Installing Void Linux :: Jake@Linux

Sören Weber

Void Linux Install with BtrFS and encryption

So you want to install Void Linux, well you've come to the right place, so without further ado, let's get started.

Section 1 - Select and Prepare the Device

!! NOTE: !! ***-ALL COMMANDS IN THIS TUTORIAL ARE RUN AS THE ROOT USER, IF YOU ARE NOT ROOT, BE SURE TO EITHER USE SUDO OR CHANGE TO ROOT***

This section will cover partitioning your drive and adding encryption. There are many tools avalable to partition your drive, tools like cfdisk, parted, gparted, gdisk, etc, we will be using fdisk. As for encryption, we will be using luks1 since as of the writing of this, grub does not fully support luks2.

Step 1 - Partition

From the command line launch fdisk and point it to the device of your choosing, for this example I will use /dev/sdX. once fdisk is launched with the correct device, we need to create 2 parititons, the following are the selections needed:

```
fdisk /dev/sdX

Create GPT partition table
Command (m for help): g

Create EFI System Partition (ESP)
Command (m for help): n
Partition number (1-128, default 1): (enter for default)
First sector (2048-500118158, default 2048): (enter for default)
Last sector, +/-sectors or +/-size{K,M,G,T,P}...): +200M (choose 128M to 1G)
Do you want to remove the signature? [Y]es/[N]o: y
Command (m for help): t
Partition type or alias (type L to list all): 1

Create the root partition
Command (m for help): n
Partition number (2-128, default 2): (enter for default)
First sector (2099200-500118158, default 2099200): (enter for default)
Last sector, +/-sectors or +/-size{K,M,G,T,P}...): (enter for default)

Write changes to the disk
Command (m for help): w
```

Step 2 - Encrypt

Now that the device has been partitioned, we are ready to encrypt the root volume.

Again, since grub does not fully support luks2 we will be using luks1 using the following commands.

Encrypt the root volume with the following command:

```
cryptsetup luksFormat --type luks1 -y /dev/sdX2
```

It will warn you that this action will overwrite anything on the selected disk, type YES (all caps) to accept, then it will ask you to create and verify a password.

Now the the root volume is encrypted it is locked and cannot be edited, we need to open the root volume and give it

a name with the following command: (I am calling it cryptvoid, you can name it what ever you would like)

```
cryptsetup open /dev/sdX2 cryptvoid
```

you will be prompted for the password you just created and once verified, the encrypted root volume will be open.

Step 3 - Format Partitions

Now we are ready to format the paritions, we will make partition 1 the efi partition and install a fat filesystem and parition 2 will be the root partition with BtrFS.

format the efi parititon:

```
mkfs.fat -F32 -n EFI /dev/sdX1
```

format the root partition:

```
mkfs.btrfs -L Void /dev/mapper/cryptvoid
```

Step 4 - Create BtrFS Subvolumes

Now that we have the disk partitioned and the file systems created, now we need to create the subvolumes on the BtrFS volume.

First we will create a variable to store all the options we want to use, this is not completely necessary but will save on a lot of redundant typing

```
BTRFS_OPTS="rw,noatime,compress=zstd,discard=async"
```

- rw = read/write permissions
- noatime = no access time, prevents sys from updating access timestamp every time a file is accessed
- compress = type of compression
    - zstd = using Zstandard compression algorithm
- discard = improve efficiency of solid state drives by allowing them to reclaim unused space
    - async = operation will be performed asyncronously

Now we will mount the top level subvolume.

```
mount -o $BTRFS_OPTS /dev/mapper/cryptvoid /mnt
```

Now that we have the top level subvolume mounted, it is time to create the subvolumes for root, home, and snapshots

Root subvolume

```
btrfs su cr /mnt/@
```

Home subvolume

```
btrfs su cr /mnt/@home
```

Snapshots subvolume

```
btrfs su cr /mnt/@snapshots
```

Unmount cryptvoid

```
umount /mnt
```

The last step of prep before we start the base install is to mount all the subvolumes and the EFI partition

Mount the root subvolume

```
mount -o $BTRFS_OPTS,subvol=@ /dev/mapper/cryptvoid /mnt
```

We now have our encrypted volume mounted and we need to create mountpoints for home, root, and snapshots

```
mkdir /mnt/{efi,home,.snapshots}
```

Mount the home subvolume

```
mount -o $BTRFS_OPTS,subvol=@home /dev/mapper/cryptvoid /mnt/home
```

Mount snapshots subvolume

```
mount -o $BTRFS_OPTS,subvol=@snapshots /dev/mapper/cryptvoid /mnt/.snapshots
```

While a key function and benefit of BtrFS is snapshots, there are certain directories that don't need to be included in the snapshots, we will create some nested subvolumes for these specific directories

```
mkdir -p /mnt/var/cache
btrfs su cr /mnt/var/cache/xbps
btrfs su cr /mnt/var/tmp
btrfs su cr /mnt/srv
```

Mount the EFI system parition

```
mount -o rw,noatime /dev/sdX1 /mnt/efi
```

Check mountpoints and verify they are correct using one of the following 2 commands

```
df -h
```

or

```
lsblk
```

Section 2 - Installation

With the drive now partitioned, the file system created, and all volumes and subvolumes mounted, we can begin the base installation of Void Linux

Step 1 - Mirror, C library, Architecture, and Base-system metapackage

While this step is not required it is a good idea to choose a mirror close to your location, A list of mirrors can be found at https://xmirror.voidlinux.org/. However; you can just go with the default mirror if you choose to.

The closest mirror to me is Chicago, so in this example, I will use the Chicago mirror, `https://mirrors.servercentral.com/voidlinux/`.

Aside from which mirror you want to use, there are a few other choices you will need to make, one of these is which C library to use, Void gives the option of using glibc (Gnu C library) or musl (designed to be more lightweight).

- glibc: /current
- musl: /current/musl

There is also the question of architecture, you will need to select which architecture you plan to use from the following list:

- x86_64
- x86_64-musl
- i686
- aarch64
  I will use x86_64 for this example

Once you decide which mirror and library you want to use (we will be using Chicago mirror and glibc library), we can create a couple more variables to make the next steps a little easier, we need to create a variable for our repo, and a variable for our architecture.

```
REPO=https://mirrors.servercentral.com/voidlinux/current/
ARCH=x86_64
```

Create directory and copy RSA keys for verifying package integrity

```
mkdir -p /mnt/var/db/xbps/keys
cp /var/db/xbps/keys/* /mnt/var/db/xbps/keys/
```

Now we can install the base system along with a few other odds and ends. (this tutorial will install vim and the linux-mainline kernel, if you want stable, install linux instead of linux-mainline)

```
XBPS_ARCH=$ARCH xbps-install -S -R "$REPO" -r /mnt base-system linux-mainline btrfs-progs cryptsetup vim
```

Step 2 - Chroot

With the base system installed we are ready to chroot into our void environment but first we need to mount a pseudo-filesystem needed for chroot

```
for dir in dev proc sys run; do mount --rbind /$dir /mnt/$dir; mount --make-rslave /mnt/$dir; done
```

Copy dns configuration into new root so xbps can download new packages inside

```
cp /etc/resolve.conf /mnt/etc/
```

chroot into system

```
BTRFS_OPTS=$BTRFS_OPTS PS1='(chroot) # ' chroot /mnt/ /bin/bash
```

Step 3 - Installation Configuration

Next we will begin to configure our new install, this will involve setting a hostname, timezone, our hosts, creating an fstab, and more.
First lets look at our rc.conf, you can find and go over the options in this file in the void docs at `httsp://docs.voidlinux.org/config/rc-files.conf` we can make changes in here but it is not necessary

```
vim /etc/rc.conf
```

Set timezone, you can list out all available timezones by running the following command

```
ls /usr/share/zoneinfo
```

Once you locate the timezone you need, set the timezone in the /etc/localtime file, I will set the timezone to chicago in this example

```
ln -sf /usr/share/zoneinfo/America/Chicago /etc/localtime
```

Next we will set our locale by editing /etc/default/libc-locales and uncommenting your locale. For me this would be en_US.UTF-8 and en_US ISO-8859-1

```
vim /etc/default/libc-locales
```

Once you uncomment and save the changes, you need to reconfigure

```
xbps-reconfigure -f glibc-locales
```

Set hostname in /etc/hostname
Replace <your chosen hostname> with what ever you want your hostname to be.

```
echo "<your chosen hostname>" > /etc/hostname
```

Create the /etc/hosts file, replace myhostname with the hostname you created in the last step.

```
cat <<EOF > /etc/hosts
#
# /etc/hosts: static lookup table for host names
#
```

```
127.0.0.1        localhost
::1              localhost
127.0.1.1        myhostname.localdomain myhostname
EOF
```

Step 4 - User Management

Now we will move into user management
First we need to create the root password

```
passwd
```

Now we can create a new user and give that user a password, add them the necessary groups, and give them sudo privilege
Replace <USER> with the name of the user you are adding

```
useradd <USER>
passwd <USER>
usermod -aG wheel, (any other groups you want to add user to) <USER>
```

Now we can change the shell for the root user to bash and edit the sudoers file

```
chsh -s /bin/bash root
EDITOR=vim visudo
```

Uncomment the line that says # %wheel ALL=(ALL:ALL) ALL
Then you can also add the following line below the line you just uncommented, this is not required but it is an option

```
<USER> ALL=(ALL:ALL) ALL
```

Step 5 - REPOS

Sync repositories

```
xbps-install -S
```

This step is not required unless you want to be able to access software that does not have free licenses or if you are a gamer or need 32bit packages

```
xbps-install void-repo-nonfree
xbps-install -S
xbps-install void-repo-multilib
xbps-install -S
```

step 6 - Create fstab

Now we can create our fstab, there are multiple ways to do this but we are going to use some command line magic and make this as easy as possible

Create variables for different volumes

```
EFI_UUID=$(blkid -s UUID -o value /dev/sdX1)
ROOT_UUID=$(blkid -s UUID -o value /dev/mapper/cryptvoid)
LUKS_UUID=$(blkid -s UUID -o value /dev/sdX2)
```

Next we will use these variables and create and populate our fstab

```
cat <<EOF > /etc/fstab
UUID=$ROOT_UUID / btrfs $BTRFS_OPTS,subvol=@ 0 1
UUID=$ROOT_UUID /home btrfs $BTRFS_OPTS,subvol=@home 0 2
UUID=$ROOT_UUID /.snapshots btrfs $BTRFS_OPTS,subvol=@snapshots 0 2
UUID=$EFI_UUID /efi vfat defaults,noatime 0 2
tmpfs /tmp tmpfs defaults,nosuid,nodev 0 0
EOF
```

Step 7 - Install and setup bootloader

We are now on the homestretch, a few more commands and it will be time to reboot

Install the grub package for efi

```
xbps-install grub-x86_64-efi
```

Enable encryption

```
echo GRUB_ENABLE_CRYPTODISK=y >> /etc/default/grub
```

Edit /etc/default/grub file

```
vim /etc/default/grub
GRUB_CMDLINE_LINUX_DEFAULT="loglevel=4 rd.auto=1 rd.luks.allow-discards"
```

Install grub

```
grub-install --target=x86_64-efi --efi-directory=/efi --bootloader-id="Void"
```

Section 3 - Miscellaneous

Step 1 - Keyfile (optional)

Create a keyfile to avoid typing passphrase 2x on boot

Create a keyfile out of random data

```
dd bs=515 count=4 if=/dev/urandom of=/boot/keyfile.bin
```

Add a 2nd key slot to the LUKS encrypted volume with keyfile.bin as the key

```
cryptsetup -v luksAddKey /dev/sdX2 /boot/keyfile.bin
```

Secure the keyfile by setting appropriate permissions

```
chmod 000 /boot/keyfile.bin
```

Allow only root access to /boot

```
chmod -R g-rwx,o-rwx /boot
```

Setup crypttab

```
cat <<EOF >> /etc/crypttab
cryptvoid UUID=$LUKS_UUID /boot/keyfile.bin luks
EOF
```

Cofigure dracut to include the keyfile and crypttab in the initial RAM disk

```
echo 'install_items+=" /boot/keyfile.bin /etc/crypttab "' > /etc/dracut.conf.d/10-crypt.conf
ln -s /etc/sv/dhc /etc/runit/runsvdir/default
```

Step 2 - Install software/programs/tools

Install software or programs you may want to install at this time

```
xbps-install <list of desired programs>
```

Step 3 - Link services

Link services that we want to start on boot
If you want to use a wired connection

```
ln -s /etc/sv/dhcpcd-eth0 /var/service
ln -s /etc/sv/dhcpcd /var/service
```

If you want to use wifi you can use wpa-supplicant or you can install Network Manager I prefer to use Network Manager so that is what I will use as an example

```
xbps-install NetworkManager
ln -s /etc/sv/NetworkManager /var/service
```

Use xbps to verify installed programs are configured correctly

```
xbps-reconfigure -fa
```

Step 4 - Exit and Reboot

Exit chroot

```
exit
```

Reboot system

```
reboot
```