

Home / Documentation / [How to: Chroot into a broken system via live CD/ISO or alternate Linux system](#)

How to: Chroot into a broken system via live CD/ISO or alternate Linux system

Occasionally, you may need to rescue a broken/unbootable TurnKey server. Perhaps you have had a hardware failure, or perhaps you've inadvertently broken something, or perhaps you've installed a buggy update that has broken things?

Often these issues can be fixed by "chrooting" into the broken system from a functioning (Linux) system, then taking the required steps to resolve the issue (reinstall grub, reinstall a kernel, rebuild an initramfs, etc). Obviously the steps to fix a broken system will depend on what has actually broken. If you're not sure, then you're probably best to start a new thread in the [forums](#) (requires free website user account to start new thread) so we can assist to diagnose the issue.

In this doc, I will just advise up until the point where you enter the chroot, and won't go into the details of how to resolve any specific issue (although, if/when I get a chance, perhaps I'll write up doc pages for some of the most common issues?!).

Note: This guide assume that you are logged in with the root user account. If not, most (if not all) the commands run outside the chroot may need to be prefaced

by 'sudo'. If you enter the chroot via sudo, then it should not be required once within the chroot.

So first up; a little background. If you'd rather just jump straight into [setting up the chroot](#), please feel free.

What is a "chroot"

According to Wikipedia, a [chroot](#):

“ changes the apparent root directory for the current running process and its children. A program that is run in such a modified environment cannot name (and therefore normally cannot access) files outside the designated directory tree.

As a Linux user, there are 2 types of chroot you may encounter. What I call a "standard" chroot (what this doc page covers), then there is also a "chroot jail".

The purpose of a chroot jail is to lock a user or process within a certain part of a directory tree. E.g. a program may be "chrooted" into its own directory (Postfix is set up like this for example), with no access to the rest of the filesystem. Or a limited user may be "chrooted" into their home directory, so they can use (for example) SFTP, but cannot browse the whole filesystem.

A "standard" chroot can be used to investigate and possibly repair a unbootable system. It is also commonly used to build new Linux systems. For example, TurnKey uses a "standard" chroot (on [TKLDev](#)) to build the TurnKey appliances. That is the sort of chroot I'll document here.

Setting up a chroot

Preparation

The first thing to do is to ensure that the filesystem of the broken system is accessible from the working system. In the case of a bare metal install or a VM, the easiest way to do this is often to just boot from a Live CD/USB/ISO. As a

general rule, it's usually best to use the same OS as what the broken system is. In the case of TurnKey, you can use **Core** (or any other server; just Core is the smallest) of the same major version. E.g. v15.0 Core is fine to use to fix any v15.x appliance. If that's not an option, a Debian Live system is also a good option. Worst case scenario, you could get away with any relatively recent Linux distro, although TurnKey (or Debian) of a matching version is preferable.

Whatever you do **DO NOT** install over the top of your broken system! That will make it unrecoverable!

Alternatively, you could move the disk (physical or virtual) to another (working) machine (with TurnKey/Debian/etc as above running). For an AWS server that is also essentially what you'll need to do (i.e. detach the primary volume from the broken system and reattach to a working system).

Finding the right volume to mount

After you've attached the broken primary volume to a working system, the next step is to mount it. To mount it, you'll need to ensure that you have a clean mount location and you'll also need to work out which disk/volume/partition(s) you need to mount.

One easy way to ensure that you have a clean mount location to use, is to create it! So something like this:

```
mkdir /rescue
```

Ensuring that you find the correct drive/volume/partition(s) can be a little trickier, but one tool you can use is `fdisk`. To view all attached volumes, run:

```
fdisk -l
```

Here is the output on an AWS server I'm currently working on. It has a secondary drive (the root volume of another server) attached as `/dev/xvdf`, the default root volume (i.e. the root volume of the working server) is `/dev/xvda`. That may be different for you (although if you're on AWS is likely the same). Here is the full output:

```
root@core ~# fdisk -l
Disk /dev/xvda: 10 GiB, 10737418240 bytes, 20971520 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 6183BC26-31B1-41F2-9AB9-319B7C4A85B0
```

Device	Start	End	Sectors	Size	Type
/dev/xvda1	2048	6143	4096	2M	BIOS boot
/dev/xvda2	6144	20969471	20963328	10G	Linux filesystem

```
Disk /dev/xvdf: 10 GiB, 10737418240 bytes, 20971520 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 0029AB38-4BE5-4009-B163-9A7B4F597FE4
```

Device	Start	End	Sectors	Size	Type
/dev/xvdf1	2048	6143	4096	2M	BIOS boot
/dev/xvdf2	6144	20969471	20963328	10G	Linux filesystem

If you are unsure which volume is the one, check which volumes are already mounted. Do that like this:

```
mount | grep ^/dev
```

That will check for all lines from the mount output that start with /dev. In my case, this confirms what I know already, the volume I already have mounted is /dev/xvda:

```
root@core ~# mount | grep ^/dev
/dev/xvda2 on / type ext4 (rw,relatime,data=ordered)
/dev/xvda2 on /tmp type ext4 (rw,relatime,data=ordered)
/dev/xvda2 on /var/cache/tklbam type ext4 (rw,relatime,data=ordered)
```

Mounting the broken system ready to chroot

So once you've worked out which partition you'll need to mount it. That's pretty simple. In my case, the partition that I need to mount is /dev/xvdf2 (i.e. the

second partition on /dev/xvf). If you look above, at my output from fdisk, you'll see that /dev/xvdf1 is type "BIOS boot" and /dev/xvdf2 is "Linux filesystem". To mount that is as easy as this:

```
mount /dev/xvdf2 /rescue
```

If you now check in /rescue (i.e. ls /rescue), you should see something like this:

```
root@core ~# ls -l /rescue
total 92
drwxr-xr-x  2 root root  4096 Apr 16 02:49 bin
drwxr-xr-x  3 root root  4096 Apr 16 02:50 boot
drwxr-xr-x  5 root root  4096 Oct  3  2018 dev
drwxr-xr-x 100 root root  4096 Apr 16 02:50 etc
drwxr-xr-x  3 root root  4096 Oct  3  2018 home
lrwxrwxrwx  1 root root    29 Oct  3  2018 initrd.img -> boot/initrd.img-4.9.0
lrwxrwxrwx  1 root root    29 Oct  3  2018 initrd.img.old -> boot/initrd.img-4
drwxr-xr-x 18 root root  4096 Oct  3  2018 lib
drwxr-xr-x  2 root root  4096 Mar 11  2018 lib64
drwx----- 2 root root 16384 Oct  3  2018 lost+found
drwxr-xr-x  2 root root  4096 Mar 11  2018 media
drwxr-xr-x  4 root root  4096 Apr 16 02:49 mnt
drwxr-xr-x  2 root root  4096 Mar 11  2018 opt
drwxr-xr-x  2 root root  4096 Feb 23  2018 proc
drwx----- 5 root root  4096 Oct  3  2018 root
drwxr-xr-x 12 root root  4096 Oct  3  2018 run
drwxr-xr-x  2 root root  4096 Apr 16 02:49 sbin
drwxr-xr-x  2 root root  4096 Mar 11  2018 srv
drwxr-xr-x  2 root root  4096 Feb 12  2017 sys
drwxrwxrwt  7 root root  4096 Apr 16 02:51 tmp
drwxr-xr-x 10 root root  4096 Mar 11  2018 usr
drwxr-xr-x 13 root root  4096 Oct  3  2018 var
lrwxrwxrwx  1 root root    26 Oct  3  2018 vmlinuz -> boot/vmlinuz-4.9.0-8-amd
lrwxrwxrwx  1 root root    26 Oct  3  2018 vmlinuz.old -> boot/vmlinuz-4.9.0-8
```

Obviously the dates will be different, but otherwise, that shows that the system is mounted as expected. All current builds of TurnKey include the /boot partition, although in some older releases, if the install used LVM, then the /boot partition may need to be mounted separately. In that case, after the main OS is mounted, mount the boot directory. I.e. something like this (adjust to suit):

```
mount /dev/sdXN /rescue/boot
```

Where /dev/sdXN is the partition where the boot info resides.

But wait there's more...! Once the main OS has been mounted, you then also need to mount some special directories from the host so that the chroot will function properly. Primarily these are /dev, /proc and /sys. /dev/pts is also worth mounting too. In my experience, you can get away without mounting that, but you will often get errors, so it's worth mounting that to reduce the noise.

```
mount -t proc proc /rescue/proc
mount -t sysfs sys /rescue/sys
mount -o bind /dev /rescue/dev
mount -t devpts pts /rescue/dev/pts
```

Chrooting in

Now you simply need to chroot in. That's done like this:

```
chroot /rescue
```

Simple as that...! :)

Now do whatever you need to do to fix your system. Once you are happy (or just ready to test) then you can exit the chroot like this:

```
exit
```

Cleaning up afterwards

If you need to stop the server to remove the broken drive, then no clean up is required. All the components should be auto unmounted as part of the shutting down process. If if you don't need to stop the server, rebooting is the easiest way to clean up the chroot. However, if you don't stop the server you're using, or for some other reason would like to undo the chroot config, then after you have exited the chroot, you just need to backtrack on the above commands. I.e. use umount instead of mount, and unmount them in the reverse order they were

mounted. I.e.:

```
umount /rescue/dev/pts
umount /rescue/dev
umount /rescue/sys
umount /rescue/proc
```

If you mounted a separate boot partition, unmount that next. I.e. 'umount /rescue/boot'. Then finally, the main volume:

```
umount /rescue
```

[◀ How to upgrade an appliance](#) [up](#) [Hub/EC2 - Add new Elastic IP address \(via AWS console\) ▶](#)

[Add new comment](#)

Comments



chroot using live Try Ubuntu

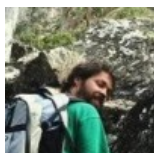
Rediet - Thu, 2022/08/18 - 08:27

Please help me! When I finally write the command

```
sudo chroot /rescue
```

I have found this error " chroot: failed to run command
'/bin/bash': No such file or directory

[reply](#)



Sounds like you haven't done the submounts!?

Jeremy Davis - Thu, 2022/08/25 - 06:10

In the tutorial above, there are a few lines that need to be run *before* you chroot in (which mounts /dev, /proc, etc from the host, into the chroot guest). The output you've posted suggests that either you didn't do that, or one or more of those commands failed.

If you're not running the commands as root, then my guess is that you didn't preface the relevant commands with 'sudo'. Another way to go is to just run the commands as root. You can do that by first running 'sudo su -', then all future commands will run as root. To get back to your normal user, just type 'exit'.

[reply](#)

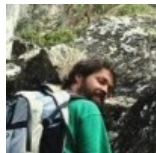


Same. Seems like it is distro

Linards Liepins - Sat, 2023/02/18 - 17:29

Same. Seems like it is distro-specific.

[reply](#)



Do you have /dev, /proc and /sys mounted properly?

Jeremy Davis - Sun, 2023/02/19 - 02:09

As I hinted in my post above, in my experience, that error message is usually caused by not having the system mounts set up correctly. Double check you have them mounted properly.

[reply](#)



I did exactly same steps but still getting same error

Paurush Saxena - Sat, 2024/09/28 - 04:27

I did exactly same steps but still getting same error as described here..

chroot : failed to run command /bin/bash no such file or directory

Is there something more that needs to be mounted? I am using RHEL9

APPS

Specials	
Web development	>
IT Infrastructure	
Content management	>
Business management	>
Messaging	>
Issue tracking	>
Database	>
Developer tools	

HELP


Forums	>
Development	
Documentation	

User login

E-mail or username *

Password *

☐ Remember me

 [Log in using OpenID](#)
[Create new account](#)
[Request new password](#)

LOG IN



ABOUT

[Why use TurnKey?](#)

[Code of Conduct](#)

[Mirrors](#)

[Contact Us](#)

[Partners](#)

[Sponsor](#)

SOLUTIONS

[Core: common base image](#)

[TurnKey Hub: cloud deployment](#)

[TKLBAM: backup and migration](#)

[TKLDev: TurnKey build system](#)

HELP

[Forums](#)

[Frequently Asked Questions](#)

[Development](#)

[Donate](#)

APPS

[Content Management](#)

[IT Infrastructure](#)

[Web Development](#)

[Issue Tracking](#)

[Messaging](#)

[Special Purpose](#)

TOP DOWNLOADS

[OpenVPN](#)

[Node.js stack](#)

[LAMP stack](#)

[Wordpress](#)

[File server](#)

[Redmine](#)

