▸ Table of contents

VoidLinuxInstaller script

The **VoidLinuxInstaller script** is an attempt to make my gist an interactive bash script.

As stated in the gist, this script provides:

- Optional Full Disk Encryption (including `/boot`) with LUKS1/2;
- Optional Logic Volume Management (LVM);
- BTRFS as filesystem;
- EFISTUB (LUKS1/2) or GRUB2 (LUKS1 only due to limitations for encrypted `/boot`) as bootloader;
- Optional swapfile enabling also zswap (swap is needed if you plan to use hibernation);
- Enable trim if the selected drive supports it.

To know how the script works in details, please jump to the How does it work? section!

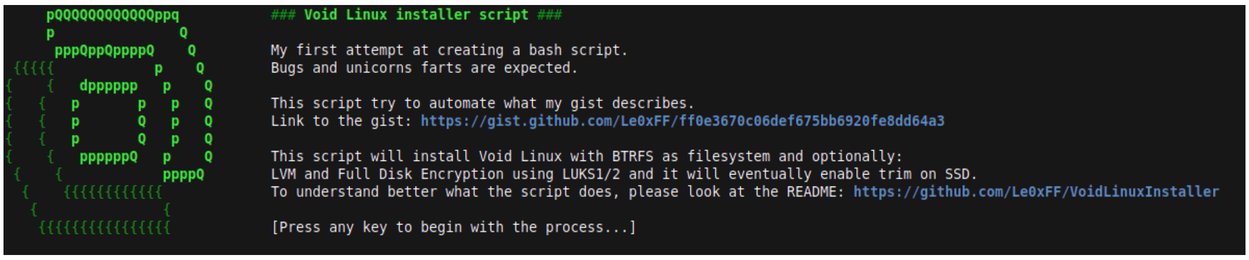
To know how to run the script, please jump to the How to run it? section!

This script comes from my need to automate my gist as much as I can, and also as a way to learn Bash scripting as well. *This is my first Bash script ever created so bugs, errors and really ugly code are expected!*

I've tried this script a lot with virtual machines and following every step always brought me to a functional system, so there should be no problem from this point of view!

Pull requests are absolutely welcome!

Some fancy screenshots



```
     pQQQQQQQQQQQQppq           ### Void Linux installer script ###
     p               Q
      pppQppQppppQ    Q         My first attempt at creating a bash script.
 {{{{{            p    Q        Bugs and unicorns farts are expected.
{    {   dpppppp   p    Q
{   {   p       p   p   Q       This script try to automate what my gist describes.
{   {   p       Q   p   Q       Link to the gist: https://gist.github.com/Le0xFF/ff0e3670c06def675bb6920fe8dd64a3
{   {   p       Q   p   Q
{    {   ppppppQ   p    Q       This script will install Void Linux with BTRFS as filesystem and optionally:
 {    {            ppppQ        LVM and Full Disk Encryption using LUKS1/2 and it will eventually enable trim on SSD.
  {    {{{{{{{{{{{{             To understand better what the script does, please look at the README: https://github.com/Le0xFF/VoidLinuxInstaller
   {               {
    {{{{{{{{{{{{{{{{            [Press any key to begin with the process...]
```

How to run it?

First update `xbps` package:

```
xbps-install -Suvy xbps
```

then install `wget` or `curl` package:

```
# For wget
xbps-install -Suvy wget

# For curl
xbps-install -Suvy curl
```

then download the needed file:

```
# For wget
wget https://raw.githubusercontent.com/Le0xFF/VoidLinuxInstaller/main/vli.

# For curl
curl -o $HOME/vli.sh https://raw.githubusercontent.com/Le0xFF/VoidLinuxIns
```

then make it executable:

```
chmod +x $HOME/vli.sh
```

and finally run it:

```
bash $HOME/vli.sh
```

How does it work?

Here is documented how the script works in details and what will ask to the user in each different step. It will:

1. prompt the user to eventually change their keyboard layout from a list of all the different available layouts.
2. check internet connection and eventually guide the user to connect to the internet;
3. wipe a user choosen drive and that drive will be the one also selected for partitioning;
4. partition a user choosen drive:
   - if the previous drive was not the right one, it will ask the user if they want to change it eventually;
   - check the Suggested partition layout to follow the script workflow;
5. ask user if they want to encrypt a partition for Full Disk Encryption:
   - it will ask for a mountpoint name, so that the encrypted partition will be mounted as `/dev/mapper/<encrypted_name>` ;
6. ask user if they want to apply Logical Volume Management to the previous encrypted partition, to have the flexibility to resize `/` and to add more space in the future without reformatting the whole system:
   - it will ask for a Volume Group name, so that will be mounted as `/dev/mapper/<vg_name>` ;
   - it will ask for a Logical Volume name for **root** partition and its size will be the previously selected partition, so that will be mounted as `/dev/mapper/<vg_name>-<lv_root_name>` ;
   - check the Final partitioning result to get an overview of what the outcome will be;
7. Formatting partitions to proper filesystems:
   - it will prompt user to select which partition to use as **boot** partition and to choose its label; it will be formatted as FAT32 and mounted as `/boot/efi` ;
   - it will prompt user to select a label for the **root** logical volume, that will be formatted as BTRFS;

8. create BTRFS subvolumes with specific fstab mount options; if user wants to change them, please edit the script looking for `create_btrfs_subvolumes` function (BTRFS mount options official documentation):
   - **BTRFS mounting options**:
     - `rw,noatime,discard=async,compress-force=zstd,space_cache=v2,commit=120`
   - **BTRFS subvolumes that will be created**:
     - `/@`
     - `/@home`
     - `/@snapshots`
     - `/var/cache/xbps`
     - `/var/tmp`
     - `/var/log`
9. install base system:
   - It will ask user to choose between `x86_64` and `x86_64-musl` ;
10. chroot:
    - set *root* password and `/` permissions;
    - create proper `/etc/fstab` file;
    - if encryption was choosen, generate random key to avoid typing password two times at boot;
    - create proper dracut configuration and initramfs;
    - optionally create a swapfile and enable zswap;
    - install any additional package;
    - enable or disable available services;
    - ask user which bootloader to install: EFISTUB or GRUB2;
    - create new users and configure them;
    - git clone void-packages or a custom public repository for selected users (it's not possible to `binary-bootstrap` because `xbps-src` can't do that while being already in a chrooted environment; see related issues: #30496, #35018, #35410)
    - choose timezone, keyboard layout integrating it eventually even in GRUB2, locale, hostname and default shell for root user;
    - configure AppArmor and reconfigure every package.

Suggested partition layout

To have a smooth script workflow, the following is the suggested disk layout:

- GPT as disk label type for UEFI systems, also because this script will only works on UEFI systems;
- Less than 1 GB for bootable EFI partition, as first partition and as EFI System type;
- Rest of the disk for the Volume Group, where encryption and LVM will eventually be applied, as second partition as Linux filesystem.

These two will be physical partition.
You don't need to create a `/home` partition because BTRFS subvolumes will take care of that.

Final partitioning result

Following the script, at the very end your drive will end up being like the following, if you chose LUKS, LVM and GRUB2 as bootloader:

```
/dev/nvme0n1                                259:0    0 953,9G  0 disk
├─/dev/nvme0n1p1                            259:1    0     1G  0 part  /boo
└─/dev/nvme0n1p2                            259:2    0 942,9G  0 part
  └─/dev/mapper/<encrypted_name>            254:0    0 942,9G  0 crypt
    └─/dev/mapper/<vg_name>-<lv_root_name> 254:1    0 942,9G  0 lvm   /.sn
                                                                      /hom
                                                                      /
```

> Note: `/.snapshots` will be available after following the Follow up for `@snapshots` subvolume section.

Follow up for `@snapshots` subvolume

With this script, the `@snapshots` subvolume will be created, but not the `/.snapshots` folder. This is done to avoid stupid snapper issues when trying to create a configuration for `/` .

So after installing `snapper` from Void Linux's repositories and after creating a configuration for `/` , you have to delete the subvolume that snapper will automatically create. After that create the `/.snapshots` folder and then uncomment the relative line from `/etc/fstab` :

```
# run these commands as root

snapper -c root create-config /
btrfs subvolume delete /.snapshots
mkdir /.snapshots
sed -i '/@snapshots/s/^#//' /etc/fstab
reboot
```

> Source: Arch Wiki

How to unlock encrypted partition in GRUB2

In case a wrong password was put in GRUB2, a shell will be dropped.

In order to boot the system, the encrypted partition have to be unlocked and the right GRUB configuration file must be loaded.

GRUB2 standard keyboard layout

If a custom keyboard layout was **not** choosen, the following commands must be put in GRUB2 shell:

```
# unlock all the encrypted partitions
cryptomount -a

# input your password, then insert module normal and run it
insmod normal
normal
```

GRUB2 custom keyboard layout

If a custom keyboard layout was choosen, the following commands must be put in GRUB2 shell:

```
# unlock all the encrypted partitions
cryptomount -a

# input your password and if LVM was used, then
set root=(lvm/<vg_name>-<lv_root_name>)/@

# if LVM was not used, then (<uuid> is the same printed on screen; you can
set root=(cryptouuid/<uuid>)/@

# finally load main GRUB2 configuration file
set prefix=$root/boot/grub
configfile $prefix/grub.cfg
```

What to do if system breaks?

In case anything will break, you will just have to delete the `@` subvolume, create it again and reinstall your distro. `/home` folder won't be affected in any way.

In details, after booting a LiveCD, mount the encrypted partition:

```
cryptsetup open /dev/nvme0n1p2 <encrypted_name>
```

Scan for Volume Groups and then enable the one you need:

```
vgscan
vgchange -ay <vg_name>
```

Mount the true btrfs root by its subvol or by its subvolid:

```
# by subvol
mount -o subvol=/ /dev/mapper/<vg_name>-<lv_root_name> /mnt

# or by subvolid
mount -o subvolid=0 /dev/mapper/<vg_name>-<lv_root_name> /mnt
```

After that if you do an `ls /mnt/` you will see all the subvolume previously created.
Now you must delete **ONLY** the `@` subvolume and finally unmount `/mnt` :

```
btrfs subvolume delete /mnt/@
umount /mnt
```

You now have to reinstall Void Linux manually (the script is not programmed to help you this time). For this you can follow the original gist and start again from *Mount partitions and create btrfs subvolumes* instruction, without creating the `@home` subvolume.

When the package reconfiguration is finished, you have to create a user with the same name of the one you created before, possibly adding it to the same groups as before, but you can do it later too.
Don't add the `-m` flag or your original home folder will be destroyed:

```
useradd -G wheel <same_user>
passwd <same_user>
```

This is not necessary because adding the same user will automatically change the home folder permission, but just in case:

```
chown -R <same_user>:<same_user> /home/<same_user>
```

How to add more space with a new drive with LVM

> Probably the following could also be done from a running system, but maybe it's better to boot a LiveCD just in case.

First a new physical drive must be added to the system. This drive in the following example will be called `/dev/sda`.

After booting a LiveCD, mount the encrypted partition:

```
cryptsetup open /dev/nvme0n1p2 <encrypted_name>
```

Scan for Volume Groups and then enable the one you need:

```
vgscan
vgchange -ay <vg_name>
```

Scan for Logical Volumes and then enable the one you need:

```
lvscan
lvchange -ay <vg_name>/<lv_root_name>
```

Then a new Physical Volume for the new drive must be created:

```
pvcreate /dev/sda
```

After that it must be added to the existing Logical Volume:

```
vgextend <lv_root_name> /dev/sda
```

Then the Logical Volume must be extended to cover the new free space:

```
lvm lvextend -l +100%FREE <vg_name>/<lv_root_name>
```

Finally also the BTRFS filesystem must be extended to cover all the free space; to do that, the BTRFS partition must be mounted:

```
mount -t btrfs /dev/mapper/<vg_name>-<lv_root_name> /mnt/
```

```
btrfs filesystem resize max /mnt/
```

Notes

- To speed up decryption at boot you could lower the number of iteration for each key as described on the Arch Wiki, but keep in mind that this comes with its own security concerns.

Resources

> **[1]** https://tldp.org/LDP/Bash-Beginners-Guide/html/index.html

> **[2]** https://gist.github.com/tobi-wan-kenobi/bff3af81eac27e210e1dc88ba660596e

> **[3]** https://gist.github.com/gbrlsnchs/9c9dc55cd0beb26e141ee3ea59f26e21

> **[4]** https://unixsheikh.com/tutorials/real-full-disk-encryption-using-grub-on-void-linux-for-bios.html

> **[5]** https://wiki.archlinux.org/title/GRUB/Tips_and_tricks#Manual_configuration_of_core_image_for_early_boot

> **[6]** https://cryptsetup-team.pages.debian.net/cryptsetup/encrypted-boot.html#using-a-custom-keyboard-layout