.3.2. Validating the server - user testing

This is a very important step to ensure that all relevant functions are working post failover.

This includes:

1. Can users connect
2. Can test submits work
    a. Are any triggers being tested properly and shown to be working in the new replica?
3. Is the Swarm integration working correctly?
4. Any other key process workflows shown to be working?
    a. This might include integrations with third party systems such as Jenkins or TeamCity.
5. Do logins work (e.g. whether via `ldap` or Helix Authentication)?

2.3.3. Moving of Checkpoints

After failing over, on Unix there will be journals which may need to be copied/moved and renamed due to the SDP structure.

For example, an HA server might have stored its journals in `/p4/1/checkpoints.ha_bos` (assuming it was create by `mkrep.sh` with serverid `p4d_ha_bos`):

```
/p4/1/checkpoints.ha_bos/p4_1.ha_bos.jnl.123
/p4/1/checkpoints.ha_bos/p4_1.ha_bos.jnl.124
```

As a result of failover, these files need to be copied/moved to:

```
/p4/1/checkpoints/p4_1.jnl.123
/p4/1/checkpoints/p4_1.jnl.124
```

Easiest way is to install `prename` on your Linux. Then `p4rename -n 's/.ha_bos//'`

© 2007-2024 Perforce Software, Inc.

`p4_1.*` - and re-run without the `-n` if it looks good.

The reason different directories are sometimes used is because in some installations the `/hxdepots` filesystem is shared on NFS between commit server and HA server. When `/hxdepots` is shared, the `journalPrefix` must be set differently to avoid path conflicts on the shared storage.

If these files are not present, then normal SDP crontab tasks such as `daily_checkpoint.sh` will fail as they won't be able to find the required journals to be applied to the `offline_db`.

The following command will rotate journal and replay any missing ones to `offline_db` (it is both fairly quick and safe to run without placing much load on the server host as it doesn't do any checkpointing):

```
/p4/common/bin/rotate_journal.sh 1
```

If it fails, then check `/p4/1/logs/checkpoint.log` for details - it may have an error such as:

```
Replay journal /p4/1/checkpoints/p4_1.jnl.123 to offline db.
```

```
Perforce server error:
     open for read: /p4/1/checkpoints/p4_1.jnl.123: No such file or directory
```

This indicates missing journals which will need to be moved/copied as above.

2.3.4. Check daily_checkpoint.sh runs successfully

This is an important script for a master server, and it is good to make sure ASAP after failover that it works as expected.

```
nohup /p4/common/bin/daily_checkpoint.sh 1 &
```

And check output in: `/p4/1/logs/checkpoint.log`

This script can take a while on big repositories. It is OK to have it run in the crontab over night.

2.3.5. Check on Replication

We recommend that you connect to all your replicas/proxies/brokers and make sure that they are successfully working after failover.

It is surprisingly common to find forgotten configuration details meaning that they are attempting to connect to old server for example!

 © 2007-2024 Perforce Software, Inc.

For proxies and brokers - you probably just need to run:

```
p4 info
```

For downstream replicas of any type, we recommend logging on to the host and running:

```
p4 pull -lj
```

and checking for any errors.

We also recommend the following is executed on both HA server and all replicas and the output examined for any unexpected errors:

```
grep -A4 'Perforce server error:' /p4/1/logs/log
```

Or you can review the contents of `/p4/1/logs/errors.csv` if you have enabled structured logging.

2.4. Failing over on Windows

The basic steps are the same as for on Unix, but with some extra steps at the end.

After the `p4 failover --yes` command has completed its work (on the HA server machine):

1. Review the settings for the Windows service (examples are for instance `1`) - note below **-S** is uppercase:

   ```
   p4 set -S p4_1
   ```

   Example results:

   ```
   C:\p4\1>p4 set -S p4_1
   :
   P4JOURNAL=c:\p4\1\logs\journal (set -S)
   P4LOG=c:\p4\1\logs\p4d_ha_aws.log (set -S)
   P4NAME=p4d_ha_aws (set -S)
   P4PORT=1666 (set -S)
   P4ROOT=c:\p4\1\root (set -S)
   :
   ```

2. Change the value of `P4NAME` and `P4LOG` to correct value for `master`:

   ```
   p4 set -S p4_1 P4NAME=master
   p4 set -S p4_1 P4LOG=c:\p4\1\logs\master.log
   ```

© 2007-2024 Perforce Software, Inc.

And re-check the output of `p4 set -S p4_1`

3. Restart the service:

```
c:\p4\common\bin\svcinst stop -n p4_1
c:\p4\common\bin\svcinst start -n p4_1
```

4. Run `p4 configure show` to check that the output is as expected for the above values.

2.4.1. Post Failover on Windows

This is slightly different for the Windows and Linux SDP, since there is currently no equivalent of `mkrep.sh` for Windows, and replication topologies on Windows are typically smaller and simpler.

Thus it is Windows HA instances are likely to have journals rotated into `c:\p4\1\checkpoints` instead of something like `c:\p4\1\checkpoints.ha_bos`.

However, it is still worth ensuring things like:

- `offline_db` on HA is up-to-date
- all triggers (including any Swarm triggers) are appropriately configured
- `daily_backup.bat` will work appropriate after failover

2.5. Preparation for Failing Back - Setup original commit server as new Failover replica

After failover it is quite common to want to to "failback" to the original machine. This is fairly straight forwards if you are going to use the same server ids (master and failover replica).

The steps are:

- Reset the original master to be the failover replica
- Once it is up and running as a replica, then you can just use `p4 failover` on it, and then use these same instructions to reset the original HA replica to be running as a failover target.

The original master p4d service will have been stopped as part of failover, and will not be able to restart without some configuration changes.

Note that for convenience below, we assume SDP instance `1` and failover serverid `p4d_ha_bos`.

The commands below are run on the **on original master** server which will now become the HA failover replica.

1. Change the serverid to replica serverid:

```
cd /p4/1/root
mv server.id save/
```


© 2007-2024 Perforce Software, Inc.

```
mv state* save/
echo p4d_ha_bos> server.id
```

2. Reset the journal and logs:

```
cd /p4/1/logs
mkdir -p save
mv journal* save/
mv log log.old
```

3. Restart the service and ensure the `serviceUser` is logged in:

```
sudo systemctl restart p4d_1
/p4/common/bin/p4login -v -service 1
p4 pull -lj
```

4. You may wish to ensure that verification of recent archive/depot files has been done, especially if the service was stopped for hours or days since the original failover.

```
nohup /p4/common/bin/p4verify.sh -o MISSING -recent &
```

Please note details for definition of "recent" and how to extend that: p4verify.sh

At this stage, the replica should be replicating as normal. If it is not, then please follow standard replication trouble shooting procedures.

The Server spec will have had its `Options:` set to `nomandatory` - and this should be set back to `mandatory` when the replica is working correctly.

© 2007-2024 Perforce Software, Inc.

Chapter 3. Unplanned Failover

In this case there is no active participation of upstream server, so there is an increased 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 ...
```

3.1. Post Unplanned Failover

This is similar to Section 2.3, “Post Failover” with the exception of the next section below.

3.1.1. Resetting Downstream Replicas

In an unplanned failover scenario, if the standby server is not a 'mandatory' standby, it is possible that there is a journal synchronization problem with downstream replicas.

The output of `p4 pull -lj` may indicate an error, and/or there may be errors in the log:

 © 2007-2024 Perforce Software, Inc.

```
grep -A4 error: /p4/1/logs/log | less
```

If you need to reset the replica to restart from the beginning of the current journal it is attempting to pull, then the process is:

1. Stop the replica:

   ```
   sudo systemctl stop p4d_1
   ```

2. Remove the `state` file:

   ```
   cd /p4/1/root
   mv state save/
   ```

3. Restart the replica

   ```
   sudo systemctl start p4d_1
   ```

4. Recheck the log for errors as above.

3.2. Unplanned Failover on Windows

The extra steps required are basically the same as in Section 2.4, “Failing over on Windows” as well as the steps in Section 3.1, “Post Unplanned Failover”

© 2007-2024 Perforce Software, Inc.