iSCSI Boot

You can install Arch on an **iSCSI Target**. This allows you to boot from an iscsi target using a diskless machine. No physical disk is required unless you need an ipxe boot USB (because your NIC isn't iBFT capable or you don't want to setup a TFTP server).

Related articles

- iSCSI Target
- iSCSI Initiator

Contents

Server / Target Setup

You can set up an iSCSI target with any hosting server OS. Follow the procedure outlined in **iSCSI Target** if you use Arch Linux as the hosting server OS.

Client / Initiator Setup

Overview

1. install `open-iscsi (https://www.archlinux.org/packages/?name=open-iscsi)` package in installer system
2. connect to iSCSI target and create partitions on logical drive of target.
3. install Arch Linux system in usual way
4. install `open-iscsi (https://www.archlinux.org/packages/?name=open-iscsi)` package in installed system
5. create initial RAM disk image containing open-iscsi modules.

Note: In addition to the above, you have to prepare sanboot-able infrastructure which is necessary to load boot loader (GRUB, etc.) from remote disc. Some network interface cards support sanboot. If you don't have such cards, you can use **iPXE (http://ipxe.org/)**, **gPXE (http://etherboot.org/wiki/start)**, and so on.

Install over iSCSI

Download Arch Linux ISO image **[1] (https://www.archlinux.org/download/)** and boot Arch Linux using the ISO image. After Arch Linux is booted, either use net as the install source or manually ifconfig and dhcp.

1. Unfortunately ISO install image does not include modules for iSCSI, you have to install and setup them at first.

Before you continue to "Partition the disks", **install** the **`open-iscsi (https://www.archlinux.org/packages/?name=open-iscsi)`** package from the **official repositories** and connect to target.

In the following, server's(target's) IP address is 192.168.1.100, client's(initiator's) IP address is 192.168.1.101, iSCSI initiator name is "iSCSI.Initiator.Name" and target name is "iSCSI.Target.Name". You should, of course, be sure to your network configration and so on.

```
pacman -Sy
pacman -S open-iscsi
modprobe iscsi_tcp
```

2. Get a list of targets

```
iscsiadm -m discovery -t st -p 192.168.1.100
```

Sample output (your output may differ depending on the portal ip and target name)

```
192.168.1.100:3260,-1 iqn.2011-03.example.org.istgt:arch
```

Connect to the target

```
iscsiadm -m node -T iqn.2011-03.example.org.istgt:arch -p 192.168.1.100 -l
```

Now your local host connects to the drive of target host (see dmesg output).

Note: It is recommended that you **NOT** include swap on the iSCSI drive when creating the partitions, you can just **ignore** the warning.

3. You can create a partition table and partitions in the same way as a local drive. And continue to install Arch Linux in the usual way.

4. **Install** the **`open-iscsi (https://www.archlinux.org/packages/?name=open-iscsi)`** package in the "future" root file system.

5. Before doing **mkinitcpio** in the "future" root file system, you have to prepare the following two files.

i) `/mnt/usr/lib/initcpio/install/iscsi`

```
build ()
{
    local mod
    for mod in iscsi_ibft iscsi_tcp libiscsi libiscsi_tcp scsi_transport_iscsi crc32c; do
        add_module "$mod"
    done

    add_checked_modules "/drivers/net"
    add_binary "/usr/bin/iscsistart"
    add_runscript
}

help ()
{
cat <<HELPEOF
  This hook allows you to boot from an iSCSI target.
HELPEOF
}
```

Using an iBFT Compatible Rom

If using a iBFT compatible NIC or boot device (such as ipxe) you can use auto configuration to set the network configuration and iscsi target.

ii) `/mnt/usr/lib/initcpio/hooks/iscsi`

```
run_hook ()
{
    modprobe iscsi_tcp
    modprobe iscsi_ibft

    echo "Network configuration based on iBFT"
    iscsistart -N || echo "Unable to configure network"

    echo "iSCSI auto connect based on iBFT"
    iscsistart -b || echo "Unable to auto connect"
}
```

Manually Setting the iSCSI Target

If you are not using an iBFT compatible boot rom you must explicitly setup the network and the iscsi target manually.

ii) `/mnt/usr/lib/initcpio/hooks/iscsi`

```
run_hook ()
{
    modprobe iscsi_tcp
    ifconfig eth0 192.168.1.101 netmask 255.255.255.0 broadcast 192.168.1.255
    sleep 2
    iscsistart -i iSCSI.Initiator.Name -t iSCSI.Target.Name -g 1 -a 192.168.1.100
}
```

Using DHCP

If you want to use dhcp for the above script, you may use the following hook, but make sure that **dhcpcd (https://www.archlinux.org/packages/?name=dhcpcd)** is installed and is added to the BINARY line in **/etc/mkinitcpio.conf**

ii) `/mnt/usr/lib/initcpio/hooks/iscsi`

```
run_hook ()
{
    modprobe iscsi_tcp
    mkdir -p /var/lib/dhcpcd
    dhcpcd eth0
    sleep 2
    iscsistart -i iSCSI.Initiator.Name -t iSCSI.Target.Name -g 1 -a 192.168.1.100
}
```

Add "iscsi" to the HOOKS line in **/etc/mkinitcpio.conf**.

Then run "mkinitcpio -p linux" and new `/boot/initramfs-linux.img` and `/boot/initramfs-linux-fallback.img` will be generated.

Note: If you plan on booting this installation of Arch on machines with nic cards that require different modules, remove "autodetect" from HOOKS

Note: Rebuilding the initial ramdisk will take some time if autodetect is removed from HOOKS

Now your new system can mount the file systems from iSCSI target drive after reboot.

Configuring Open iSCSI initiator on the diskless system

The initiator on the diskless system must be configured to make it resistant to network problems or even the restart of the target system, the open-iscsi **README (https://github.com/open-iscsi/open-iscsi/blob/master/README)** describes optimal iSCSI settings for iSCSI root

When accessing the root partition directly through a iSCSI disk, the iSCSI timers should be set so that iSCSI layer has several chances to try to re-establish a session and so that commands are not quickly requeued to the SCSI layer. Basically you want the opposite of when using dm-multipath.

For this setup, you can turn off iSCSI pings by setting:

```
node.conn[0].timeo.noop_out_interval = 0
node.conn[0].timeo.noop_out_timeout = 0
```

And you can turn the replacement_timer to a very long value:

```
node.session.timeo.replacement_timeout = 86400
```

If a network problem is detected by the initiator, the running commands are failed immediately. There is one exception to this and that is when the SCSI layer's error handler is running. To check if the SCSI error handler is running iscsiadm can be run as:

```
iscsiadm -m session -P 3
```

You will then see:

Host Number: X State: Recovery When the SCSI EH is running, commands will not be failed until node.session.timeo.replacement_timeout seconds.

To modify the timer that starts the SCSI EH, you can either write directly to the device's sysfs file:

```
echo X > /sys/block/sdX/device/timeout
```

where X is in seconds or on most distros you can modify the udev rule.

To modify the timeout using a udev rule create /etc/udev/rules.d/50-iscsi.rules, and add the following lines:

```
ACTION=="add", SUBSYSTEM=="scsi" , ATTR{type}=="0|7|14", \
     RUN+="/bin/sh -c 'echo 90 > /sys$$DEVPATH/timeout'"
```

Change the echo 90 part of the line to the value that you want.

The default timeout for normal File System commands is 30 seconds when udev is not being used.

Since low level I/O commands will go through the IO scheduler on the target it is recommended for performance to use elevator=noop on the diskless system, see **Improving performance** to set the elevator.

Troubleshooting

Device not found

If you are having problems with detecting your eth0 interface you may need to explicitly install the kernel module for your NIC in the MODULES line in **/etc/mkinitcpio.conf**.

Retrieved from "https://wiki.archlinux.org/index.php?title=ISCSI_Boot&oldid=505368"