SDP Windows to Linux Migration Guide

Perforce Professional Services

Version v2023.1, 2023-12-01

Table of Contents

DRAFT NOTICE

This document is in DRAFT status and should not be used. It is a preview of a document to be completed in a future release.

Preface

This guide documents the process for migrating a Helix Core service from Windows server to Linux. A migration can be minimally disruptive to users if planned and executed properly, and has significant benefits. The purpose of this document is to help inform the planning.

For purposes of this document, it does not matter if the servers are on-premises ("on-prem") or in a private or public cloud environment such as AWS, Azure, or GCP.

The Windows service may or may not be operated using Windows SDP. Regardless of whether the Windows service is managed with SDP, the Windows service is left alone during the migration.

Please Give Us Feedback

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

Chapter 1. Overview

A Migration has these elements:

- Planning: See Chapter 2, *Migration Planning*.
- Provision New Linux Server machines.
- Install Perforce Helix on Linux.
- Setup Linux Replica server spec on Windows.
- Pull archives. This may take a long while if there is a lot of data to pull.
- Correct data issues identified in planning.

Each of these components is covered in detail in this guide.

Chapter 2. Migration Planning

There are several things to account for in planning a Windows to Linux migration. This section covers things to be aware of when planning.

2.1. Migration Strategy

Several migration strategies are possible. This document focuses on the Failover Style strategy. This entails creating a server spec (ServerID) we'll call `p4d_fs_linux_xfer`, that will operate for a time as a Linux replica of the current production Windows commit server. Depending on various factors such as data scale, project priority and complexity, etc. this Linux replica of the Windows commit server may operate for days, weeks or even months before it is eventually ready for the planned "failover" that will make the Linux server become the new commit server.

This Failover strategy has several benefits:

- Minimum disruption to end users for the cutover.
- Allows for extensive testing of the new Linux server(s) and infrastructure prior to cutover.
- The effect on the original Windows server(s) and infrastructure is minimal.
- Rollback, while hopefully not necessary, is straightforward.

While planning and preparation will take time and effort, the disruption to end users can be minimal.

The Failover strategy requires that the Windows Helix Core P4D service be at version 2019.1 or later. If it s not already at 2019.1 or later, than the plan should account for first upgrading the Windows service in place to 2019.1.

Other strategies could be considered that would not require upgrading if avoiding an in-place upgrade is a priority. That would entail longer downtime and other complexity. Such options are not explored in this document.

2.2. Incompatible Configuration Settings

Using the `p4 configure` command to interact with `db.config` is a good way, and in many cases the only way, to set various configuration items with a Helix Core server. However, there are certain settings that must not be defined with `p4 configure`, as they conflict with settings the SDP defines with shell environment variables.

Review the output of the command `p4 configure show allservers` and see if any of the following are set:

- `P4JOURNAL`
- `P4PORT`
- `P4LOG`

If any of these are set with `p4 configure`, the migration plan will need to deal with unsetting them after first ensuring they are set in some other way on the Windows service. Following is an example of how to replace how P4LOG is set displays in the output of `p4 configure show all servers`. Note that changing this requires a brief service restart to take affect.

2.2.1. Sample Procedure to replace P4LOG configurable

This is an example of how this might be done if the Windows service name is `Perforce`:

```
p4 set -S Perforce P4LOG=L:\p4logs\p4d.log
```

That will set the P4LOG variable so that it is associated with the Windows service named `Perforce`. Once that is done, it can be unset, such as in this example:

```
p4d.exe -r E:\PerforceRoot "-cunset P4LOG"
```

Next, stop and then start the Windows service as you normally would.

2.2.2. Other Windows Paths in Configuration

Also scan for things like Windows paths, such as Structured Logs defined to reference a Windows path. Such things will need to be be overridden in the server spec for the Linux replica. For example, if you see:

```
any: serverlog.file.11=E:\PerforceRoot\triggers.csv
```

You'll want to create an override for the Linux replica by doing:

```
p4 configure set p4d_fs_linux_xfer#serverlog.file.11=/p4/1/logs/triggers.csv
```

2.3. Depot Root and Depot Spec Map Fields

Perforce Helix depot specs have a field named `Map:` that, if used, must be eliminated prior to the deployment of a Linux replica. Further, the `server.depot.root` configurable must be set on the commit server.

If done carefully, the changes to set `server.depot.root` and clear the `Map:` field of each depot spec can be done non-disruptively on the live running Windows Perforce Helix Core service, and must be done before creating the checkpoint used to seed the Linux replica.

The key to making the change non-disruptively is to understand that the p4d server will use the `Map:` field value to see if it is set to anything other than the default, and otherwise will fall back to the `server.depot.root` configurable to find depots. If the value of the `Map:` field of any given depot is `TheDepotName/…`, that means the value is not explicitly set.

Before making changes, the singular `server.depot.root` value must be made to work for all depots. A common goal early on is to make the single `server.depot.root` path work without actually moving any files, but by using Windows directory symlinks. If individual depots are on different drives, put symlinks to all depots in the directory pointed to by the `server.depot.root` configurable so that p4d can find all depot files from that path. You may also find the Map fields use Windows UNC paths or if Windows junctions.

Special planning may be required if there are any depots of type `archive`.

2.4. The journalPrefix

The Windows commit server must have the `journalPrefix` value be set in order to set up the Linux replica. It can be set to any value that works to enable the p4d service to find its archives, but cannot be unset.

2.5. Uncompressed Journals

Examine how checkpoints and journals are currently taken on the Windows environment (or of they are taken at all).

If journals on the Windows service are compressed, replication will not work. Replicas require uncompressed journals.

As a general rule, the `p4d -jc` command is best done with `-Z`, which compresses the checkpoint file, but not the numbered journal files. Changes to any custom scripts that manage checkpoints in the Windows environment may be warranted.

2.6. Helix Core Components

Consider what Perforce Helix systems are in your environment that may need to be handled, such as:

- Helix Core Server (P4D)
- Helix Broker (P4Broker)
- Helix Proxy (P4P)
- Helix Swarm
- P4DTG

2.7. Helix Core Topology

Is your server a single machine, or are there many server machines? In any case, you'll want to think in terms of a "Big Blue/Green Deploy." Every active Windows server machine in the current production topology (the "Blue" servers), including all replicas, edges, and proxies, will all need equivalent Linux server machines to replace them (the "Green" servers). Replicas are straightforward to handle. Handling edges is more complex but doable. Don't forget proxies — they need to do the Windows → Linux thing too. (Proxies could have been Linux all along even with a

Windows P4D, but don't forget to check that).

2.8. Custom Triggers and Extensions

Any custom Triggers or Extensions will need to be reviewed. Any that can't be discarded will need to evaluated for porting and testing needs.

2.9. Moving Archive Files

Once the Linux replicas are setup, a variety of strategies can be used to transfer archive files.

Plan 3 cycles of `p4verify.sh`, to get p4d to pull the archives. The first, starting with no archive files, is to start a bulk pull. That could take days or weeks depending on data scale. The second to fill in gaps, and the 3rd should be clean.

Depending on scale of data, you may want to consider using outside-p4d mechanisms for transferring some archives (especially the `.gz` files, `,v` files should be transferred with `p4 pull` ideally).

Lots of variations on how to get the archives files there. Using `p4 pull` has an advantage better that, if the Linux p4d writes the archive, it can always find it, even it it's funky with Unicode cruft in the path. By contrast, files copied outside p4d may not be found by the Linux p4d. However, for bulk pulls of Terabytes of data, a Windows port of rsync, at least for `.gz` files, will be much faster. You'll need a live running rsync service on Linux for the Windows port of rsync to talk to. There are many options here; somehow or other get the files in place so p4verify.sh is happy.

2.10. Avoid Case Conversion

If there is a desire to convert the case to become case-sensitive, that should be deferred and done as a separate project. A Windows to Linux migration that preserves the original Windows case-insensitive behavior is non-disruptive. A case conversion is likely to be disruptive to users, and is complex enough that it should be relegated to a separate project from a Windows to Linux migration. The case conversion should be done after the Windows to Linux migration is complete and tested.

2.11. Combining Upgrade with Migration

If the priority is to avoid upgrading or touching the Windows environment, an upgrade to a modern Helix Core version can be done to the Linux server during the cutover, as part of the Windows to Linux migration project.

Alternately, you can upgrade the Windows P4D in place first, and then set up the Linux replica on the same modern P4D version. If the starting Windows version is 2019.1+, a Failover style migration is possible; otherwise a different strategy is needed.

Typically we recommend doing the failover-then-upgrade in the same maintenance window as the Windows to Linux migration. That is, failover to the new server on Linux on the same p4d version as Windows was initially. Then once on Linux, do the standard SDP upgrade procedure for Linux, using `upgrade.sh`.

2.12. DRY RUN

At least one Dry Run is required to confidently execute a migration. Plan to have at least one.

In the dry run, the `p4 failover` command is NOT used. Instead, the Linux service is stopped, and the `$P4ROOT/server.id` file is simply hand-edited to be the ServerId the the commit server. Then the service is restarted.

At that point, the Linux commit server will believe itself to be the new commit server, even though users will still be using the Windows server for real work. Then the Linux server can be tested in various ways:

- Test connectivity from all user access points.
- Test connectivity from all server access points, including replicas, proxies, and any integrated systems such as Jenkins, Swarm, P4DTG, etc.
- If there are any `ldap` specs, ensure the targeted LDAP servers can be reached from the Linux server. (This may require firewall adjustments).

2.13. Setup Linux Replica ServerID

On the Windows commit server, create a server spec to represent p4d on Linux. Call it `p4d_fs_linux_xfer`.

Chapter 3. Provision New Linux Server Machines

EDITME - Add content here.

3.1. Deploy New Linux Server machine

3.1.1. Select Operating System

As of this writing, the best options are:

- Ubuntu 20.04 (not 22.04 just yet)
- RHEL/Rocky Linux 8 (not 9 just yet)
- Amazon Linux 2 (not Amazon Linux 2023 just yet)

Chapter 4. Install Perforce Helix on Linux

4.1. Install Helix Core Software

On the Green Linux server machines that do not yet have any data, use the Helix Installer, do a Configured Install.

The Helix Installer is only to be used on truly "green" server machines, those with no Helix Core data on them yet.

```
su -
mkdir -p /hxdepots/reset
cd /hxdepots/reset
curl -L -s -O
https://swarm.workshop.perforce.com/download/guest/perforce_software/helix-
installer/main/src/reset_sdp.sh
chmod +x reset_sdp.sh
./reset_sdp.sh -C > settings.cfg
```

In `settings.cfg`, change these settings:

- DNS_name_of_master_server=
- P4_PORT=
- Instance=
- Password=
- CaseSensitive=0
- P4USER=
- ServerID=
- ServerType=
- P4BinRel=
- P4APIRel=

Then run the script:

```
./reset_sdp.sh -no_sd -c settings.cfg 2>&1 | tee log.reset_sdp.txt
```

```
su - perforce
p4 set
```

```
cd /p4/common/site
[[ -d config ]] || mkdir config
cd config
```

4.2. Create the Linux replica.

Temporary Hack:

vi /p4/common/site/config/p4_N.vars.local

```
export P4MASTER_ID=windows.p4d
export P4MASTERPORT=192.168.1.5:1666
export P4PORT=$P4MASTERPORT
```

p4login -v

cd /p4/common/config vi SiteTags.cfg

azwestus2: Azure data center

Add to Protections:

```
super group ServiceUsers * //...
```

mkrep.sh -t fs -s azwestus2 -r TestMachine

Undo Temporary Hack:

vi /p4/common/site/config/p4_N.vars.local

```
#export P4MASTER_ID=Master
#export P4PORT=$P4MASTERPORT
export P4MASTERPORT=120.2:43430
```

Appendix A: Why Migrate?

Migrations from Windows to Linux have been the single most consistent theme in Perforce Consulting in many years, for many reasons.

EDITME Add some of the many reasons.

Chapter 5. DRAFT NOTICE

This document is in DRAFT status and should not be used. It is a preview of a document to be completed in a future release.