Perforce P4D Sample Storage Setup - LVM

Perforce Professional Services

Version v2025.1, 2025-10-28

Table of Contents

Preface

This document illustrates a basic set of commands to setup data storage using Linux Volume Manager (LVM).

LVM is the preferred method of setup for Perforce volumes. Both Physical servers and Cloud/VM servers benefit from features of LVM covered in this document.

The goal in these examples is to configure three LVM storage volumes separate from the OS volume on a server destined to become a Helix Core server. At the start of this procedure, empty volumes with no data are formatted.

Console sessions

When console sessions are shown in the text:

Normal user shown with $ prompt

```
$ ls -lR
```

Root user shown with # prompt

```
# systemctl daemon-reload
```

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.

Chapter 1. Why use LVM ?

Some of the benefits of using LVM over standard disk partitions are:

- Expansion of physical volumes is not limited to the size remaining on the disk. Additional disks can be added to the Volume Group as needed for expansion.
- Volume name/UUID are embedded into the media. This helps identify snapshots and cloned volumes used for migrations.
- LVM avoids need for UUID's in `/etc/fstab`, as volumes are always assigned unique device names.

LVM also supports a volume snapshot and revert feature. However, LVM snapshots incur a significant performance penalty due to the Copy-On-Write (COW) method utilized. For this reason LVM snapshots are not recommended for use on Perforce volumes.

Chapter 2. Sample Storage Setup for SDP - Mounts and Storage

2.1. Starting State

This procedure assumes the following start state:

- Destination Server has a basic install of Ubuntu 24.04. (RHEL/Rocky9 TBD) with following installed:
  - LVM2, if missing run `apt install lvm2`
  - XFS, if missing run `apt install xfsprogs`
- Three separate storage volumes are attached Server / VM. (in addition to the OS root volume), intended to become `/p4depots`, `/p4logs`, and `/p4metadta`:
  - `/p4depots` - Give this as much space as you think you'll need. This is highly variable. You may want 30G source code projects, or 1T or more for virtual production or game development. Size can be increased easily and non-disruptively later, so you don't need to overprovision (and overpay) for storage.
  - `/p4metadata` - Use 25G to start. This needs to hold (2) copies of the database.
  - `/p4logs` - Use 20G to start. Typically low usage, but large enough to contain large journals during any occasional purge/obliterate operations.
- Volumes may be Physical disks, Direct-Attached-Storage (DAS), EBS volumes on AWS, VMFS disks on a VM or other block storage.

There is no easy method of matching the device being attached to the assigned kernel name such as: `nvme1n1` or `nvme1n2`. However if we choose disks of different sizes they can easily be matched up with kernal assigned names.

For instance, in the example above, the 1000G volume `nvme2n1` is for `/p4depots`, the 25G volume `nvme1n1` is for `/p4metadata` and the 20G volume is for `/p4logs`

- See Also: https://www.reddit.com/r/linuxadmin/comments/8cg1t4/benefits_of_lvm/

2.2. Storage Formatting and Mounting Procedure

First, become root with the `sudo su -` command, and then list the mounted storage before we mount the new volumes:

```
$ sudo su -

# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        1.8G     0  1.8G   0% /dev
tmpfs           1.8G     0  1.8G   0% /dev/shm
```

```
tmpfs           1.8G   17M  1.8G   1% /run
tmpfs           1.8G     0  1.8G   0% /sys/fs/cgroup
/dev/nvme0n1p1   10G  1.6G  8.5G  16% /
tmpfs           356M     0  356M   0% /run/user/0
```

You don't yet see the newly attached volumes, as they're not yet mounted. But you can list them with `lsblk`:

```
# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
nvme0n1     259:0    0   10G  0 disk
 └─nvme0n1p1 259:1    0   10G  0 part /
nvme1n1     259:2    0   25G  0 disk
nvme2n1     259:3    0   20G  0 disk
nvme3n1     259:4    0  100G  0 disk
```

2.2.1. LVM

The Linux Volume Manager (LVM) is used in place of old style disk partitions, and is much more flexible. Volumes may be expanded while the system is running, and volumes may span more than one disk.

Ubuntu 24.04 installer includes LVM.

- see: https://documentation.ubuntu.com/server/explanation/storage/about-lvm/#

This document provides a summary of only the basic commands to setup LVM following best practices.

LVM components

PV's

Physical Volumes. `PV` volumes provide storage to their assigned `VG` Marking a disk as a `PV` makes it available to be assigned to a `VG`

VG's

Volume Groups. `VG` are comprised of one or more `PVs` that supply storage to the `VG` The VG storage is comprised of all disks attached to it and is not limited to the size of any single disk.

LV's

Logical Volumes `LVs` are composed by allocating space from a `VG` Disks with different performance characteristics should be separated into different VG's so they may be assigned to appropriate `LVs`

LVM marks PV's, VG's and LV's with unique `UUIDs` and includes the name of the `VG`

or LV on the media.

2.2.2. Physical Volumes(PVs)

Normally the entire disk is marked as a PV, however in the past folks have also marked disk **partitions** as PVs.

The reason full disks are used without parititions , is that any later disk expansion can then avoid expanding that partition.

First mark each of the 3 new blank volumes as an PV.

```
[root@~]# pvcreate /dev/nvme1n1
Writing physical volume data to disk "/dev/nvme1n1"
Physical volume "/dev/nvme1n1" successfully created.

[root@~]# pvcreate /dev/nvme2n1
Writing physical volume data to disk "/dev/nvme2n1"
Physical volume "/dev/nvme2n1" successfully created.

[root@~]# pvcreate /dev/nvme3n1
Writing physical volume data to disk "/dev/nvme3n1"
Physical volume "/dev/nvme3n1" successfully created.
```

Display PV's

You can display the new PV's in short or long mode.

Short

```
# pvs
PV            VG   Fmt     Attr   PSize  PFree
/dev/nvme1n1       lvm2    ---   <25G    <25G
/dev/nvme2n1       lvm2    ---   <20G    <20G
/dev/nvme3n1       lvm2    ---   <1000G  <1000G
```

Long

```
# pvdisplay
... < many lines showing Name, size, UUID >
```

2.2.3. Volume Groups (VGs)

Next create VGs for each of the three types of storage.

We will name the 3 VGs as:

- vg_p4metadata

- vg_p4logs
- vg_p4depots

Create the new 'VGs' and assign the appropriate PV with:

```
# vgcreate vg_p4metadata /dev/nvme1n1      ①
Volume group "vg-p4metadata" successfully created

# vgcreate vg_p4logs /dev/nvme2n1          ②
Volume group "vg-p4logs" successfully created

# vgcreate vg_p4depots /dev/nvme3n1        ③
Volume group "vg-p4depots" successfully created
```

① p4metdata was determined to use nvme1n1 by matching disk size of 25GB

② p4logs was determined to use nvme2n1 by matching disk size of 20GB

③ p4depots was determined to use nvme3n1 by matching disk size of 1000GB

Display VG's

You can display the new VG's in short or long mode.

Short

```
# vgs
VG              #PV  #LV  #SN  Attr   VSize   VFree
vg_p4metadata   x    0     0  wz--n-  <250G
vg_p4logs       x    0     0  wz--n-  <200G
vg_p4depots     x    0     0  wz--n-  <1000G
```

Long

```
# vgdisplay
... < many lines showing Name, Size, PV UUID >
```

2.2.4. Logical Volumes (LVs)

Finally create logical volumes LVs inside each VG, using all of the available storage.

```
# lvcreate -n lv_p4metadata -l 100%FREE  vg_p4metadata
Logical volume "lv_p4metadata" created.

# lvcreate -n lv_p4logs -l 100%FREE  vg_p4logs
Logical volume "lv_p4logs" created.

# lvcreate -n lv_p4depots -l 100%FREE  vg_p4depots
```

```
Logical volume "lv_p4depots" created.
```

Display LV's

You can display the new LV's in short or long mode.

Short

```
# lvs
LV             VG              Attr     LSize  Pool  Origin  Data% Meta%
lv_p4metadata  vg_p4metadata   -wi-a-    25G
lv_p4logs      vg_p4logs       -wi-a- 0  20G
lv_p4depots    vg_p4depots     -wi-a- 0  1000G
```

Long

```
# lvdisplay
... < many lines showing Name, Size, Status, LV UUID >
```

2.2.5. Formating LVM volumes

LVM **physical block devices** are mapped to Linux **virtual block devices** by the Device Mapper.

The device mapper will automatically create devices for the above LVM disks as follows:

- /dev/mapper/vg_p4metadata-lv_p4metadata
- /dev/mapper/vg_p4logs-lv_p4logs
- /dev/mapper/vg_p4depots-lv_p4depots

The explicit LVM device names assigned makes formatting these volumes much safer.

Next, Format these new LVs as XFS.

```
# mkfs.xfs /dev/mapper/vg_p4metadata-lv_p4metadata
meta-data=/dev/mapper/vg_p4metadata-lv_p4metadata          isize=512    agcount=16,
agsize=1638400 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1
  data   =                       bsize=4096   blocks=26214400, imaxpct=25
         =                       sunit=1      swidth=1 blks
  naming =version 2              bsize=4096   ascii-ci=0, ftype=1
  log    =internal log           bsize=4096   blocks=12800, version=2
         =                       sectsz=512   sunit=1 blks, lazy-count=1
  realtime =none                 extsz=4096   blocks=0, rtextents=0
```

```
# mkfs.xfs /dev/mapper/vg_p4logs-lv_p4logs
meta-data=/dev/mapper/vg_p4logs-lv_p4logs           isize=512    agcount=16,
agsize=1638400 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1
  data   =                       bsize=4096   blocks=26214400, imaxpct=25
         =                       sunit=1      swidth=1 blks
  naming =version 2              bsize=4096   ascii-ci=0, ftype=1
  log    =internal log           bsize=4096   blocks=12800, version=2
         =                       sectsz=512   sunit=1 blks, lazy-count=1
  realtime =none                 extsz=4096   blocks=0, rtextents=0

# mkfs.xfs /dev/mapper/vg_p4depots-lv_p4depots
meta-data=/dev/mapper/vg_p4depots-lv_p4depots           isize=512    agcount=16,
agsize=1638400 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1
  data   =                       bsize=4096   blocks=26214400, imaxpct=25
         =                       sunit=1      swidth=1 blks
  naming =version 2              bsize=4096   ascii-ci=0, ftype=1
  log    =internal log           bsize=4096   blocks=12800, version=2
         =                       sectsz=512   sunit=1 blks, lazy-count=1
  realtime =none                 extsz=4096   blocks=0, rtextents=0
```

Formatting the wrong device may destroy data !

2.2.6. Update /etc/fstab mount table

Make a backup copy of the `/etc/fstab` file, and then modify that file to create new volumes.

```
# cd /etc

# ls -l fstab*
-rw-r--r--. 1 root root 394 Nov 15 04:43 fstab

# cp -p fstab fstab.bak.2022-03-08
```

Edit fstab

Next add entries to `/etc/fstab` for the mount points.

```
# backup fstab
cp -f /etc/fstab /etc/fstab.bak
# append to fstab
# echo "/dev/mapper/vg_p4metadata-lv_p4metadata /p4metadata xfs defaults 0 0" >>
/etc/fstab
```

```
# echo "/dev/mapper/vg_p4logs-lv_p4logs /p4logs xfs defaults 0 0" >> /etc/fstab
# echo "/dev/mapper/vg_p4depots-lv_p4depots /p4depots xfs defaults 0 0" >> /etc/fstab
```

The previous method of mounting Volumes in `/etc/fstab` by their `UUID` values is not used with LVM. LVM uses UUID's internally, and the Linux kernel Device mapper always maps UUID's to the same device.

Update mounts

After changing `/etc/fstab` run this command up update mount info.

```
# systemctl daemon-reload
```

Run the `systemctl daemon-reload` command after each change to `/etc/fstab` or bad things might happen. It is also a good idea to mount new filesystems the first time with `mount -a` which should report any errors in `/etc/fstab` before rebooting the machine.

Proceed with creating empty directories that will be the "mount points" for the volume to be mounted.

```
# mkdir /p4depots /p4logs /p4metadata
```

Next, use the `mount -a` command. This will now assocate the mount points you just created with the storage device information that is now in that `/etc/fstab` file, and mount the volumes.

```
# mount -a
```

Then see if they are mounted. This is what victory looks like, with the `/p4*` volumes all mounted with desired sizes:

```
# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        1.8G     0  1.8G   0% /dev
tmpfs           1.8G     0  1.8G   0% /dev/shm
tmpfs           1.8G   17M  1.8G   1% /run
tmpfs           1.8G     0  1.8G   0% /sys/fs/cgroup
/dev/nvme0n1p1   10G  1.6G  8.5G  16% /
tmpfs           356M     0  356M   0% /run/user/0
/dev/mapper/vg_p4metadata-lv_p4metadata  120G  890M  120G   1% /p4metadata
/dev/mapper/vg_p4logs-lv_p4logs          100G  747M  100G   1% /p4logs
/dev/mapper/vg_p4depots-lv_p4depots      500G  3.6G  497G   1% /p4depots
```

At this point, you are ready to install the Server Deployment Package (SDP) software.

Chapter 3. Volume Expansion

An LVM partition can usually be expanded while the system is running without any outage.

The procedure involves the following steps:

1. Expand the associated VG by one of following
    a. Expand the underlying PV disk (such as a virtual disk)
        i. Rescan the storage buss
        ii. Resize the PV
        iii. Extend the LV -or-
    b. Add additonal disks to the VG (typically physical disks)
        i. Rescan the storage buss
        ii. Extend the VG
        iii. Extend the LV

3.1. Expand VG by expanding underlying disk

In a virtual environment it is typically easy to expand the size of the underlying disk.

Even though this procedure can be run without downtime, to be safe, ensure you have a recent backup of the system.

1. Expand the virtual block storage device, by changing its size on AWS, VMware, or other environment.
2. Rescan the VM's **scsi-buss** to pick up the change

Note replace **nvme1n1** with your block device

```
$ sudo su
# echo 1 > /sys/block/*nvme1n1*/device/rescan_controller
# exit

//  Alternate procedure
$ sudo apt install scsi-tools
$ sudo rescan-scsi-bus -s
```

3. Resize the PV to use the new expaned size

```
### Note replace *nvme1n1* with your block device
$ sudo pvresize /dev/*nvme1n1*
```

4. Resize the LV to use the new storage

Note replace `/dev/mapper/vg_p4depots-lv_p4depot` with your device Run one of the following

```
$ sudo lvextend -l +100%FREE /dev/mapper/vg_p4depots-lv_p4depots
$ sudo lvextend -l +100%FREE /dev/mapper/vg_p4logs-lv_p4logs
$ sudo lvextend -l +100%FREE /dev/mapper/vg_p4metadata-lv_p4metadata
```

5. Grow the Filesystem

Note replace `/dev/mapper/vg_p4depots-lv_p4depots` with your device

```
# xfs_growfs /dev/mapper/vg_p4depots-lv_p4depots     # Full Size
```

3.2. Expand VG by adding more disks

Even though this procedure can be run without downtime, to be safe, ensure you have a recent backup of the system.

1. Expand the VG by adding additional disks to the VG.
2. Hot-pluggable server disks are required to add without downtime. After adding the disk, rescan the VM's **scsi-buss** to pick up the change

To determine the exact device name of the added disk, look at the kernel ring buffer for the expected device name immediatly after adding the disk.

```
$ sudo dmesg | grep nvme     # If device is expected to be nvme*
```

Note replace **nvme5n1** with your block device shown with `dmesg` sudo dmesg # Look for new block device signature, e.g `nvme5n1`

```
$ sudo su
# echo 1 > /sys/block/*nvme5n1*/device/rescan_controller
# exit

#Alternate procedure
$ sudo apt install scsi-tools
$ sudo rescan-scsi-bus
```

3. Mark new Disk as a PV

Note replace **nvme5n1** with your new device

```
$ sudo pvcreate /dev/*nvm5n1*
```

4. Extend the VG

Replace **nvme5n1** with your new disk

```
$ sudo vgextend vg_p4metadata-lv_p4metadata /dev/*nvme5n1*
  Volume group "vg_p4metadata" sucessfully extended
```

5. Extend the LV

Replace the LV with your desired LV

```
$ sudo lvextend -l +100%FREE /dev/mapper/vg_p4metadata-lv_p4metadata
  Size of logical volume "lv_p4metadata" changed fromm 205GB to 300GB
  Logical volume "lv_p4metadata" sucessfully resized.
```

3.3. LVM Storage setup script

For reference if device names are known in advance, the LVM setup procedure may be scripted as follows:

```
#!/bin/bash

# Devices to format  ( root on nvme1n1 )
DEV_META="/dev/nvme1n1"
DEV_LOGS="/dev/nvme2n1"
DEV_DEPOTS="/dev/nvme3n1"

# Mount points
MP_META="/p4metadata"
MP_LOGS="/p4logs"
MP_DEPOTS="/p4depots"

# VGs
VG_META=vg_p4metadata
VG_LOGS=vg_p4logs
VG_DEPOTS=vg_p4depots

# LVs
LV_META=lv_p4metadata
LV_LOGS=lv_p4logs
LV_DEPOTS=lv_p4depots

# Vols
VOL_META=/dev/mapper/$VG_META-$LV_META
VOL_LOGS=/dev/mapper/$VG_LOGS-$LV_LOGS
VOL_DEPOTS=/dev/mapper/$VG_DEPOTS-$LV_DEPOTS
```

```
# Backup /etc/fstab
cp /etc/fstab "/etc/fstab.bak.$(date +%F)"
echo "Creating PVs"

# If device unused, create PV's
for dev in $DEV_META $DEV_LOGS $DEV_DEPOTS; do
    if blkid "$dev" &> /dev/null; then
        echo .
        #echo "Warning: $dev already has a file system. Skipping pvcreate"
    else
        echo "Creating PV on $dev"
        pvcreate "$dev"
    fi
done

echo "Creating VGs"

# Create VG's, if VG doesn't exist
if ! vgdisplay -t $VG_META >& /dev/null ; then
   vgcreate $VG_META $DEV_META
fi

if ! vgdisplay -t $VG_LOGS >& /dev/null ; then
   vgcreate $VG_LOGS $DEV_LOGS
fi

if ! vgdisplay -t $VG_DEPOTS >& /dev/null ; then
   vgcreate $VG_DEPOTS $DEV_DEPOTS
fi

echo "Creating LV's"

#Create LV's, these generate warnings if already existing
if ! lvdisplay -t $VG_META >& /dev/null; then
   lvcreate -n $LV_META   -l 100%FREE $VG_META
fi

if ! lvdisplay -t $VG_LOGS >& /dev/null; then
   lvcreate -n $LV_LOGS   -l 100%FREE $VG_LOGS
fi

if ! lvdisplay -t $VG_DEPOTS >& /dev/null; then
   lvcreate -n $LV_DEPOTS -l 100%FREE $VG_DEPOTS
fi

echo "Formatting volumes"

# Format volumes
# If device has no filesystem, format as XFS
for dev in $VOL_META $VOL_LOGS $VOL_DEPOTS; do
    echo "$dev"
```

```
    if blkid -o value -s TYPE "$dev" &> /dev/null; then
        echo "Warning: $dev already has a file system. Skipping Formatting"
    else
        echo "Formatting $dev as XFS"
        #mkfs.xfs "$dev"
        if mkfs.xfs "$dev"; then
           #if [ $? -eq 0 ]; then
           echo "Sucessful format of $dev with XFS"
        else
           echo "Error: Failed to format $dev with XFS"
           exit 1
        fi
    fi
done

# Create mount points if they don't exist
[ ! -d "$MP_META" ] && mkdir -p "$MP_META"
[ ! -d "$MP_LOGS" ] && mkdir -p "$MP_LOGS"
[ ! -d "$MP_DEPOTS" ] && mkdir -p "$MP_DEPOTS"

echo "Updating /etc/fstab"

# Add entries to /etc/fstab, if not already there.
if ! grep ${VG_META} /etc/fstab; then
  {
    echo "/dev/mapper/${VG_META}-${LV_META}  ${MP_META} xfs defaults 0 0"
    echo "/dev/mapper/${VG_LOGS}-${LV_LOGS}  ${MP_LOGS} xfs defaults 0 0"
    echo "/dev/mapper/${VG_DEPOTS}-${LV_DEPOTS}  ${MP_DEPOTS} xfs defaults 0 0"
  } >> /etc/fstab
fi

# Reload systemd daemon
echo "Reloading systemd to apply changes to /etc/fstab."
systemctl daemon-reload

# Mount all file systems
mount -a

echo "Verify mountpoints"
# Verify mounts
mountpoint "$MP_DEPOTS" || echo "Error: $MP_DEPOTS is not mounted."
mountpoint "$MP_LOGS" || echo "Error: $MP_LOGS is not mounted."
mountpoint "$MP_META" || echo "Error: $MP_METADATA is not mounted."

echo "Display space on mountpoints"
# Display current mounts
df -h | grep mapper
```

Chapter 4. Notes and References

A few possible issues that may come up are:

1. Mounting duplicate named Disks
    - If you clone a disk and mount it to the same machine under a different mount point you will get an error because the UUID's and LVM assigned name are the same.
    - https://unix.stackexchange.com/questions/495669/how-to-mount-lvm-partitions-with-duplicate-names
2. Snapshot and revert before dangerous ops.
    - https://digitalcave.ca/resources/computer/lvm-snapshots.jsp
    - https://askubuntu.com/questions/424225/setting-up-lvm-snapshot-as-a-backup-restore-point-in-ubuntu
    - https://www.percona.com/blog/disaster-lvm-performance-in-snapshot-mode/