PERFORCE

Server Deployment Package

User Guide (for Unix)

Perforce Software, Inc.

8 January 2016

PERFORCE

Preface

This guide tells you how to set up a new Perforce Helix Server installation using the Server Deployment Package (SDP). Recommendations for optimal system maintenance and performance are included as well. The SDP follows best practices for Perforce server configuration and administration. The SDP consists of standard configuration settings, scripts, and tools, which provide several key features.

- A volume layout designed for maximum data integrity and server performance.
- Automated offline checkpointing and backup procedures for server metadata.
- Replication to another server.
- Easy maintenance of user accounts, labels, workspaces, and other data.
- User authentication using LDAP or Active Directory.

This guide assumes some familiarity with Perforce, and does not duplicate the basic information in the Perforce user documentation. For basic information on Perforce, consult Introducing Perforce. For system administrators, the Perforce System Administrator's Guide is essential reading. All documentation is available from the Perforce web site at http://www.perforce.com.

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.

Table of Contents

Overview

The SDP has four main components:

- Hardware and storage layout recommendations for Perforce.
- Scripts to automate offline checkpoints and other critical maintenance activities.
- Scripts to replicate the Perforce journal to another volume or server.
- Scripts to assist with user account maintenance and other routine administration tasks.

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

The SDP should be versioned in a depot (e.g. `//perforce`) as part of the installation process.

The directory structure of the SDP is shown below in Figure 1: SDP Package Directory Structure. This includes all SDP files, including documentation and maintenance scripts. A subset of these files are deployed to server machines during the installation process.

```
sdp
   doc
   Maintenance (Admin scripts)
   Server (Core SDP Files)
         setup (typemap, configure, etc)
         test (automated test scripts)
         Unix
             setup
             p4
               1
                 bin
               common
                     bin (Backup scripts, etc)
                        triggers (Example triggers)
                     config
                     etc
                        cron.d
                        init.d
                     lib
                     test
```

Figure 1: SDP Package Directory Structure

Configuring the Perforce Server

This chapter tells you how to configure a Perforce server machine and an instance of the Perforce Server. These topics are covered more fully in the System Administrator's Guide and in the Knowledge Base; this chapter covers the details most relevant to the SDP.

The SDP can be installed on multiple server machines, and each server machine can host one or more Perforce server instances. (In this guide, the term *server* refers to a Perforce server instance unless otherwise specified.) Each server instance is assigned a number. This guide uses instance number 1 in the example commands and procedures. Other instance numbers can be substituted as required.

Optionally, instances can be given a short tag name, such as 'abc', rather than a number. Manual configuration is required to use tag names rather than the default numeric values.

This chapter also describes the general usage of SDP scripts and tools.

Volume Layout and Hardware

To ensure maximum data integrity and performance, use three different physical volumes for each server instance. Three volumes can be used for all instances hosted on one server machine, but using three volumes per instance reduces the chance of hardware failure affecting more than one instance.

- **Perforce metadata (database files):** Use the fastest volume possible, ideally RAID 1+0 on a dedicated controller with the maximum cache available on it. This volume is normally called `/metadata`.
- **Journals and logs:** Use a fast volume, ideally RAID 1+0 on its own controller with the standard amount of cache on it. This volume is normally called `/logs`. If a separate logs volume is not available, put the logs on the `metadata` volume.
- **Depot data, archive files, scripts, and checkpoints**: Use a large volume, with RAID 5 on its own controller with a standard amount of cache or a SAN or NAS volume. This volume is the only volume that must be backed up. The backup scripts place the metadata snapshots on this volume. This volume can be backed up to tape or another long term backup device. This volume is normally called `/depotdata`.

For optimal performance on UNIX machines, use the XFS file system.

If three controllers are not available, put the logs and `depotdata` volumes on the same controller.

Do not run anti-virus tools or back up tools against the `metadata` volume(s) or `logs` volume(s), because they can interfere with the operation of the Perforce server.

Back up everything on the depotdata volume(s). Avoid backing up the metadata volume directly, because doing so can interfere with the operation of a live Perforce server, potentially corrupting data. The checkpoint and journal process archive the metadata on the depotdata volume. Backing up the logs volume is optional.

The SDP assumes (but does not require) the three volumes described above. On Unix/Linux platforms, the SDP CREATES a convenience directory containing links to the three volumes for each instance. This convenience directory is called `/p4`. The volume layout is shown in Figure 2: SDP Runtime Structure and Volume Layout, including the links that constitute the `/p4` directory for Unix/Linux platforms. (On Windows, a similar convenience directory c:\p4 is created and mklink is used.) The convenience directory enables easy access to the different parts of the file system for each instance. For instance, the directory `/p4/1/root` has the database for instance 1, `/p4/1/logs` has the logs for instance 1, `/p4/1/bin` has the binaries for instance 1, and `/p4/common/bin` contains the scripts common to all instances.

View Figure 2: SDP Runtime Structure and Volume Layout (below), viewed from the top down, displays a Perforce *application* administrator's view of the system, which shows how to navigate the directory structure to find databases, log files, and versioned files in the depots. Viewed from the bottom up, it displays a Perforce *system* administrator's view, emphasizing the physical volume where Perforce data is stored.

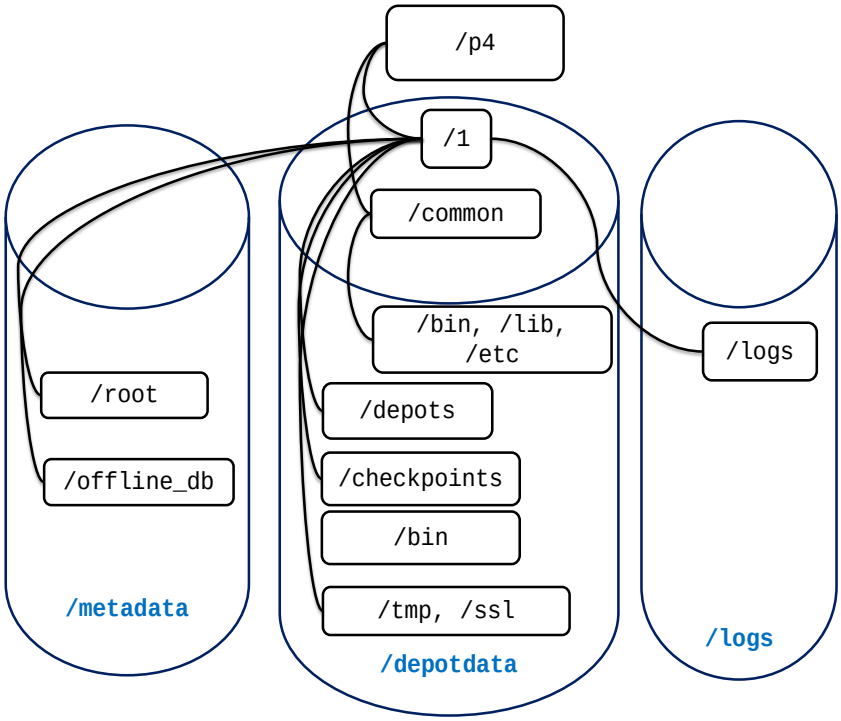

Figure 2: SDP Runtime Structure and Volume Layout

Memory and CPU

Make sure the server has enough memory to cache the **db.rev** database file and to prevent the server from paging during user queries. Maximum performance is obtained if the server has enough memory to keep all of the database files in memory.

Below are some approximate guidelines for allocating memory.

- 1.5 kilobyte of RAM per file stored in the server.
- 32 MB of RAM per user.

Use the fastest processors available with the fastest available bus speed. Faster processors with a lower number of cores provide better performance for Perforce. Quick bursts of computational speed are more important to Perforce's performance than the number of processors, but have a minimum of two processors so that the offline checkpoint and back up processes do not interfere with your Perforce server.

General SDP Usage

This section presents an overview of the SDP scripts and tools. Details about the specific scripts are provided in later sections.

Unix/Linux

Most scripts and tools reside in `/p4/common/bin`. The `/p4/instance/bin` directory contains scripts that are specific to that instance such as wrappers for the `p4d` executable.

Older versions of the SDP required you to always run important administrative commands using the `p4master_run` script, and specify fully qualified paths. This script loads environment information from `/p4/common/bin/p4_vars`, the central environment file of the SDP, ensuring a controlled environment. The p4_vars file includes instance specific environment data from `/p4/common/config/instance.vars`. The p4master_run script is still used when running p4 commands against the server unless you set up your environment first by sourcing p4_vars with the instance as a parameter. Administrative scripts, such as daily_backup.sh, no longer need to be called with p4master_run however, they just need you to pass the instance number to them.

When invoking a Perforce command directly on the server machine, use the `p4_instance` wrapper that is located in `/p4/instance/bin`. This wrapper invokes the correct version of the `p4` client for the instance. The use of these wrappers enables easy upgrades, because the wrapper is a link to the correct version of the `p4` client. There is a similar wrapper for the `p4d` executable, called `p4d_instance`.

Below are some usage examples for instance 1.

Example	Remarks
`/p4/common/bin/p4master_run 1 /p4/1/bin/p4_1 admin stop`	Run `p4 admin stop` on instance 1
`/p4/common/bin/live_checkpoint.sh 1`	Take a checkpoint of the live database on instance 1
`/p4/common/bin/p4login 1`	Log in as the p4admin user on instance 1.

Some maintenance scripts can be run from any client workspace, if the user has administrative access to Perforce. For example, to run the script that archives old workspaces and branches, run:

```
/ws_root/Perforce/sdp/Maintenance/accessdate.py
```

If an error occurs due to the default Python interpreter used by the script, invoke Python first:

```
/bin/python /ws_root/Perforce/sdp/Maintenance/accessdate.py
```

In the preceding example `/ws_root` is the root of the client workspace, and the python interpreter is located in `/bin`.

Monitoring SDP activities

The important SDP maintenance and backup scripts generate email notifications when they complete.

For further monitoring, you can consider options such as:

- Making the SDP log files available via a password protected HTTP server.
- Directing the SDP notification emails to an automated system that interprets the logs.

Installing the Perforce Server and the SDP

This chapter tells you how to install a Perforce server instance in the SDP framework. For more details about server installation, refer to the Perforce System Administrator's Guide.

Many companies use a single Perforce Server to manage their files, while others use multiple servers. The choice depends on network topology, the geographic distribution of work, and the relationships among the files being managed. If multiple servers are run, assign each instance a number and use that number as part of the name assigned to depots, to make the relationship of depots and servers obvious.

The default P4PORT setting used by the SDP is `instance666`. For example, instance `1` runs on port `1666`. Each Perforce instance uses its hostname as an identifying name; this identification is used for replicated servers. This can easily be changed in `/p4/common/bin/p4_vars`.

For any instances that are named rather than numbered, then the `/p4/common/bin/p4_vars` file must be customized to assign a numeric P4PORT value to each named instance.

To install the SDP, you must have root (super-user or administrator) access to the server machine.

Installing on Unix/Linux Machines

To install Perforce Server and the SDP, perform the following basic steps that are discussed below:

- Set up a user account, file system, and configuration scripts.
- Run the configuration script.
- Start the server and configure the required file structure for the SDP.

Initial setup

Prior to installing the Perforce server, perform the following steps.

1. Create a user called `p4admin` (It can be a different name if you prefer, in which case modify the OSUSER entry in the mkdirs.sh script - see item 12 below). Set the user's home directory to `/p4` on a local disk.
2. Create a group called `perforce` (again, can be a different name – see OSGROUP in mkdirs.sh) and make it the `perforce` user's primary group.
3. Create or mount the server file system volumes (`/depotdata, /metadata, /logs`).
4. Copy the SDP to the directory `/depotdata/sdp`. We will refer to this directory as `$SDP`. Make the entire `$SDP` directory writable.
5. Download the appropriate `p4` and `p4d` binaries for your release and platform from `ftp.perforce.com` (log in as `anonymous`) and place them in `$SDP/Server/Unix/p4/common/bin`. ***Do not*** rename them to include the version number; this step is done automatically for you by the SDP.
6. cd to `$SDP/Server/Unix/setup` and edit `mkdirs.sh` - set all of the variables in the configuration variables section for your company.
7. As the root user, cd to `$SDP/Server/Unix/setup`, and run this:

   ```
   mkdirs.sh instance
   ```

   Examples:

   ```
   mkdirs.sh 1

   mkdirs.sh Master
   ```

   This script configures the first Perforce Server instance. To configure additional instances, run `mkdirs.sh` again, specifying the instance number each time. For example, to configure a second and third instance, issue the following commands:

   ```
   mkdirs.sh 2

   mkdirs.sh 3
   ```

8. Put the Perforce license file for the server into `/p4/1/root`. Note, if you have multiple instances and have been provided with port-specific licenses by Perforce, the appropriate license file must be stored in the appropriate `/p4/instance/root` folder.
9. Make the Perforce server a system service that starts and stops automatically when the machine reboots. Running `mkdirs.sh` creates a set of init scripts for various Perforce server products in the instance-specific bin folder:

```
/p4/1/bin/p4d_1_init

/p4/1/bin/p4broker_1_init

/p4/1/bin/p4p_1_init

/p4/1/bin/p4ftpd_1_init

/p4/1/bin/p4dtg_1_init

/p4/1/bin/p4web_1_init
```

The steps required to complete the configuration will vary depending on the Unix distribution being used.

The following sample commands enable init scripts as system services on RedHat / CentOS (up to version 6) and SuSE (up to version 11). Run these commands the `root` user:

```
cd /etc/init.d

ln –s /p4/1/bin/p4d_1_init

    chkconfig ––add p4d_1_init

    chkconfig p4d_1_init on
```

Run the '`ln –s`' and two '`chkconfig`' commands for any other init scripts, besides `p4d_1_init`, that you wish to operate on for that instance and on the current machine, such as `p4broker_1_init` or `p4web_1_init`. Remove init scripts for any services not needed on that machine. Note that the broker and P4Web require additional configuration before those services will start.

RHEL 7, CentOS 7, SuSE 12, Ubuntu (v15.04) (and other) distributions utilize **systemd / systemctl** as the mechanism for controlling services, replacing the earlier init process. At present `mkdirs.sh` does **not** generate the `systemd` configuration file(s) automatically, but a sample is included in the SDP distribution in (`$SDP/Server/Unix/setup/systemd`), along with a `README.md` file that describes the configuration process.

Ubuntu (pre 15.04), MacOS and other Unix derivatives use different mechanisms to enable services. If your Linux distribution does not have the chkconfig or systemctl utilities, consult your distribution's documentation for information on enabling services.

Upgrading an existing SDP installation

If you have an earlier version of the Server Deployment Package (SDP) installed, you'll want to be aware of the new `-test` flag to the SDP setup script, `mkdirs.sh`. The following update instructions assume a simple, single-server topology.

See the instructions in the file `README.md / README.html` in the root of the SDP directory.

Configuration script

The `mkdirs.sh` script executed above resides in `$SDP/Server/Unix/setup`. It sets up the basic directory structure used by the SDP. Carefully review the header of this script before running it, and adjust the values of the variables near the top of the script as required. The important parameters are:

Parameter	Description
MD	Name of the `metadata` volume
DD	Name of the `depotdata` volume
LG	Name of the `logs` volume
ADMINUSER	P4USER value of a Perforce super user that operates SDP scripts, typically `perforce` or `p4admin`.
OSUSER	Operating system user that will run the Perforce instance, typically `perforce`.
OSGROUP	Operating system group that `OSUSER` belongs to, typically `perforce`.
SDP	Path to SDP distribution file tree
CASESENSITIVE	Indicates if server has special case sensitivity settings
CLUSTER	Indicates if server is running in cluster
P4ADMINPASS	Password to use for Perforce superuser account
P4SERVICEPASS	Service User's password for replication
P4DNSNAME	Fully qualified DNS name of the Perforce master server machine

For a detailed description of this script, see Appendix A – Directory Structure Configuration Script for Linux/Unix.

Starting/Stopping Perforce Server Products

The SDP includes templates for initialization (start/stop) scripts, "init scripts," for a variety of Perforce server products, including:

- p4d
- p4broker
- p4p
- p4dtg
- p4ftpd
- p4web

The init scripts are named `/p4/`***`instance`***`/bin/svc_`***`instance`***`_init.`

For example, the init script for starting `p4d` for Instance 1 is `/p4/1/bin/p4d_1_init`. All init scripts accept at least `start`, `stop`, and `status` arguments. The perforce user can start `p4d` by calling:

```
p4d_1_init start
```

And stop it by calling:

```
p4d_1_init stop
```

Once logged into Perforce as a super user, the `p4 admin stop` command can also be used to stop p4d.

All init scripts can be started as the `perforce` user or the `root` user (except `p4web`, which must start initially as root). The application runs as the `perforce` user in any case. If the init scripts are configured as system services, they can also be called by the `root` user using the service command, as in this example to start `p4d`:

```
service p4d_1_init start
```

Templates for the init scripts used by `mkdirs.sh` are stored in:

```
/p4/common/etc/init.d
```

There are also basic crontab templates for a Perforce master and replica server in:

```
/p4/common/etc/cron.d
```

These define schedules for routine checkpoint operations, replica status checks, and email reviews.

The Perforce should have a super user defined as named by the P4USER setting in mkdir.
To configure and start instance 1, follow these steps:

2. Start the Perforce server by calling `p4d_1_init start`.

3. Ensure that the admin user configured above has the correct password defined in `/p4/common/bin/adminpass`, and then run the `p4login` script (which calls the `p4 login` command using the adminpass file).

4. For new servers, run this script, which sets several recommended configurables:

```
$SDP/Server/setup/configure_new_server.sh
```

For existing servers, examine this file, and manually apply the `p4 configure` command to set configurables on your Perforce server.

5. Initialize the perforce user's crontab with one of these commands:

```
crontab /p4/p4.crontab
 or
crontab /p4/p4.crontab.rep
```

and customise execution times for the commands within the crontab files to suite the specific installation.

To verify that your server installation is working properly:

1. Issue the `p4 info` command, after setting appropriate environment variables. If the server is running, it will display details about its settings.

Now that the server is running properly, copy the following configuration files to the `depotdata` volume for backup purposes:

- Any init scripts used in `/etc/init.d`.
- A copy of the crontab file, obtained using `crontab -l`.
- Any other relevant configuration scripts, such as cluster configuration scripts, failover scripts, or disk failover configuration files.

Archiving configuration files

Now that the server is running properly, copy the following configuration files to the `depotdata` volume for backup:

- The scheduler configuration.
- Cluster configuration scripts, failover scripts, and disk failover configuration files.

Configuring protections, file types, monitoring and security

After the server is installed and configured, most sites will want to modify server permissions (protections) and security settings. Other common configuration steps include modifying the file type map and enabling process monitoring. To configure permissions, perform the following steps:

1. To set up protections, issue the `p4 protect` command. The protections table is displayed.

2. Delete the following line:

```
write user * * //depot/...
```

3. Define protections for your server using groups. Perforce uses an inclusionary model. No access is given by default, you must specifically grant access to users/groups in the protections table. It is best for performance to grant users specific access to the areas of the depot that they need rather than granting everyone open access, and then trying to remove access via exclusionary mappings in the protect table even if that means you end up generating a larger protect table.

4. To set the server's default file types, run the p4 typemap command and define typemap entries to override Perforce's default behavior.

Add any file type entries that are specific to your site. Suggestions:

- For already-compressed file types (such as `.zip, .gz, .avi, .gif`), assign a file type of `binary+Fl` to prevent the server from attempting to compress them again before storing them.
- For regular binary files, add binary+l to make so that only one person at a time can check them out.
- A sample file is provided in `$SDP/Server/config/typemap`

1. To make your changelists default to restricted (for high security environments):

```
p4 configure set defaultChangeType=restricted
```

Other server configurables

There are various configurables that you should consider setting for your server.

Some suggestions are in the file: `$SDP/Server/setup/configure_new_server.sh`

Review the contents and either apply individual settings manually, or edit the file and apply the newly edited version. If you have any questions, please see the configurables section in Appendix of the Command Reference Guide (get the right version for your server!). You can also contact support regarding questions.

Backup, Replication, and Recovery

Perforce servers maintain *metadata* and *versioned files*. The metadata contains all the information about the files in the depots. Metadata resides in database (`db.*`) files in the server's root directory (`P4ROOT`). The versioned files contain the file changes that have been submitted to the server. Versioned files reside on the `depotdata` volume.

This section assumes that you understand the basics of Perforce backup and recovery. For more information, consult the Perforce System Administrator's Guide and the Knowledge Base articles about replication.

Typical Backup Procedure

The SDP's maintenance scripts, run as *cron* tasks, periodically back up the metadata. The weekly sequence is described below.

Seven nights a week, perform the following tasks.

1. Truncate the active journal.
2. Replay the journal to the offline database. (Refer to Figure 2: SDP Runtime Structure and Volume Layout for more information on the location of the live and offline databases.)
3. Create a checkpoint from the offline database.
4. Recreate the offline database from the last checkpoint.

Once every six months, perform the following tasks.

1. Stop the live server.
2. Truncate the active journal.
3. Replay the journal to the offline database. (Refer to Figure 2: SDP Runtime Structure and Volume Layout for more information on the location of the live and offline databases.)
4. Archive the live database.
5. Move the offline database to the live database directory.
6. Start the live server.
7. Create a new checkpoint from the archive of the live database.
8. Recreate the offline database from the last checkpoint.
9. Verify all depots.

This normal maintenance procedure puts the checkpoints (metadata snapshots) on the `depotdata` volume, which contains the versioned files. Backing up the `depotdata` volume with a normal backup utility like *robocopy* or *rsync* provides you with all the data necessary to recreate the server.

To ensure that the backup does not interfere with the metadata backups (checkpoints), coordinate backup of the `depotdata` volume using the SDP maintenance scripts.

The preceding maintenance procedure minimizes server downtime, because checkpoints are created from offline or saved databases while the server is running.

Be sure to back up the entire `depotdata` volume using a normal backup utility.

tional configuration, the norm
nges. To provide an optimal R
ools for replication.

Full One-Way Replication

Perforce supports a full one-way replication of data from a master server to a replica, including versioned files. The `p4 pull` command is the replication mechanism, and a replica server can be configured to know it is a replica and use the replication command. The `p4 pull` mechanism requires very little configuration and no additional scripting. As this replication mechanism is simple and effective, we recommend it as the preferred replication technique. Replica servers can also be configured to only contain metadata, which can be useful for reporting or offline checkpointing purposes. See the Distributing Perforce Guide for details on setting up replica servers.

If you wish to use the replica as a read-only server, you can use the P4Broker to direct read-only commands to the replica or you can use a forwarding replica. The broker can do load balancing to a pool of replicas if you need more than one replica to handle your load. Use of the broker may require use of a P4AUTH server for authentication.

Replication handles all server metadata and versioned file content, but not the SDP installation itself or other external scripts such as triggers. Use tools such as `robocopy` or `rsync` to replicate the rest of the `depotdata` volume.

Replication Setup

To configure a replica server, first configure a machine identically to the master server (at least as regards the link structure such as /p4, /p4/common/bin and /p4/*instance*/*), then install the SDP on it to match the master server installation. Once the machine and SDP install is in place, you need to configure the master server for replication.

Perforce supports many types of replicas suited to a variety of purposes, such as:

- Real-time backup,
- Providing a disaster recovery solution,
- Load distribution to enhance performance,
- Distributed development,

- Dedicated resources for automated systems, such as build servers, and more.

We always recommend first setting up the replica as a read-only replica and ensuring that everything is working. Once that is the case you can easily modify server specs and configurables to change it to a forwarding replica, or an edge server etc.

In the sample below, the replica name will be `replica1`, it is instance 1 on a particular host, the service user name is `svc_replica1`, and the master server's hostname is `svrmaster`.

The following sample commands illustrate how to setup a simple read-only replica.

First we ensure that journalPrefix is set appropriately for the master server (in this case we assume instance 1 rather than a named instance):

```
p4 configure set master#journalPrefix=/p4/1/checkpoints/p4_1
```

Then we set values for the replica itself:

```
p4 configure set replica1#P4TARGET=svrmaster:1667

p4 configure set "replica1#startup.1=pull -i 1"

p4 configure set "replica1#startup.2=pull -u -i 1"

p4 configure set "replica1#startup.3=pull -u -i 1"

p4 configure set "replica1#startup.4=pull -u -i 1"

p4 configure set "replica1#startup.5=pull -u -i 1"

p4 configure set "replica1#db.replication=readonly"

p4 configure set "replica1#lbr.replication=readonly"

p4 configure set replica1#serviceUser=svc_replica1
```

Then the following also need to be setup:

- Create a service user for the replica (Add the `Type: service` field to the user form before saving):

  ```
  p4 user -f svc_replica1
  ```

- Set the service user's password:

```
p4 passwd svc_replica1
```

- Add the service user `svc_replica1` to a specific group `ServiceUsers` which has a `timeout` value of `unlimited`:

```
p4 group ServiceUsers
```

- Make sure the `ServiceUsers` group has super access in protections table:

```
p4 protect
```

Now that the settings are in the master server, you need to create a checkpoint to seed the replica. Run:

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

When the checkpoint finishes, rsync the checkpoint plus the versioned files over to the replica:

```
rsync -avz /p4/1/checkpoints/p4_1.ckp.###.gz
     perforce@replica:/p4/1/checkpoints/.

rsync -avz /p4/1/depots/ perforce@replica:/p4/1/depots/
```

(Assuming perforce is the OS user name and replica is the name of the replica server in the commands above, and that ### is the checkpoint number created by the daily backup.)

Once the rsync finishes, go to the replica machine run the following:

```
/p4/1/bin/p4d_1 -r /p4/1/root -jr -z /p4/1/checkpoints/p4_1.ckp.###.gz
```

Login as the service user (specifying appropriate password when prompted), and making sure that the login ticket generated is stored in the same place as specified in the P4TICKETS configurable value set above for the replica (the following uses bash syntax):

```
P4TICKETS=/p4/1/.p4tickets /p4/1/bin/p4_1 -p svrmaster:1667 -u
     svc_replica1 login
```

Start the replica instance:

```
/p4/1/bin/p4d_1_init start
```

Now, you can log into the replica server itself and run `p4 pull -lj` to check to see if replication is working. If you see any numbers with a negative sign in front of them, replication is not working. The most likely cause of this is that the service user is not logged in. Rerun the steps above to login the service user and check again. If replication still is not working, check `/p4/1/logs/log` on the replica, and also look for authentication failures in the log for the master instance on `svrmaster`.

The final steps for setting up the replica server are to set up the crontab for the replica server, and set up the rsync trust certificates so that the replica scripts can run rsync without passwords.

The replica crontab is in `/p4/common/etc/init.d/crontab.replica`.

To configure the rsync trust:

On both the master and replica servers, go to the perforce user's home directory and run:

```
ssh-keygen -t rsa
```

Just use the defaults for the questions it asks.

Now from the master, run:

```
rsync -avz ~/.ssh/id_rsa.pub perforce@replica:~/.ssh/authorized_keys
```

and from the replica, run:

```
rsync -avz ~/.ssh/id_rsa.pub perforce@master:~/.ssh/authorized_keys
```

You can validate everything is working by manually running the replica crontab scripts as below. Please note that depending on the size of your checkpoints and the bandwidth between your replica and master server, these commands may take hours to rsync the checkpoints across.

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

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

Check the replica crontab template for default configuration.

- It runs `sync_replica.sh` 6 days a week at 3am
    - copy the latest checkpoint from the master server using rsync
    - restores it to the `offline_db` directory on the replica - this makes it quick to get the replica up and running with defragmented db.* files
- It runs `weekly_sync_replica.sh` on the 7th day of the week at 3am
    - copies latest checkpoint from master using rsync
    - restores it to the `offline_db` directory as previously
    - stops the replica and replaces the main db.* files with the `offline_db` ones
    - The last step is the reason for the warning in the template crontab: "Don't run the line below on replicas that are being used" as the replica will down for a period

The log files will be in `/p4/1/logs`, so you can check for any errors from each script.

Recovery Procedures

There are three scenarios that require you to recover server data:

Metadata	Depotdata	Action required
lost or corrupt	Intact	Recover metadata as described below
Intact	lost or corrupt	Call Perforce Support
lost or corrupt	lost or corrupt	Recover metadata as described below. Recover the depotdata volume using your normal backup utilities.

Restoring the metadata from a backup also optimizes the database files.

Recovering a master server from a checkpoint and journal(s)

The checkpoint files are stored in the `/p4/instance/checkpoints` directory, and the most recent checkpoint is named `p4_instance.ckp.number.gz`. Recreating up-to-date database files requires the most recent checkpoint, from `/p4/instance/checkpoints` and the journal file from `/p4/instance/logs`.

To recover the server database manually, perform the following steps from the root directory of the server (`/p4/instance/root`).

1. Stop the Perforce Server by issuing the following command:

```
/p4/common/bin/p4master_run instance  \
    /p4/instance/bin/p4_instance admin stop
```

2. Delete the old database files in the `/p4/instance/root/save` directory
3. Move the live database files (`db.*`) to the `save` directory.
4. Use the following command to restore from the most recent checkpoint.

```
/p4/instance/bin/p4d_instance -r /p4/instance/root -jr -z  \
    /p4/instance/checkpoints/p4_instance.ckp.most_recent_#.gz
```

5. To replay the transactions that occurred after the checkpoint was created, issue the following command:

```
/p4/instance/bin/p4d_instance -r /p4/instance/root -jr \
    /p4/instance/logs/journal
```

6. Restart your Perforce server.

If the Perforce service starts without errors, delete the old database files from `/p4/instance/root/save`.

If problems are reported when you attempt to recover from the most recent checkpoint, try recovering from the preceding checkpoint and journal. If you are successful, replay the subsequent journal. If the journals are corrupted, contact Perforce Technical Support. For full details about backup and recovery, refer to the Perforce System Administrator's Guide.

Recovering a replica from a checkpoint

This is very similar to creating a replica in the first place as described above.

If you have been running the replica crontab commands as suggested, then you will have the latest checkpoints from the master already copied across to the replica.

See the steps in the script `weekly_sync_replica.sh` for details (note that it deletes the `state` and `rdb.lbr` files from the replica root directory so that the replica starts replicating from the start of a journal).

Remember to ensure you have logged the service user in to the master server (and that the ticket is stored in the correct location as described when setting up the replica).

Recovering from a tape backup

This section describes how to recover from a tape or other offline backup to a new server machine if the server machine fails. The tape backup for the server is made from the `depotdata` volume. The new server machine must have the same volume layout and user/group settings as the original server. In other words, the new server must be as identical as possible to the server that failed.

To recover from a tape backup, perform the following steps.

1. Recover the `depotdata` volume from your backup tape.
2. Create the `/p4` convenience directory on the OS volume.

   3. Create the directories `/metadata/p4/`*`instance`*`/root/save` and `/metadata/p4/`*`instance`*`/offline_db`.

4. Change ownership of these directories to the OS account that runs the Perforce processes.
5. Switch to the Perforce OS account, and create a link in the `/p4` directory to `/depotadata/p4/`*`instance`*.
6. Create a link in the `/p4` directory to `/depotdata/p4/common`.
7. As a super-user, reinstall and enable the `init.d` scripts
8. Find the last available checkpoint, under `/p4/`*`instance`*`/checkpoints`
9. Recover the latest checkpoint by running:

```
/p4/instance/bin/p4d_instance -r /p4/instance/root -jr -z last_ckp_file
```

10. Recover the checkpoint to the offline_db directory:

```
/p4/instance/bin/p4d_instance -r /p4/instance/offline_db -jr -z last_ckp_file
```

11. Reinstall the Perforce server license to the server `root` directory.
12. Start the perforce service by running `/p4/1/bin/p4d_1_init start`
13. Verify that the server instance is running.
14. Reinstall the server crontab or scheduled tasks.

15. Perform any other initial server machine configuration.
16. Verify the database and versioned files by running the p4verify`.sh` script. Note that files using the +k file type modifier might be reported as `BAD!` after being moved. Contact Perforce Technical Support for assistance in determining if these files are actually corrupt.

Failover to a replicated standby machine

See DR-Failover-Steps-Linux.docx

Server Maintenance

This section describes typical maintenance tasks and best practices for administering server machines. The directory `$SDP/Maintenance` contains scripts for several common maintenance tasks. A driver script called `maintenance` is available for Unix and Linux platforms. This script can be invoked from a *cron* utility weekly. Configure `maintenance.cfg` in the Maintenance directory for your site before attempting to run any of the scripts.

The user running the maintenance scripts must have administrative access to Perforce for most activities. Most of these scripts can be run from a client machine, but it is easiest to run them on the server via crontab.

Server upgrades

Upgrading a server instance in the SDP framework is a simple process involving a few steps.

- Download the new p4 and p4d executables for your OS from ftp.perforce.com and place them in `/p4/common/bin`
- Run:
  `/p4/common/bin/upgrade.sh` *`instance`*
- If you are running replicas, upgrade the replicas first, and then the master (outside -> in)

Database Modifications

Occasionally modifications are made to the Perforce database from one release to another. For example, server upgrades and some recovery procedures modify the database.

When upgrading the server, replaying a journal patch, or performing any activity that modifies the `db.*` files, you must restart the offline checkpoint process so that the files in the `offline_db` directory match the ones in the live server directory. The easiest way to restart the offline checkpoint process is to run the `live_checkpoint` script after modifying the `db.*` files, as follows:

```
/p4/common/bin/live_checkpoint.sh instance
```

This script makes a new checkpoint of the modifed database files in the live root directory, then recovers that checkpoint to the `offline_db` directory so that both directories are in sync. This script can also be used anytime to create a checkpoint of the live database.

This command must be run when an error occurs during offline checkpointing. It restarts the offline checkpoint process from the live database files to bring the offline copy back in sync. If the live checkpoint script fails, contact Perforce Consulting at consulting@perforce.com.

Listing inactive specifications

To list branch specifications, clients, labels and users that have been inactive for a specified number of weeks, run `accessdates.py`. This script generates four text files listing inactive specifications:

- `branches.txt`
- `clients.txt`
- `labels.txt`
- `users.txt`

Unloading and Reloading labels

To use the unload and reload commands for archiving clients and labels, you must first create an unload depot using the "p4 depot" command. Run:

```
p4master_run instance /p4/instance/bin/p4_instance depot unload
```

Set the type of the depot to unload and save the form. (This step has already been done if you ran the configure_new_server.sh script)

After the depot is created, you can use the following command to archive all the clients and labels that have not been accessed since the given date:

```
p4master_run instance /p4/instance/bin/p4_instance  \
     unload -f -L -z -a -d <date>
```

For example, to unload all clients and labels that haven't been access since Jan. 1, 2013, you would run:

```
p4master_run instance /p4/instance/bin/p4_instance  \
     unload -f -L -z -a -d 2013/01/01
```

Users can reload their own clients/labels using the reload command. They can run:

```
p4master_run instance /p4/instance/bin/p4_instance \
     reload -c <clientname>
```

or

```
p4master_run instance /p4/instance/bin/p4_instance reload -l <labelname>
```

As a super user, you can reload and unloaded item by adding the -f flag to the reload command as follows:

```
p4master_run instance /p4/instance/bin/p4_instance  \
     reload -f -c|l <specname>
```

In addition, you can avoid having to unload/reload labels by creating a trigger to set the autoreload option as the default on all new labels. That will cause the server to use the unload depot for storing the labels rather than storing them in db.label. This helps with performance of the server by not increasing the size of the database for label storage.

You can automate these tasks with `$SDP/Maintenance/unload_clients.py` and `$SDP/Maintenance/unload_labels.py`

Deleting users

To delete users, run `python p4deleteuser.py`, specifying the users to be deleted. The script deletes the users, any workspaces they own, and removes them from any groups they belong to.

To delete all users that have not accessed the server in the past 12 weeks, run `python delusers.py`. To change the number of weeks to a value other than 12, change the `weeks` variable in maintenance.cfg.

To remove a specified user from all groups it belongs to, run `python removeuserfromgroups.py`. Specify the user name or the name of a text file containing a list of users to be removed.

Listing users

To display a list of users that are in a group but do not have an account on the server, run `python checkusers.py`.

Group management

To duplicate a specified user's group entries on behalf of another user, run `python mirroraccess.py`. The script adds the target user to all groups that the source user belongs to. Invoke the script as follows:

```
python mirroraccess.py sourceuser targetuser
```

To add users to a group, run

```
python addusertogroup.py user group
```

where:

- *`user`* = user name or a file containing a list of user names, one per line.
- *`group`* = name of Perforce group to which the users are added.

Adding users

To add users to a server:

1. Create a text file, such as `users.csv`, containing the users to add, specifying one user name per line in this format:

   user,email,full name

2. If you are using LDAP/AD authentication, edit createusers.py and comment out this line:

   ```
   setpass.setpassword(user[0])
   ```

3. Run `python createusers.py users.csv`.

Email functions

To send email to all of your Perforce users:

1. Create a file called `message.txt` that contains the body of your message.
2. Run `email.bat or email.sh`, specifying the email subject in quotes.

To list the email addresses of your Perforce users, run `python make_email_list.py`.

Workspace management

The form-out trigger `$SDP/Server/Unix/p4/common/bin/triggers/SetWsOptions.py` contains default workspace options, such as `leaveunchanged` instead of `submitunchanged`.

To use the trigger, first copy it to `/p4/common/bin/triggers`

To enable the trigger, first modify the `OPTIONS` variable in the script, providing the set of desired options. Then insert an entry in the trigger table like the following:

```
setwsopts form-out client "python /p4/common/bin/triggers/SetWsOptions.py
  %formfile%"
```

The form-save trigger `$SDP/Server/Unix/p4/common/bin/triggers/PreventWsNonAscii.py` enforces the policy that no workspaces may contain non-ASCII characters.

To use the trigger, first copy it to `/p4/common/bin/triggers`

To enable the trigger, insert an entry in the trigger table like the following:

```
nowsascii form-save client "python /p4/common/bin/triggers/PreventWsNonAscii.py
  %formfile%"
```

Removing empty changelists

To delete empty pending changelists, run `python remove_empty_pending_changes.py`.

Maximizing Server Performance

The following sections provide some guidelines for maximizing the performance of the Perforce Server, using tools provided by the SDP. More information on this topic can be found in the System Administrator's Guide and in the Knowledge Base.

Optimizing the database files

The Perforce Server's database is composed of b-tree files. The server does not fully rebalance and compress them during normal operation. To optimize the files, you must checkpoint and restore the server. The weekly checkpoint script used as part of the normal server maintenance automates this task.

To minimize the size of back up files and maximize server performance, minimize the size of the `db.have` and `db.label` files. The scripts for Unloading and Reloading labels, and Deleting users, help achieve this goal. For best server performance, run these scripts weekly via /p4/sdp/Maintenance/maintenance

Proactive Performance Maintenance

This section describes some things that can be done to proactively to enhance scalability and maintain performance.

Limiting large requests

To prevent large requests from overwhelming the server, you can limit the amount of data and time allowed per query by setting the `maxresults`, `maxscanrows` and `maxlocktime` parameters to the lowest setting that does not interfere with normal daily activities. As a good starting point, set `maxscanrows` to `maxresults` * 3; set `maxresults` to slightly larger than the maximum number of files the users need to be able to sync to do their work; and set `maxlocktime` to 30000 milliseconds. These values must be adjusted up as the size of your server and the number of revisions of the files grow. To simplify administration, assign limits to groups rather than individual users.

To prevent users from inadvertently accessing large numbers of files, define their client view to be as narrow as possible, considering the requirements of their work. Similarly, limit users' access in the protections table to the smallest number of directories that are required for them to do their job.

Finally, keep triggers simple. Complex triggers increase load on the server.

Offloading remote syncs

For remote users who need to sync large numbers of files, Perforce offers a proxy server. P4P, the Perforce Proxy, is run on a machine that is on the remote users' local network. The Perforce Proxy caches file revisions, serving them to the remote users and diverting that load from the main server.

P4P is included in the Windows installer. To launch P4P on Unix machines, copy the `/p4/common/etc/init.d/p4p_1_init` script to `/p4/1/bin/p4p_1_init`. Then review and customize the script to specify your server volume names and directories.

P4P does not require special hardware but it can be quite CPU intensive if it is working with binary files, which are CPU-intensive to attempt to compress. It doesn't need to be backed up. If the P4P instance isn't working, users can switch their port back to the main server and continue working until the instance of P4P is fixed.

Tools and Scripts

This section describes the various scripts and files provided as part of the SDP package. To run main scripts, the machine must have Python 2.7, and a few scripts require Perl 5. The Maintenance scripts can be run from the server machine or from client machines.

The following various scripts.

Core Scripts

The core SDP scripts are those related to checkpoints and other scheduled operations, and all run from `/p4/common/bin`.

p4_vars

Defines the environment variables required by the Perforce server. This script uses a specified instance number as a basis for setting environment variables. It will look for and open the respective `p4_<instance>.vars` file (see next section).

This script also sets server logging options and configurables.

Location: `/p4/common/bin`

p4_<instance>.vars

Defines the environment variables for a specific instance, including P4PORT etc.

Location: `/p4/common/config`

p4master_run

This is the wrapper script to other SDP scripts. This ensures that the shell environment is loaded from `p4_vars`. It provides a '-c' flag for silent operation, used in many `crontab` so that email is sent from the scripts themselves.

Location: `/p4/common/bin`

recreate_offline_db

Recovers the `offline_db` database from the latest checkpoint and replays any journals since then. If you have a problem with the offline database then it is worth running this script first before running live_checkpoint, as the latter will stop the server while it is running which can take hours.

Run this script if an error occurs while replaying a journal during weekly or daily checkpoint process.

Location: `/p4/common/bin`

live_checkpoint

Stops the server, creates a checkpoint from the live database files, recovers the `offline_db` database from that checkpoint to rebalance and compress the files, then recovers the checkpoint in the `offline_db` directory to ensure that the database files are optimized.

Run this script when creating the server and if an error occurs while replaying a journal during the off-line checkpoint process.

Location: `/p4/common/bin`

daily_checkpoint

This script is configured to run six days a week using crontab or the Windows scheduler. The script truncates the journal, replays it into the `offline_db` directory, creates a new checkpoint from the resulting database files, then recreates the `offline_db` directory from the new checkpoint.

This procedure rebalances and compresses the database files in the `offline_db` directory, which are rotated into the live database directory once a week by the `weekly_checkpoint` script.

Location: `/p4/common/bin`

recreate_db_checkpoint

Performs the weekly checkpoint process. This script stops your server for a few minutes to rotate your database files with those in the `offline_db` directory. Specifically, this script does the following:

1. Stops the server.
2. Truncates the journal.
3. Replays the journal to `offline_db`.
4. Deletes last week's database files from `save`.
5. Moves the database files from the server root to the `save` directory.
6. Moves the db files from the `offline_db` directory to the root directory
7. Restarts the server
8. Creates a checkpoint from the database files in the save directory.
9. Recreates the database files in `offline_db` from the new checkpoint.

Location: `/p4/common/bin`

p4verify

Verifies the integrity of the depot files. This script is run by `crontab`.

Location: `/p4/common/bin`

p4review.py

Sends out email containing the change descriptions to users who are configured as reviewers for affected files (done by setting the `Reviews:` field in the user specification). This script is a version of the `p4review.py` script that is available on the Perforce Web site, but has been modified to use the server instance number. It relies on a configuration file in `/p4/common/config`, called `p4_<instance>.p4review.cfg`. On Windows, a driver called `run_p4review.cmd`, located in the same directory, allows you to run the review daemon through the Windows scheduler.

Location: `/p4/common/bin`

p4login

Executes a `p4 login` command, using the password configured in `mkdirs.bat` and stored in a text file.

Location: `/p4/common/bin`

p4d_*instance*_init

Starts the Perforce server.

This script sources `/p4/common/bin/p4_vars`, then `/p4/common/bin/p4d_base`.

Note: In clustered environments, put this script in the `/p4/`*`instance`*`/bin` directory and configure your cluster software to launch it from this location.

Location: `/p4/`*`instance`*`/bin` with a symlink to it from `/etc/init.d` (or a copy in `/etc/init.d` in a clustered environments). Templates for init scripts for other Perforce server products exist in `p4/common/etc/init.d`

More Server Scripts

These scripts are helpful components of the SDP that run on the server, but are not included in the default crontab schedules.

upgrade.sh

Runs a typical upgrade process, once new `p4` and `p4d` binaries are available in `/p4/common/bin`.

Location: `/p4/common/bin`

p4.crontab

Contains `crontab` entries to run the server maintenance scripts. The `p4.crontab.solaris` script is for Solaris.

Location: `/p4/sdp/Server/Unix/p4/common/etc/cron.d`

Maintenance Scripts

There are many useful scripts in /p4/sdp/Maintenance that are not set up to run automatically as part of the SDP installation. The scripts provide maintenance tools, and various scripts to provide reports or a useful one off activity you may need done. Each script has comments at the top indicating what it does and how to run it. Most of the scripts can be run on the server or client machines.

Other Files

The following table describes other files in the SDP distribution. These files are usually not invoked directly by you; rather, they are invoked by higher-level scripts.

File	Location	Remarks
dummy_ip.txt	$SDP/Server/config	Instructions for using a license on more than one machine. Typically used to enable a standby server. Contact Perforce Licensing before using.
backup_functions.sh	/p4/common/bin	Unix/Linux only. Utilities for maintenance scripts.
p4admin_verify_client.bat	/p4/common/bin	Unix/Linux only. Used by p4verify.sh.
p4d_base	/p4/common/bin	Unix/Linux only. Template for Unix/Linux init.d scripts.
template.(pl|sh)	/p4/common/bin	Sample script templates for Bash and Perl scripts.
mirror_ldap_groups.pl	/p4/common/bin	Script to mirror selected groups from an LDAP server (e.g. Active Directory).
Perl Modules (*.pm files)	/p4/common/lib	Modules used by some Perl scripts.
change.txt	$SDP/Maintenance	Template for new pending changelist.

Appendix A – Directory Structure Configuration Script for Linux/Unix

This script describes the steps performed by the `mkdirs.sh` script on Linux/Unix platforms. Please review this appendix carefully before running these steps manually. Assuming the three-volume configuration described in the Volume Layout and Hardware section are used, the following directories are created. The following examples are illustrated with "1" as the server instance number.

Directory	*Remarks*
`/p4`	Must be under `/` on the OS volume
`/depotdata/p4/1/bin` `/depotdata/p4/1/depots`	Files in here are generated by the `mkdirs.sh` script.
`/depotdata/p4/1/tmp`	
`/depotdata/p4/common/config`	Contains `p4_<instance>.vars` file.
`/depotdata/p4/common/bin`	Files from `$SDP/Server/Unix/p4/common/bin`.
`/depotdata/p4/common/etc` `/logs/p4/1/logs/old`	Contains `init.d` and `cron.d`.
`/metadata/p4/1/offline_db`	Contains offline copy of main server databases.
`/metadata/p4/1/root/save`	Used only during recreate_db_checkpoint.sh for extra redundancy.

Next, `mkdirs.sh` creates the following symlinks in the `/depotdata/p4/1` directory:

Link source	*Link target*	*Command*
`/metadata/p4/1/root`	`/p4/1/root`	`ln -s /metadata/p4/1/root`
`/metadata/p4/1/offline_db`	`/p4/1/offline_db`	`ln -s /metadata/p4/1/offline_db`
`/logs/p4/1/logs`	`/p4/1/logs`	`ln -s /logs/p4/1/logs`

Then these symlinks are created in the `/p4` directory:

Link source	*Link target*	*Command*
`/depotdata/p4/1`	`/p4/1`	`ln -s /depotdata/p4/1`
`/depotdata/p4/common`	`/p4/common`	`ln -s /depotdata/p4/common`

Next, `mkdirs.sh` renames the Perforce binaries to include version and build number, and then creates appropriate symlinks.

The structure is shown in this example, illustrating values for two instances, with instance #1 using Perforce 2013.1 and instance #2 using 2013.2. Files are shown in red and symlinks in green.

In `/p4/common/bin`:

`p4_2013.1_bin` → `p4_2013.1.685046`

`p4d_2013.1_bin` → `p4d_2013.1.685046`

`p4_2013.2_bin` → `p4_2013.2.700949`

`p4d_2013.2_bin` → `p4d_2013.2.700949`

`p4_1_bin` → `p4_2013.1_bin`

`p4d_1_bin` → `p4d_2013.1_bin`

`p4_2_bin` → `p4_2013.2_bin`

`p4d_2_bin` → `p4d_2013.2_bin`

In `/p4/1/bin`:

`p4_1` → `/p4/common/bin/p4_1_bin`

`p4d_1` → `/p4/common/bin/p4d_1_bin`

In `/p4/2/bin`:

`p4_2` → `/p4/common/bin/p4_2`

`p4d_2` → `/p4/common/bin/p4d_2`

Appendix B – Frequently Asked Questions/Troubleshooting

This appendix lists common questions and problems encountered by SDP users. Do not hesitate to contact consulting@perforce.com if additional assistance is required.

Journal out of sequence

This error is encountered when the offline and live databases are no longer in sync, and will cause the offline checkpoint process to fail. Because the scripts will replay all outstanding journals, this error is much less likely to occur. This error can be fixed by running the `live_checkpoint.sh` script, as described in Server upgrades. Alternatively, if you know that the checkpoints created from previous runs of `daily_checkpoint.sh` are correct, then restore the `offline_db` from the last known good checkpoint.

Unexpected end of file in replica daily sync

Check the start time and duration of the `daily_checkpoint.sh` cron job on the master. If this overlaps with the start time of the `sync_replica.sh` cron job on a replica, a truncated checkpoint may be rsync'd to the replica and replaying this will result in an error.

Adjust the replica's cronjob to start later to resolve this.

Default cron job times, as installed by the SDP are initial estimates, and should be adjusted to suit your production environment.