Architectures/RISC-V/Installing

From Fedora Project Wiki
< Architectures | RISC-V

This page describes the steps necessary to get Fedora for RISC-V running, either on emulated or real hardware.

Contents

🔗 Obtain and prepare a disk image

🔗 Tested images

You can find them here: https://dl.fedoraproject.org/pub/alt/risc-v/disk-images/fedora/

Download `Fedora-Developer-Rawhide-*-sda1.raw.xz` as well as `bbl-*.riscv64` and `initramfs-*.img` .

These disk images:

- contain a single "naked" filesystem (`root=/dev/vda`);
- have SELinux set to enforcing=1;
- have restored default SELinux security context (`restorecon -Rv /`);
- kernel, initramfs, config and BBL are available as separate downloads;
- have been booted in QEMU/libvirt a few times to verify.

If you are not sure which image to choose, go with this one.

🔗 Nightly builds

You can find them here: http://185.97.32.145/koji/tasks?state=closed&view=flat&method=createAppliance&order=-id

Select the most recent (top) build and download `Fedora-Developer-Rawhide-*-sda.raw.xz` .

These disk images:

- contain a partitioned disk (`root=/dev/vda1`);
- have SELinux set to enforcing=1;
- have restored default SELinux security context (`restorecon -Rv /`);
- kernel, initramfs, config and BBL are inside the disk image, in the `/boot` directory;
- are completely untested.

If you choose this image, you'll have to tweak most of the commands below to account for the different names for the root partition and the disk image itself.

🔗 Uncompress the image

Whether you have downloaded a tested image or a nightly build, you'll need to uncompress it before it can be used:

```
$ unxz Fedora-Developer-Rawhide-*.raw.xz
```

🔗 Optional: expand the disk image

You might want to expand the disk image before setting up the VM. Here is one example:

```
truncate -r Fedora-Developer-Rawhide-*-sda.raw expanded.raw
truncate -s 40G expanded.raw
virt-resize -v -x --expand /dev/sda1 Fedora-Developer-Rawhide-*-sda.raw
expanded.raw
virt-filesystems --long -h --all -a expanded.raw
virt-df -h -a expanded.raw
```

You can only perform this operation if you downloaded a nightly build.

The resulting disk image will work with QEMU as well as TinyEMU. Make sure you use `expanded.raw` instead of `Fedora-Developer-Rawhide-*-sda.raw` when booting the guest.

🔗 Optional: create an overlay

You can also create `qcow2` disk image with `raw` Fedora disk as backing one. This way Fedora `raw` is unmodified and all changes are written to `qcow2` layer. You will need to install `libguestfs-tools-c` .

```
qemu-img create -f qcow2 -F raw -b Fedora-Developer-Rawhide-*-sda.raw
overlay.qcow2 20G
virt-resize -v -x --expand /dev/sda1 Fedora-Developer-Rawhide-*-sda.raw
overlay.qcow2
virt-filesystems --long -h --all -a overlay.qcow2
```

You can only perform this operation if you downloaded a nightly build.

The resulting disk image will only work with QEMU. Make sure you use `overlay.qcow2` instead of `Fedora-Developer-Rawhide-*-sda.raw` when booting the guest.

🔗 Optional: set the hostname before booting

If you want to change hostname before the first boot, install `libguestfs-tools-c` and then run:

```
virt-customize -a Fedora-Developer-Rawhide-*-sda1.raw --hostname fedora-
riscv-mymagicbox
```

🔗 Nightly builds only: extracting kernel, config, initramfs and BBL

Fedora/RISC-V does not support BLS (Boot Loader Specification - more details (https://fedoraproject.org/wiki/Changes/BootLoaderSpecByDefault)).

Disk images contain a `/boot` directory from where you can copy out kernel, config, initramfs and BBL (contains embedded kernel).

This is **only** necessary for nightly builds, since for tested images these files are provided as separate downloads alongside the image.

Example session:

```
$ export LIBGUESTFS_BACKEND=direct
$ guestfish \
  add Fedora-Developer-Rawhide-20190126.n.0-sda.raw : run : \
  mount /dev/sda1 / : ls /boot | grep -E
'^(bbl|config|initramfs|vmlinuz)' | grep -v rescue
bbl-5.0.0-0.rc2.git0.1.0.riscv64.fc30.riscv64
config-5.0.0-0.rc2.git0.1.0.riscv64.fc30.riscv64
initramfs-5.0.0-0.rc2.git0.1.0.riscv64.fc30.riscv64.img
vmlinuz-5.0.0-0.rc2.git0.1.0.riscv64.fc30.riscv64
$ guestfish \
  add Fedora-Developer-Rawhide-20190126.n.0-sda.raw : run : mount
/dev/sda1 / : \
  download /boot/bbl-5.0.0-0.rc2.git0.1.0.riscv64.fc30.riscv64 bbl-
5.0.0-0.rc2.git0.1.0.riscv64.fc30.riscv64 : \
  download /boot/config-5.0.0-0.rc2.git0.1.0.riscv64.fc30.riscv64
config-5.0.0-0.rc2.git0.1.0.riscv64.fc30.riscv64 : \
  download /boot/initramfs-5.0.0-0.rc2.git0.1.0.riscv64.fc30.riscv64.img
initramfs-5.0.0-0.rc2.git0.1.0.riscv64.fc30.riscv64.img : \
  download /boot/vmlinuz-5.0.0-0.rc2.git0.1.0.riscv64.fc30.riscv64
vmlinuz-5.0.0-0.rc2.git0.1.0.riscv64.fc30.riscv64
```

You can also use `guestmount` or QEMU/NBD to mount disk image. Examples:

```
$ mkdir a
$ guestmount -a $PWD/Fedora-Developer-Rawhide-20190126.n.0-sda.raw -m
/dev/sda1 $PWD/a
$ cp a/boot/bbl-5.0.0-0.rc2.git0.1.0.riscv64.fc30.riscv64 .
$ guestunmount $PWD/a
```

```
$ sudo modprobe nbd max_part=8
$ sudo qemu-nbd -f raw --connect=/dev/nbd1 $PWD/Fedora-Developer-
Rawhide-20190126.n.0-sda.raw
$ sudo fdisk -l /dev/nbd1
Disk /dev/nbd1: 7.5 GiB, 8053063680 bytes, 15728640 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: F0F4268F-1B73-46FF-BABA-D87F075DCCA5

Device      Start      End  Sectors  Size Type
/dev/nbd1p1  2048 15001599 14999552  7.2G Linux filesystem
$ sudo mount /dev/nbd1p1 a
$ sudo cp a/boot/initramfs-5.0.0-0.rc2.git0.1.0.riscv64.fc30.riscv64.img
.
$ sudo umount a
$ sudo qemu-nbd --disconnect /dev/nbd1
```

Note NBD is highly useful if you need to run `fdisk`, `e2fsck` (e.g. after VM crash and filesystem lock up), `resize2fs`. It might be beneficial to look into `nbdkit` and `nbd` packages.

🔗 Boot the image on virtual hardware

There are several options for booting the image on virtual hardware once you've prepared it following the steps above.

🔗 Boot with libvirt

Detailed instructions how to install libvirt: https://docs.fedoraproject.org/en-US/quick-docs/getting-started-with-virtualization/

Quick instructions for libvirt installation (tested on Fedora 30):

```
dnf group install --with-optional virtualization
systemctl enable --now libvirtd
```

When running RISC-V guests, it's usually a good idea to use the very latest versions of QEMU, libvirt and virt-manager: the `virt-preview` repository offers just that for Fedora users. To enable it, simply run:

```
dnf copr enable @virtmaint-sig/virt-preview
```

and update your system.

Assuming you have QEMU ≥ 4.0.0, libvirt ≥ 5.3.0 and virt-manager ≥ 2.2.0, the installation will be as simple as:

```
# virt-install \
  --name fedora-riscv \
  --arch riscv64 \
  --vcpus 8 \
  --memory 2048 \
  --os-variant fedora30 \
  --boot kernel=/var/lib/libvirt/images/bbl-
*.riscv64,initrd=/var/lib/libvirt/images/initramfs-
*.img,kernel_args="console=ttyS0 ro root=/dev/vda" \
  --import --disk path=/var/lib/libvirt/images/Fedora-Developer-Rawhide-
*-sda1.raw \
  --network network=default \
  --graphics none
```

If you want graphics rather than serial console, simply replace `--graphics none` with `--graphics spice`.

If you are stuck with older software (QEMU ≥ 2.12.0, libvirt ≥ 4.7.0), then you're going to need a more verbose command line:

```
# virt-install \
  --name fedora-riscv \
  --arch riscv64 \
  --machine virt \
  --vcpus 8 \
  --memory 2048 \
  --os-variant fedora30 \
  --boot kernel=/var/lib/libvirt/images/bbl-
*.riscv64,initrd=/var/lib/libvirt/images/initramfs-
*.img,kernel_args="console=ttyS0 ro root=/dev/vda" \
  --import --disk path=/var/lib/libvirt/images/Fedora-Developer-Rawhide-
*-sda1.raw,bus=virtio \
  --network network=default,model=virtio \
  --rng device=/dev/urandom,model=virtio \
  --channel name=org.qemu.guest_agent.0 \
  --graphics none
```

Additionally, when using older software components you won't get PCI support, and so enabling graphics will not be possible.

Either one of the commands above will automatically boot you into the console. If you don't want that add `--noautoconsole` option. You can later use `virsh` tool to manage your VM and get to console.

A quick reference of `virsh` commands:

- `virsh list --all` - list all VMs and their states
- `virsh console <name>` - connect to serial console (remember: `Escape character is ^]`)
- `virsh shutdown <name>` - power down VM (see above for more details)
- `virsh start <name>` - power up VM
- `virsh undefine <name>` - remove VM
- `virsh net-list` - list network (useful for the next command)
- `virsh net-dhcp-leases <network_name>` - list DHCP leases, `<network_name>` most likely will be `default`. This is useful when you want to get IPv4 and SSH to the VM.
- `virsh domifaddr <name>` - alternative for the above two commands, only shows IPv4 for one VM
- `virsh reset <name>` - hard reset VM
- `virsh destroy <name>` hard power down of VM

If you want to use `ssh user@virtualMachine` you can setup libvirt NSS module. See: https://wiki.libvirt.org/page/NSS_module

You might want also to setup logging for serial console (in case kernel panics or something else).

For this we will be using two commands: `virsh edit <name>` (modifying VM XML) and `virsh dumpxml <name>` (dump VM XML for backup). You need to modify `<serial>` node by adding `<log file='/var/log/libvirt/qemu/fedora-riscv-mymagicbox.serial.log'/>`. Then power down and power up the VM.

Alternatively you can use `--serial log.file=/.../whatever.serial.log` with `virt-install` command.

🔗 Boot under QEMU

You will get the best results if your QEMU version is 4.0.0 or newer, but any version since 2.12.0 will work.

```
qemu-system-riscv64 \
    -nographic \
    -machine virt \
    -smp 8 \
    -m 2G \
    -kernel bbl \
    -initrd initramfs-*.img \
    -object rng-random,filename=/dev/urandom,id=rng0 \
    -device virtio-rng-device,rng=rng0 \
    -append "console=ttyS0 ro root=/dev/vda" \
    -device virtio-blk-device,drive=hd0 \
    -drive file=Fedora-Developer-Rawhide-*-sda1.raw,format=raw,id=hd0 \
    -device virtio-net-device,netdev=usernet \
    -netdev user,id=usernet,hostfwd=tcp::10000-:22
```

Once machine is booted you can connect via SSH:

```
ssh -p 10000 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no
-o PreferredAuthentications=password -o PubkeyAuthentication=no
root@localhost
```

🔗 Boot under TinyEMU (RISCVEMU)

Note (2019 March 10): This is not supported anymore until TinyEMU is updated to support external initrd file. Please, use QEMU or libvirt/QEMU.

RISCVEMU recently (2018-09-23) was renamed to TinyEMU (https://bellard.org/tinyemu/).

TinyEMU allow booting Fedora disk images in TUI and GUI modes. You can experiment using JSLinux (no need to download/compile/etc) here: https://bellard.org/jslinux/

Below are instructions how to boot Fedora into X11/Fluxbox GUI mode.

Step 1. Compile TinyEMU:

```
wget https://bellard.org/tinyemu/tinyemu-2018-09-23.tar.gz
tar xvf tinyemu-2018-09-23.tar.gz
cd tinyemu-2018-09-23
make
```

Step 2. Setup for booting Fedora:

```
mkdir fedora
cd fedora
cp ../temu .

# Download pre-built BBL with embedded kernel
wget https://bellard.org/jslinux/bbl64-4.15.bin

# Create configuration file for TinyEMU
cat <<EOF > root-riscv64.cfg
/* VM configuration file */
{
    version: 1,
    machine: "riscv64",
    memory_size: 1400,
    bios: "bbl64-4.15.bin",
    cmdline: "loglevel=3 console=tty0 root=/dev/vda rw TZ=${TZ}",
    drive0: { file: "Fedora-Developer-Rawhide-*-sda1.raw" },
    eth0: { driver: "user" },
    display0: {
        device: "simplefb",
        width: 1920,
        height: 1080,
    },
    input_device: "virtio",
}
EOF

# Download disk image and unpack in the same directory
```

Step 3. Boot it.

```
./temu -rw root-riscv64.cfg
```

We need to use `-rw` if we want our changes to persist in disk image. Otherwise disk image will be loaded as read-only and all changes will not persist after reboot.

🔗 Boot the image on physical hardware

🔗 Install on the HiFive Unleashed SD card

These are instructions for the HiFive Unleashed board (https://www.sifive.com/products/hifive-unleashed/).

The disk image (above) is partitioned, but usually we need an unpartitioned ("naked") filesystem. There are several ways to get this, but the easiest is:

```
$ guestfish -a Fedora-Developer-Rawhide-*-sda.raw \
    run : download /dev/sda1 Fedora-Developer-Rawhide-*-sda1.raw
```

This creates a naked ext4 filesystem called `*-sda1.raw` . The naked ext4 filesystem can be copied over the second partition of the SD card.

You can also build a custom bbl+kernel+initramfs to boot directly into the SD card using these sources (https://github.com/rwmjones/fedora-riscv-kernel).

🔗 Install on the HiFive Unleashed using NBD server

Look at https://github.com/rwmjones/fedora-riscv-kernel in the `sifive_u540` branch. This is quite complex to set up so it's best to ask on the `#fedora-riscv` IRC channel.

🔗 Install Fedora GNOME Desktop on SiFive HiFive Unleashed + Microsemi HiFive Unleashed Expansion board

Detailed instructions are provided by Atish Patra from Western Digital Corporation (WDC). See their GitHub page for details and pictures: https://github.com/westerndigitalcorporation/RISC-V-Linux

So far two GPUs are confirmed to be working: Radeon HD 6450 and Radeon HD 5450.

🔗 Use the image

Once the system is booted, login as `root` with `riscv` as password.

X11 with Fluxbox can be started using: `startx /usr/bin/startfluxbox` . The disk image also includes awesome and i3 for testing. Dillo is available as a basic web browser (no javascript support) and pcmanfm as file manager.

To gracefully shutdown just type `poweroff` into console.

If you want less information being displayed during boot then add `quiet` into `kernel_args` line, e.g.: `console=ttyS0 ro quiet root=/dev/vda`