1 FreeBSD Kernel Development Setup

FreeBSD is an open source, Unix-like, operating system that has been actively developed since 1993. It is provided in several different formats but for our purposes we will set it up within a virtual machine rather than on actual hardware, as modern systems, including laptops, are fast enough to make kernel development and debugging practical in these environments.

There are several virtualization technologies available but we will be using QEMU qemu.org because it is fast, cross platform, and open source. In our example we will use a graphical system that wraps QEMU, UTM, which is available for MacOS.

When setting up our virtual machine we have several choices to make:

Architecture x86 or Apple Silicon (M1, M2, M3 etc.)

Number of CPUs At least 2, but usually 1/4 to 1/2 of those available on the system.

RAM Size 1G of RAM per CPU selected above

Hard Disk Size At least 8G but 16 or 32 is better.

Before starting the installation you will need to download a suitable image from the FreeBSD website www.freebsd.org. You *must* pick an installation image that matched the architecture of your actual hardware, such as the laptop you are running the virtualization on. While QEMU can emulate many architectures you want to be using virtualization, and not emulation. The Downloads page for FreeBSD lists releases in order from most recent to least recently supported. You shold always start with the latest release. From the *Installer* column pick the appropriate architecture for your system. If you are using an Intel based system then you will want to select amd64 and if you are using Apple Silicon then select aarch64. You will be presented with a list of images, compressed in xz format, or uncompressed. Select the compressed image of disc1, for example FreeBSD-13.2-RELEASE-arm64-aarch64-disc1.iso.xz. Once the download is complete, uncompress the image with the unxz command. You now have an ISO image that can be used by QEMU to install the operating system and its associated tools.

Once the system has rebooted, disconnect the CD/DVD ROM ISO from the Virtual Machine, otherwise the install process will start again. If you wind up at the installer prompt a second time, disconnect the CD/DVD ROM and then reboot the virtual machine again.

1.1 Creating a case sensitive volume.

Most Unix-like operating systems are built on case sensitive filesystems. The filesystems on macOS are case preserving but not case sensitive by default and this will be a problem if you try to clone the FreeBSD source repository onto the default volume. With the Disk Utility you can create a case sensitive volume to store your repositories.

Once you have your code cloned into the new volume you can mount the volume over NFS into the FreeBSD VM. You will need to log into the virtual machine to see which IPv4 address it has been assigned. In Figure 55 we see that our VM has an IP address of 192.168.68.2, which means that the macOS host is 192.168.68.1, note this down for later.

On the macOS host we set up a file called /etc/exports as shown in Figure 56

We next make sure the NFS daemon is running on the mac host.

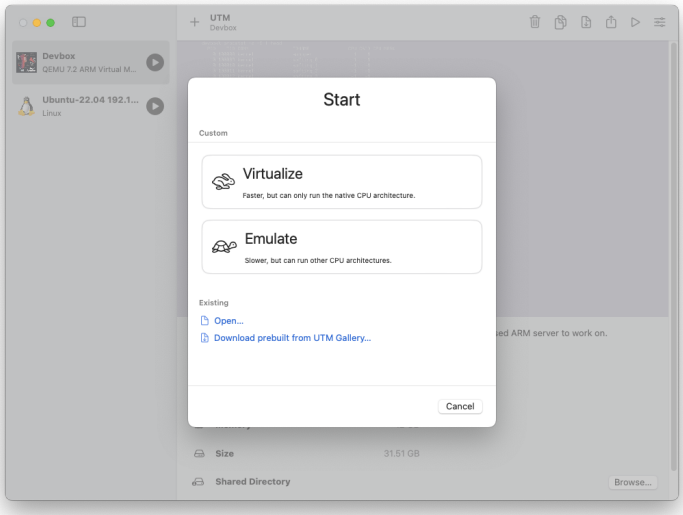

Figure 1: Seclect Virtualize

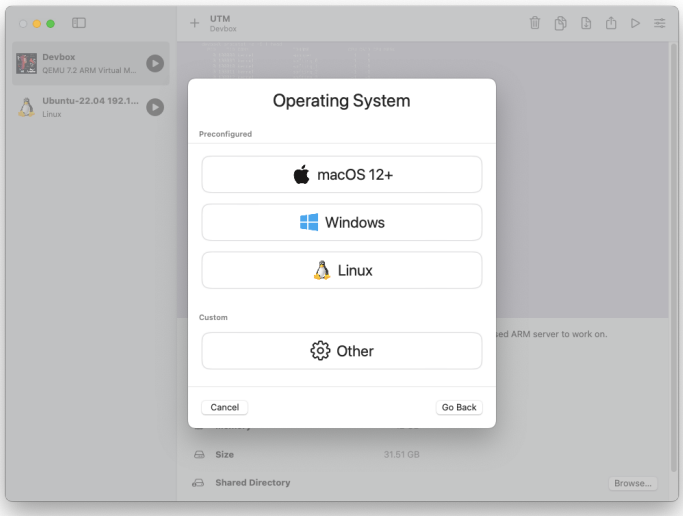

Figure 2: Select Other

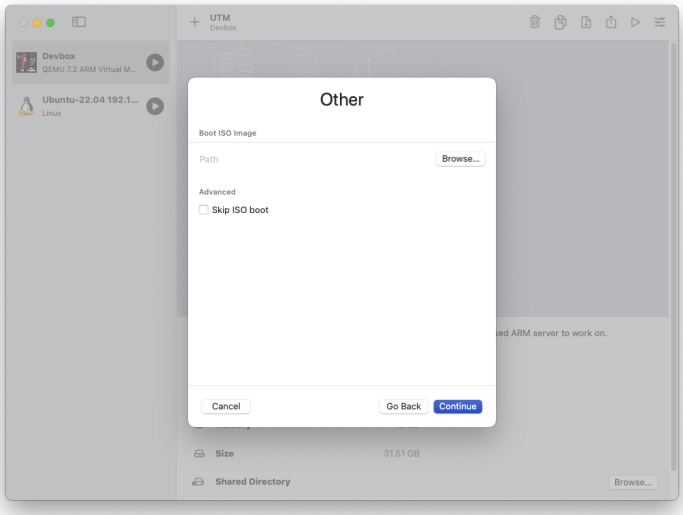

Figure 3: Browe to the ISO

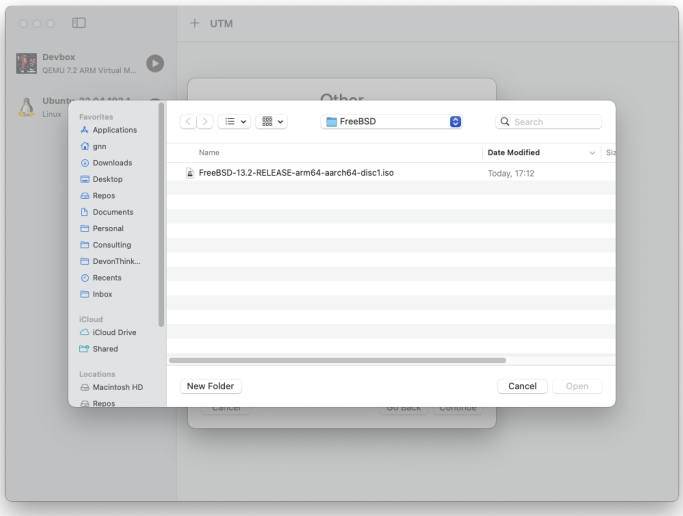

Figure 4: Select the ISO

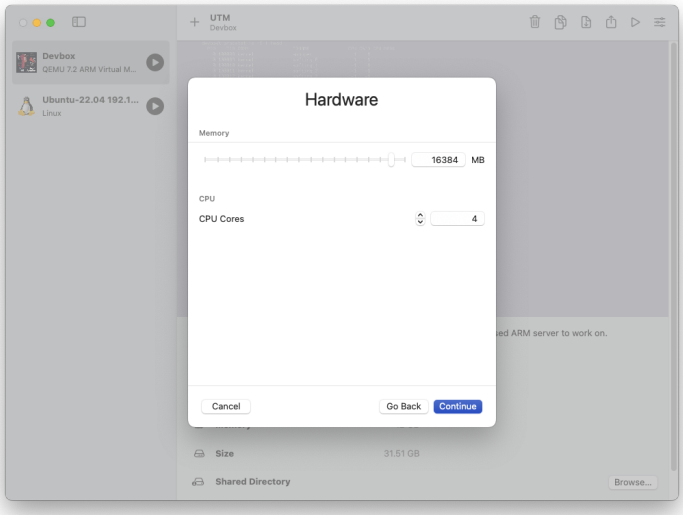

Figure 5: Select 16G of RAM and 4 cores

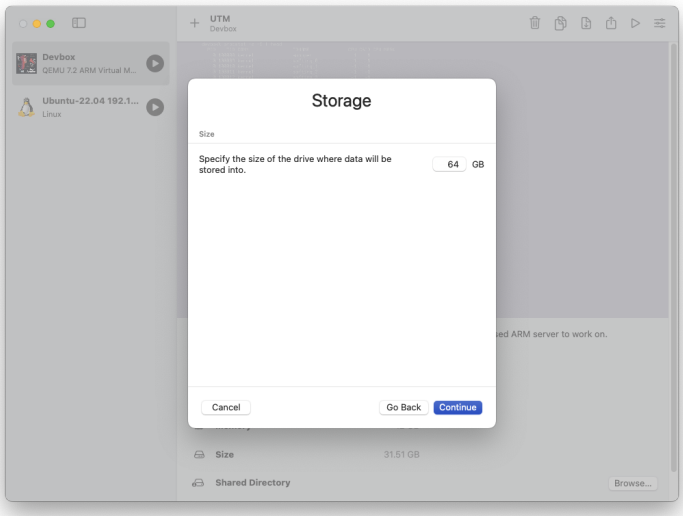

Figure 6: Select the default 64G disk size

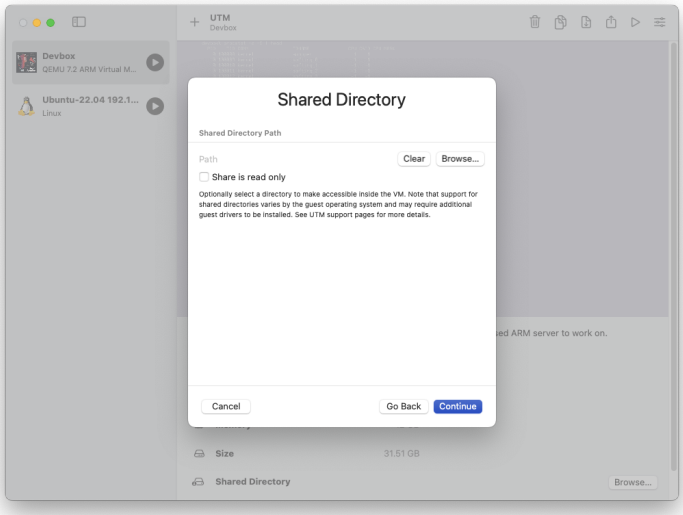

Figure 7: Do not select a Shared Directory

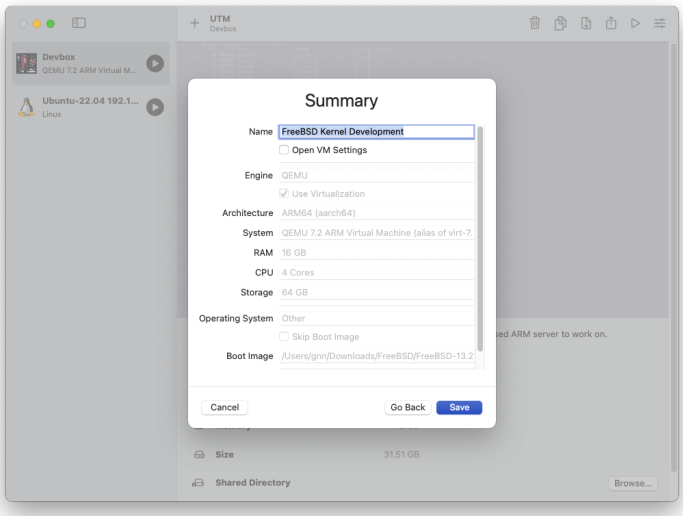

Figure 8: Name your Virtual Machine

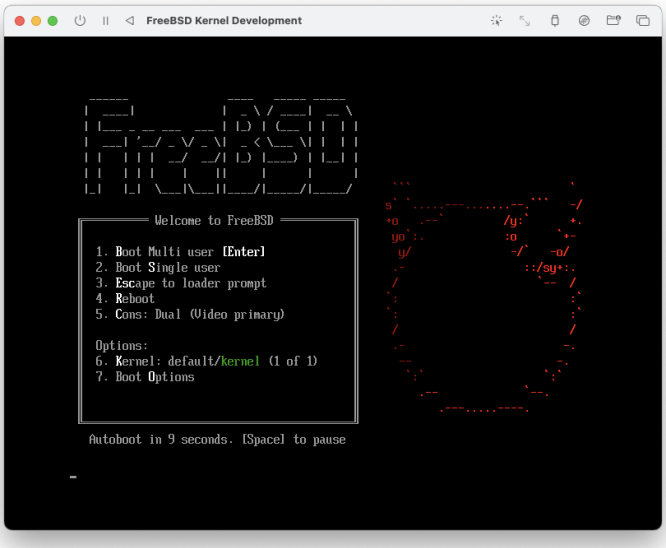

Figure 9: FreeBSD Installer Boot Screen

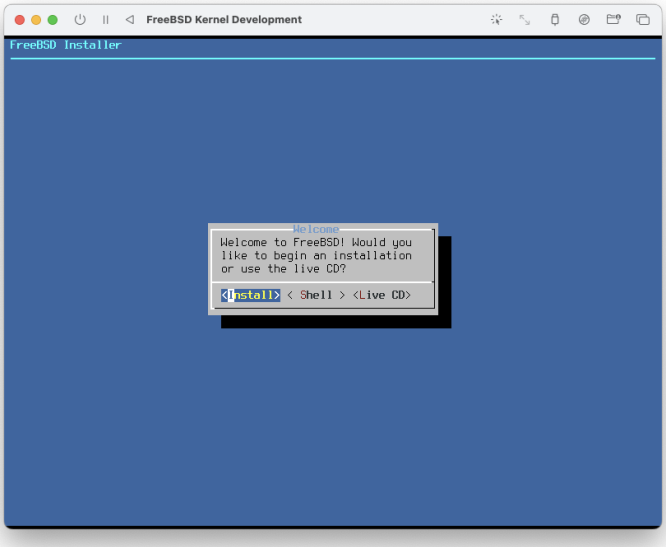

Figure 10: Selec the Install option.

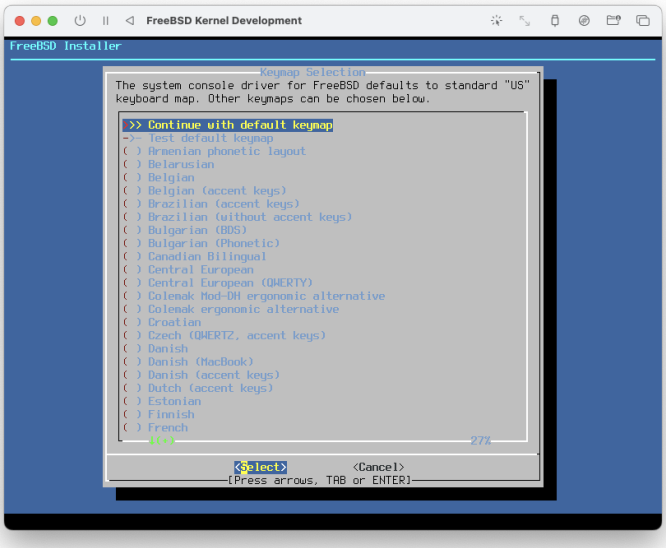

Figure 11: Continued with the default keymap

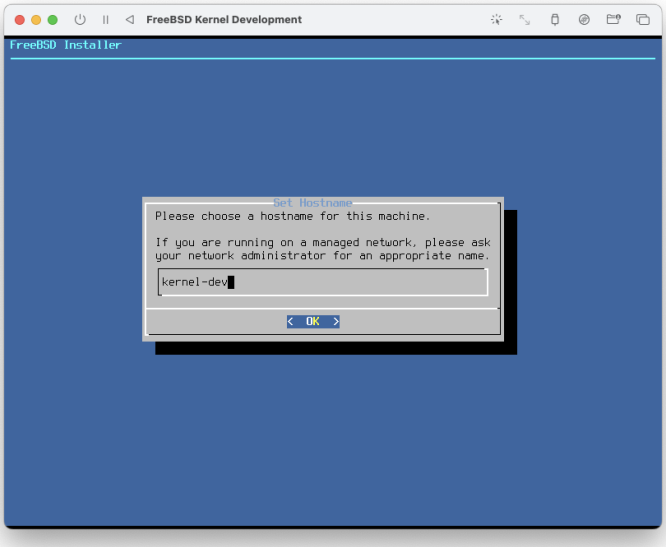

Figure 12: Name your host: kernel-dev

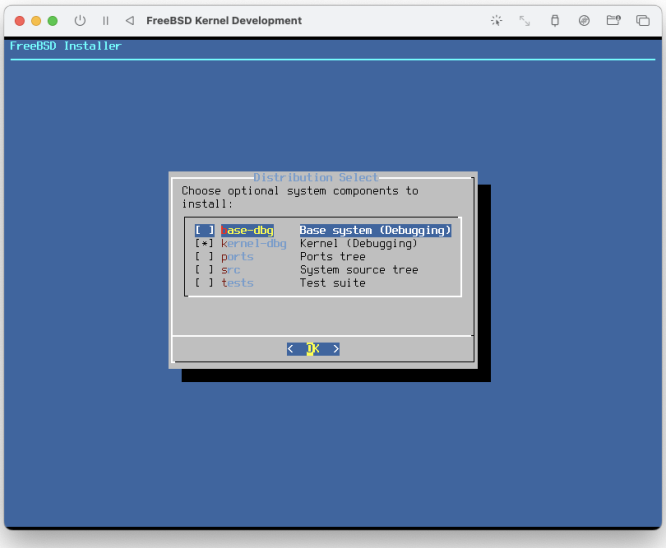

Figure 13: Go with the defaults

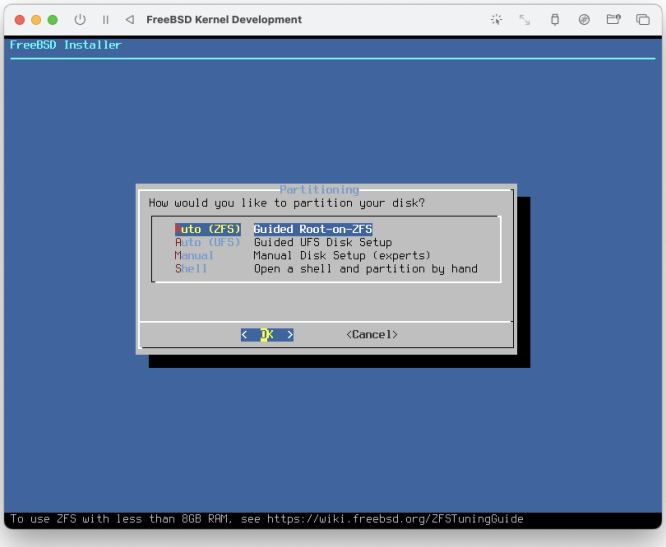

Figure 14: Select ZFS as your filesystem

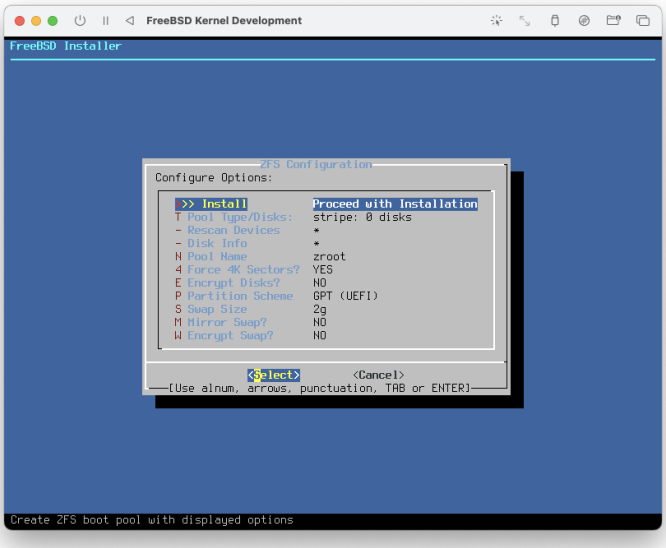

Figure 15: THe ZFS default are sufficient.

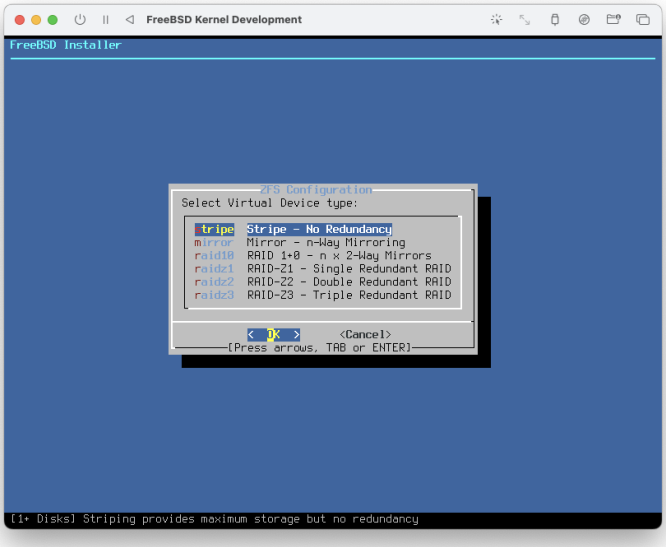

Figure 16: Stripe is all you can do with one disk

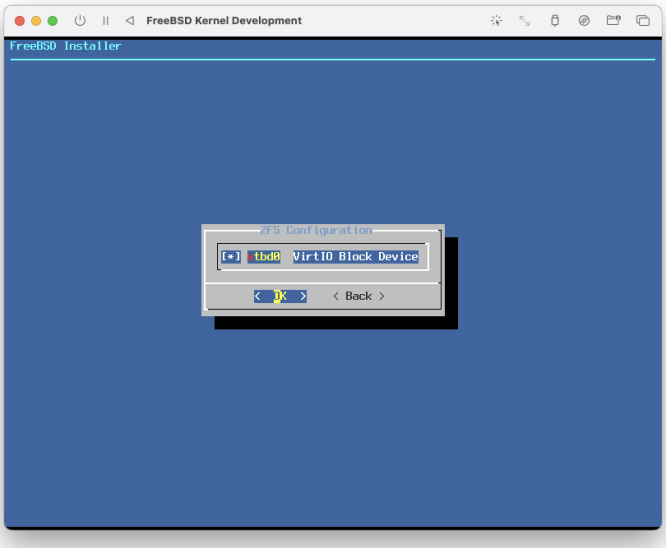

Figure 17: There is only one disk, select it.

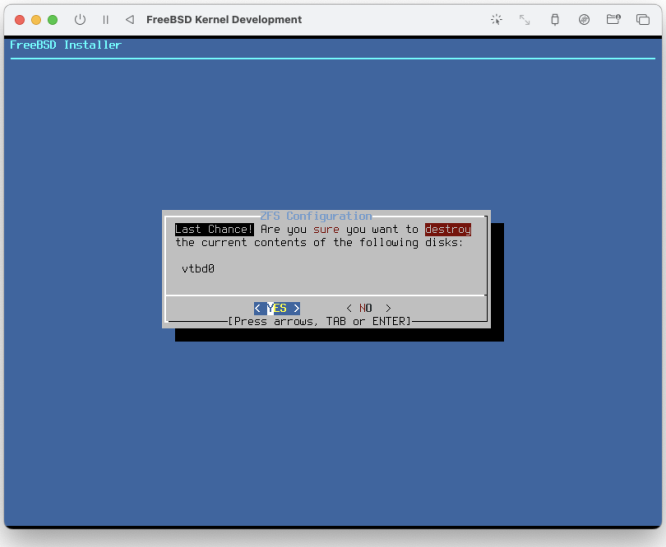

Figure 18: Last chance, select YES.

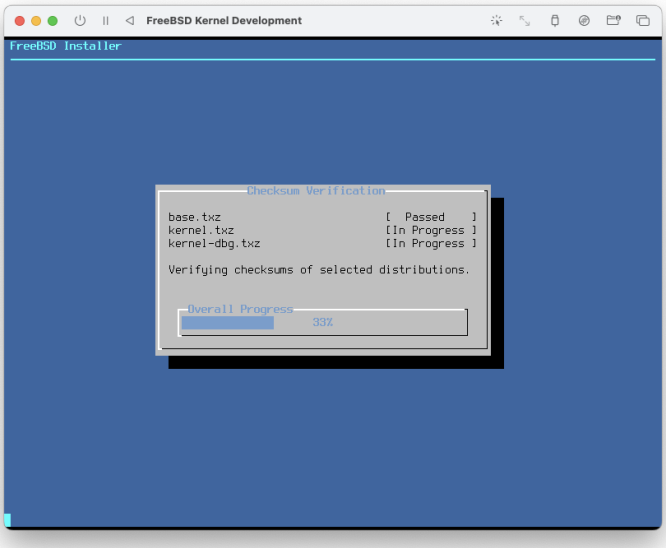

Figure 19: Installing

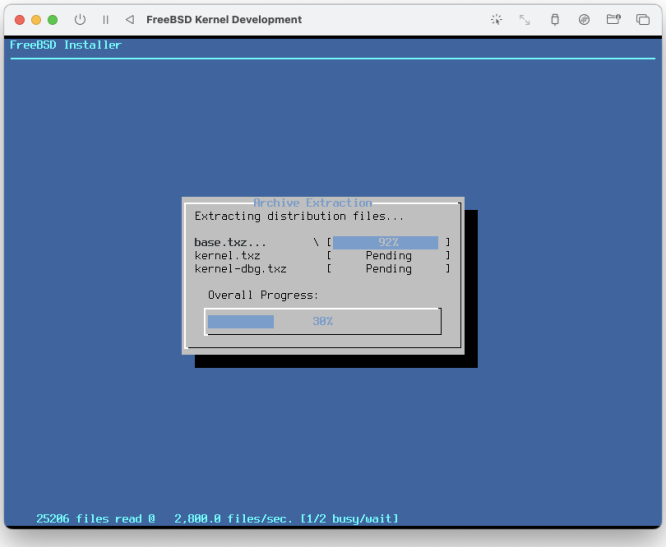

Figure 20: Extracting files

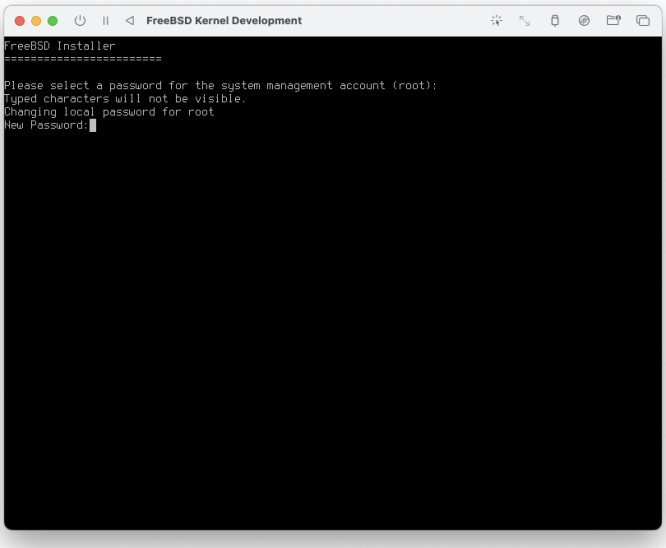

Figure 21: Select a root password

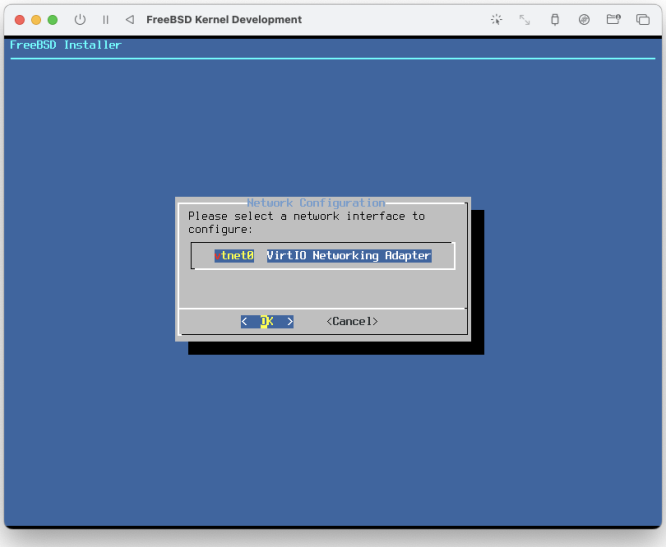

Figure 22: There is only one network device

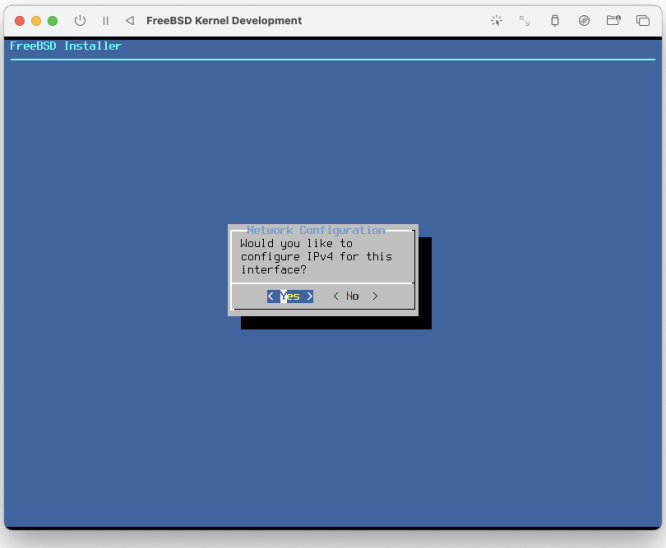

Figure 23: Use IPv4

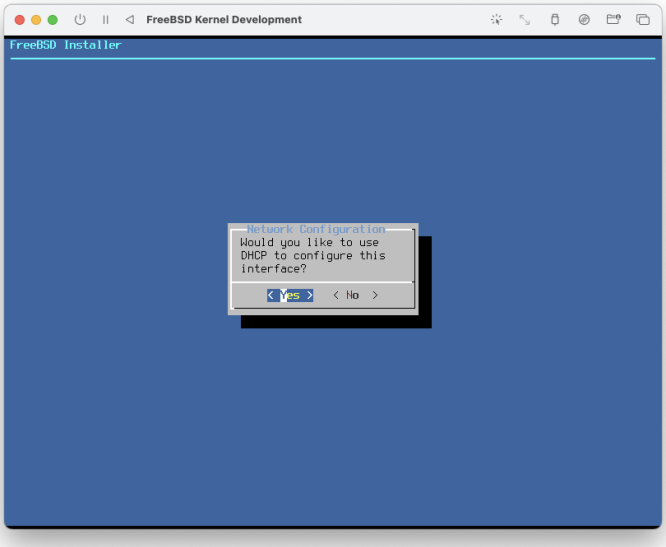

Figure 24: Use DHCP for IPv4

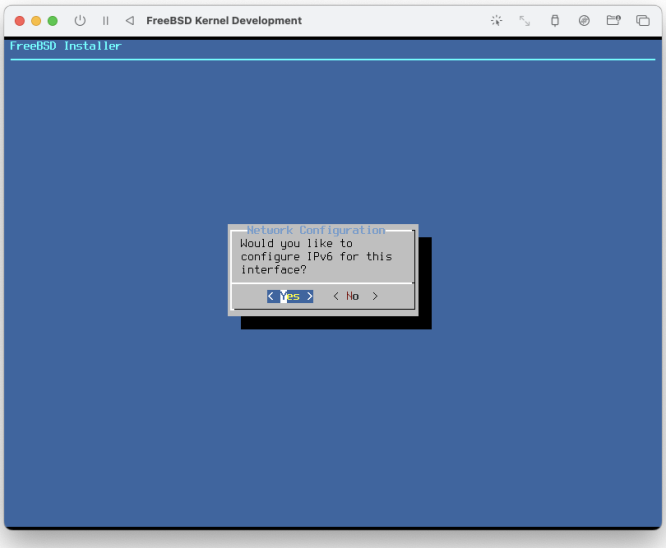

Figure 25: Do not configure IPv6

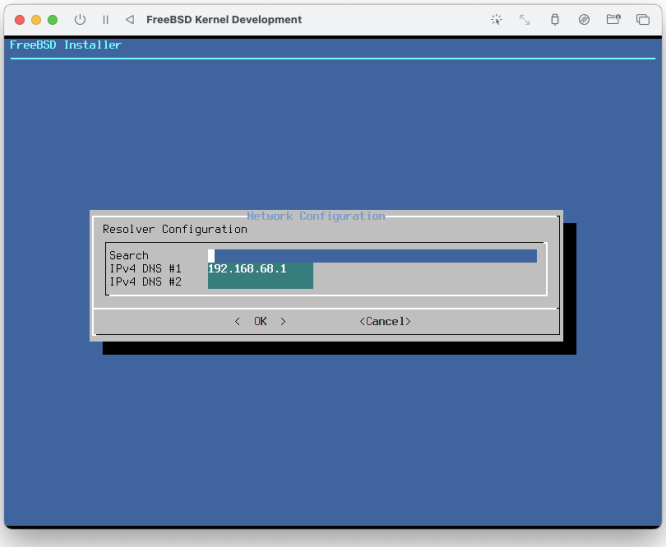

Figure 26: Accept the default DNS resolver settings

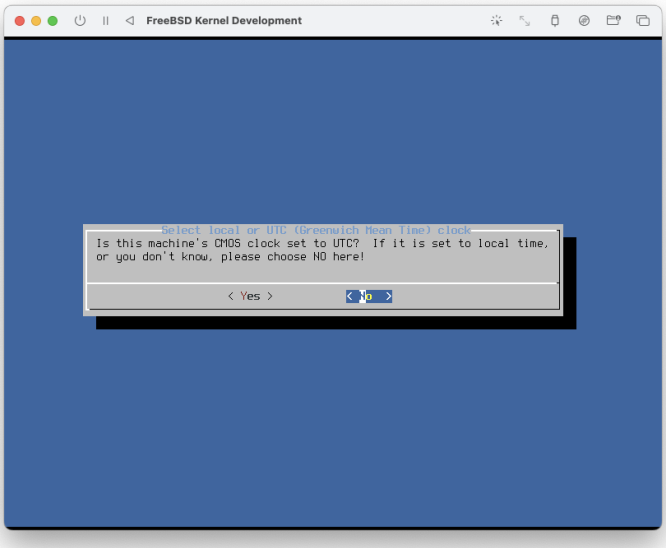

Figure 27: Start setting the clock, select NO

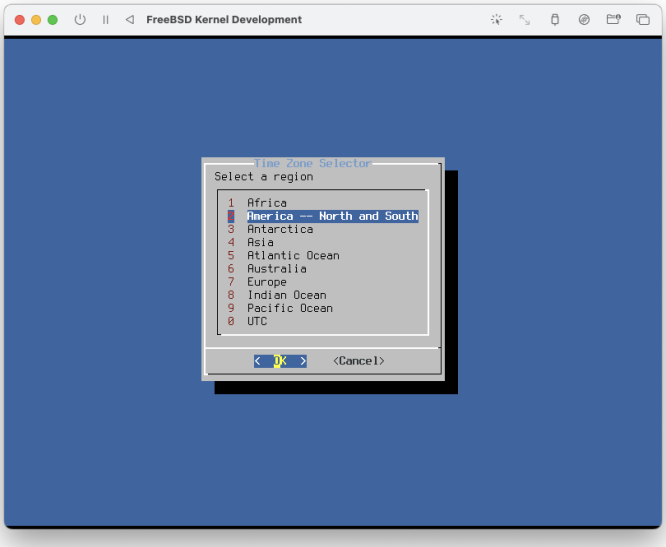

Figure 28: We're setting this up in America

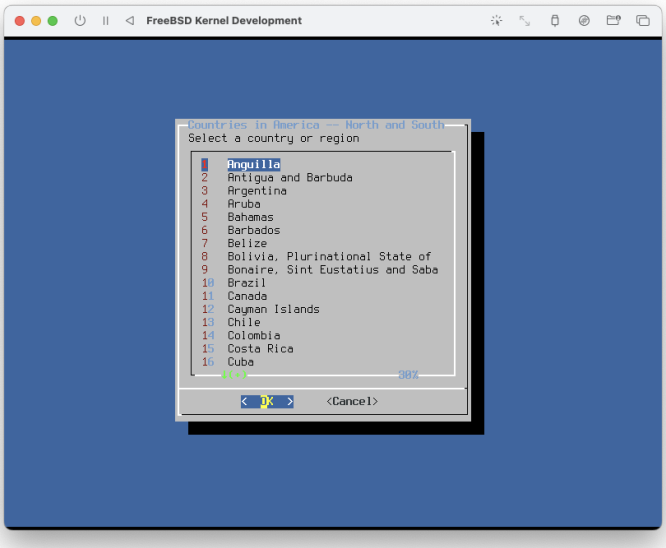

Figure 29: Scroll down to the correct country

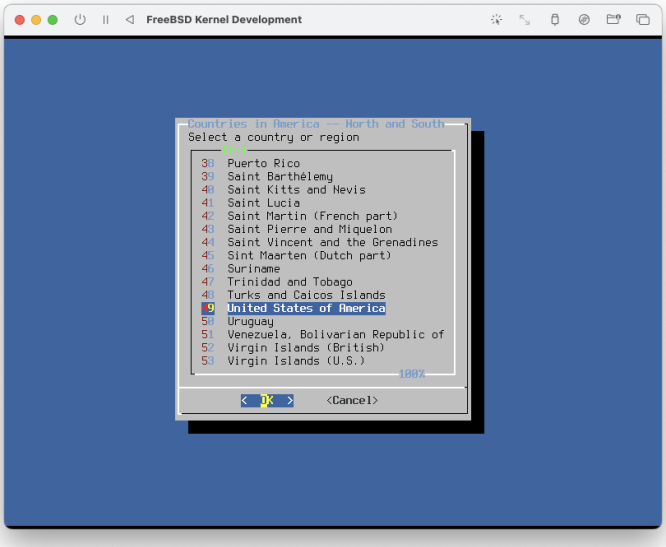

Figure 30: Select United States of America

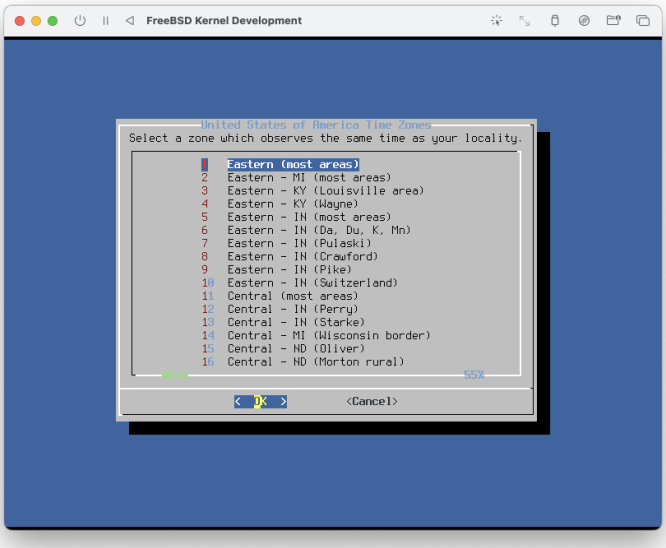

Figure 31: We are in the Eastern timezone

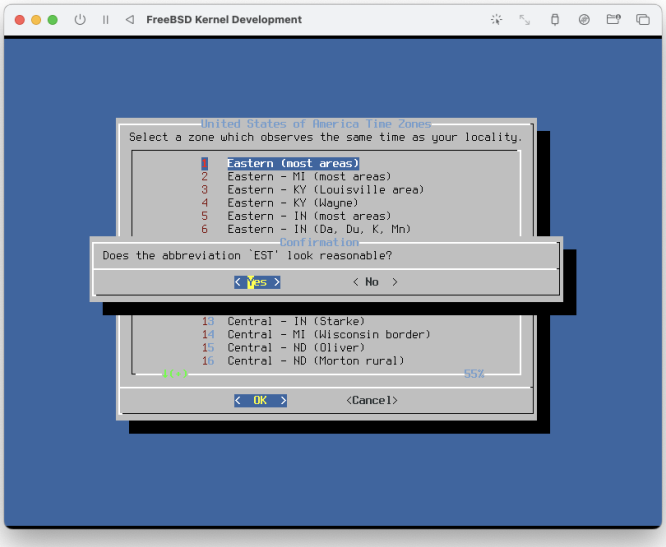

Figure 32: Accept EST

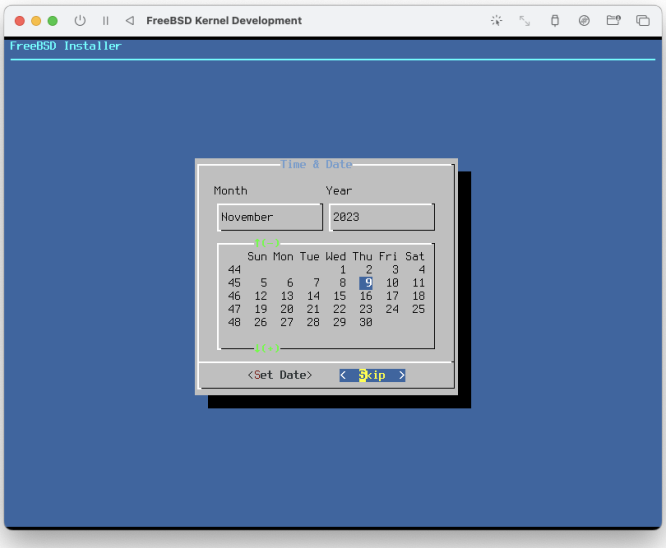

Figure 33: Skip setting the date (ntp will handle this on first boot)

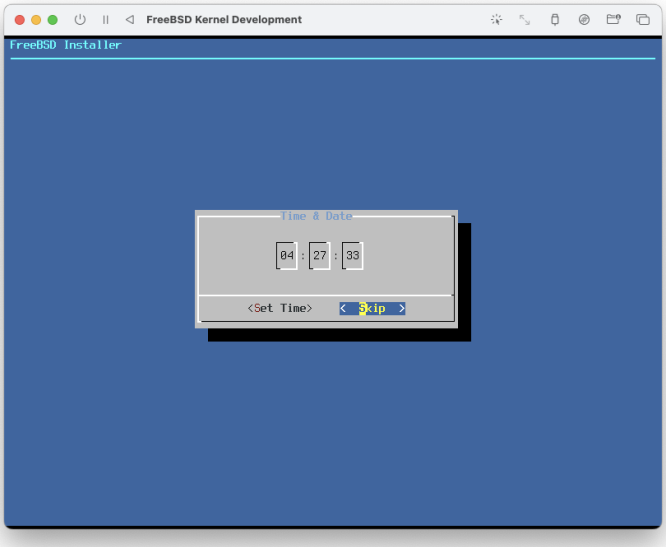

Figure 34: Skip setting the time (ntp will handle this on first boot)

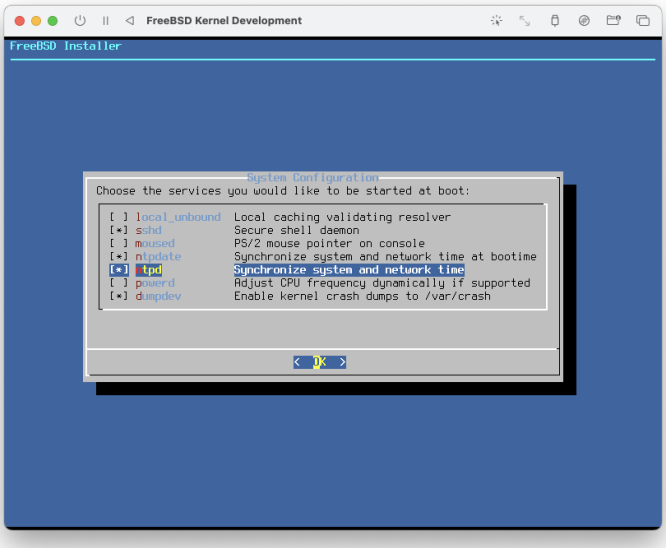

Figure 35: Select both ntpdate and ntpd

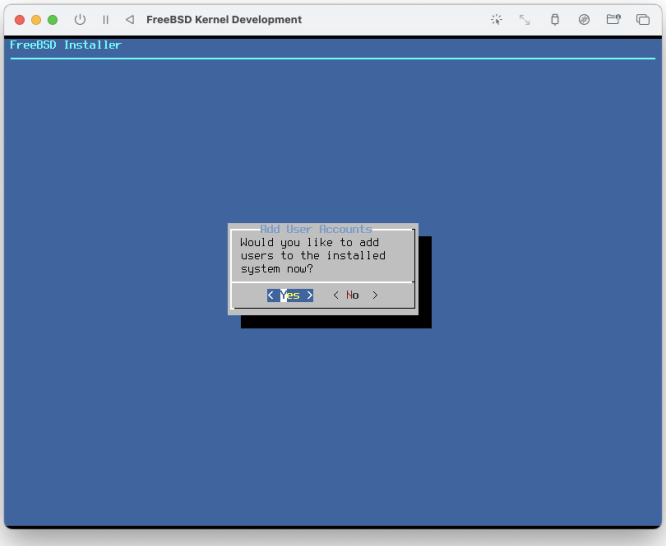

Figure 36: Add a user

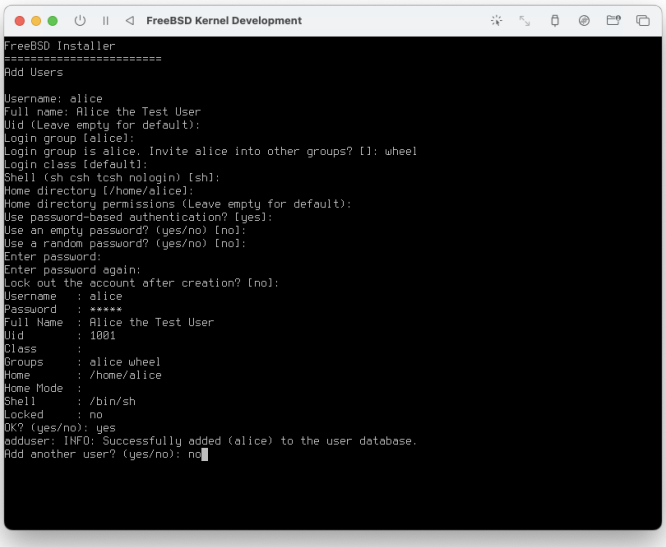

Figure 37: Add a user, example is Alice, use your own login

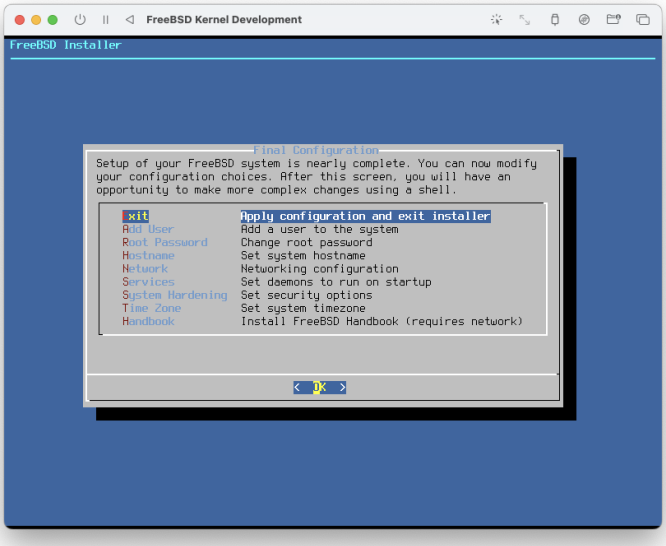

Figure 38: Nearly done, we can Exit

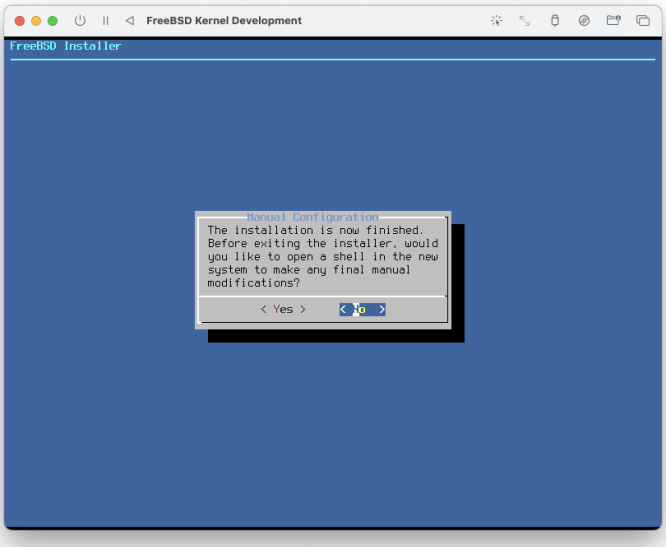

Figure 39: No need for manual modifications, select NO

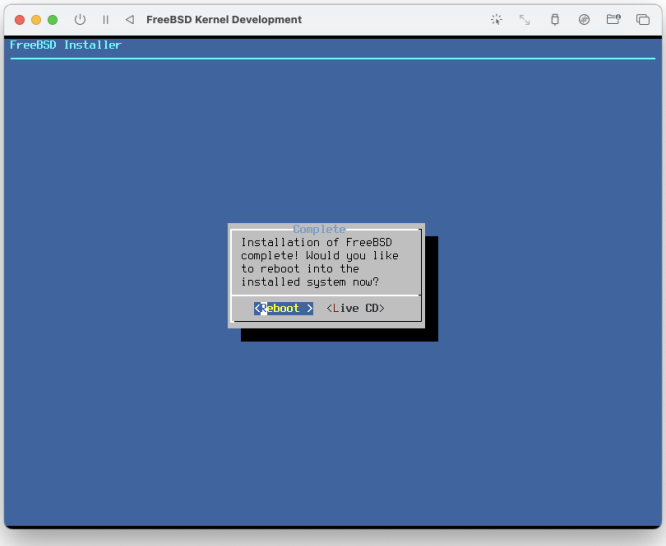

Figure 40: Reboot the system

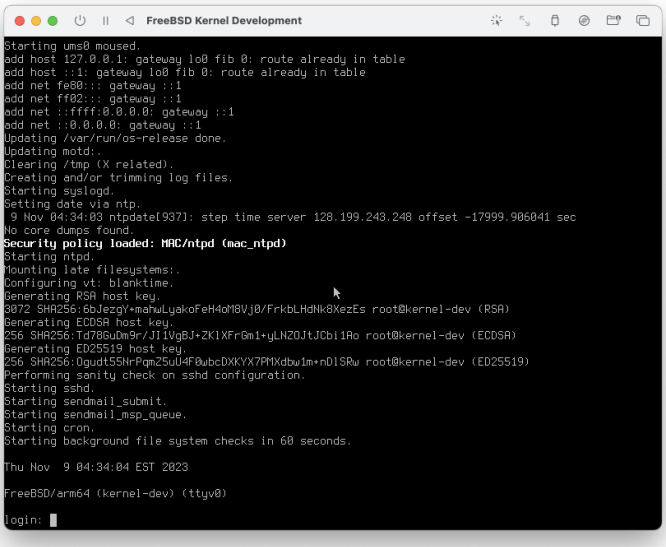

Figure 41: Normal boot to the login prompt.

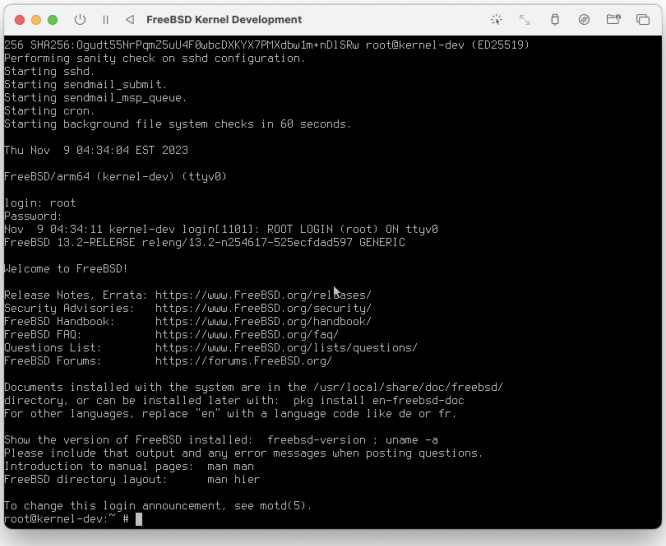

Figure 42: Login to the system as the root user

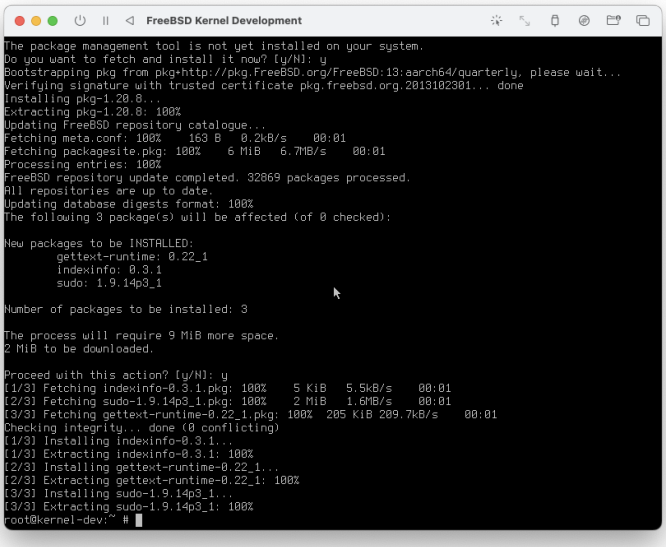

Figure 43: Use the pkg command to install sudo

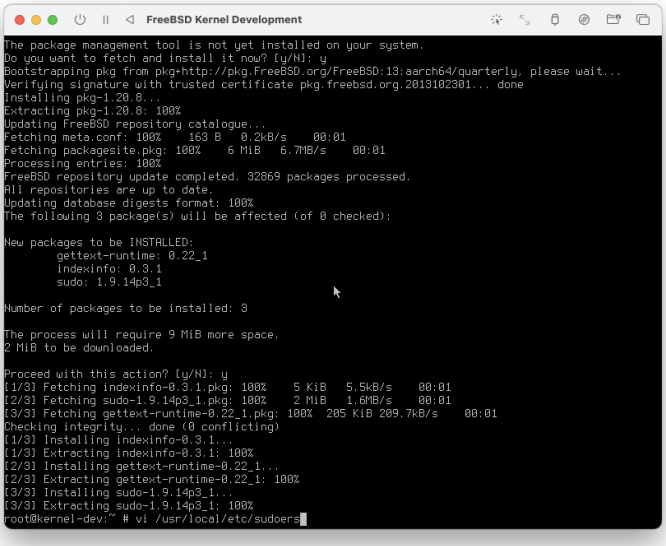

Figure 44: Edit the sudoers file with vi

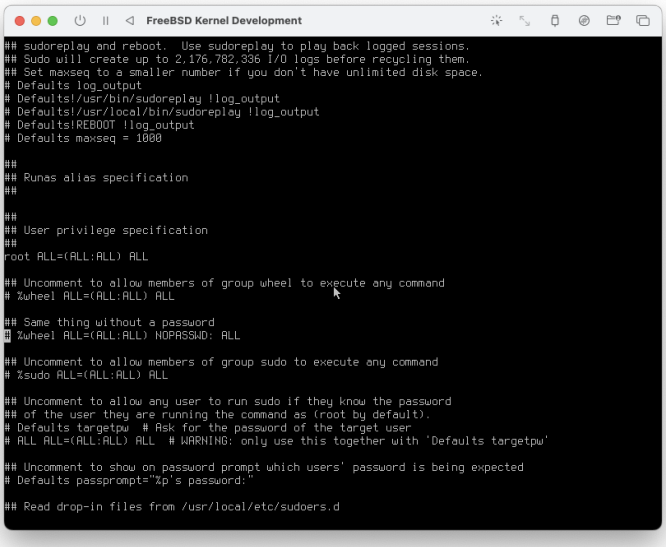

Figure 45: Go to the line to uncomment where wheel users can use sudo without a password.

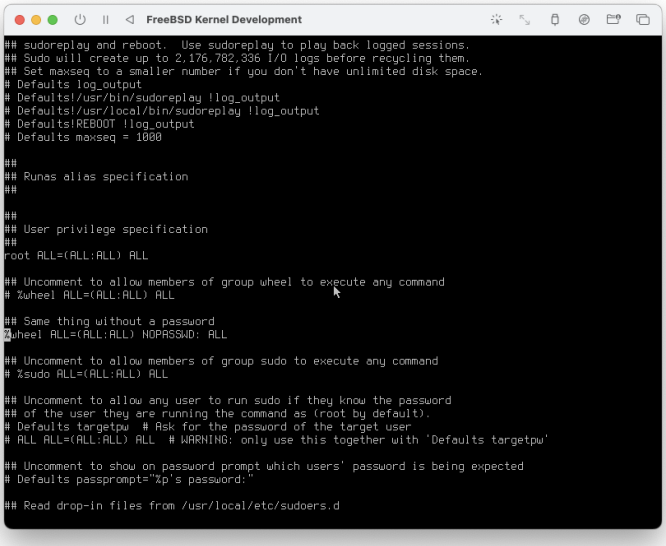

Figure 46: Remove the comment (#) character

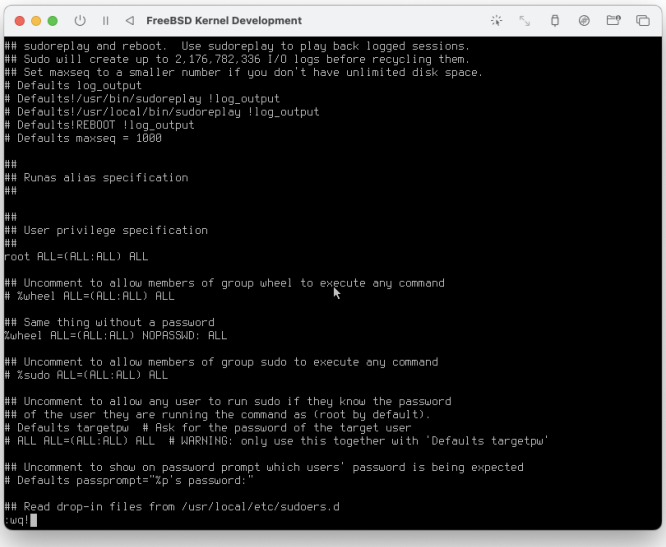

Figure 47: Save and quit vi with the :wq! command sequence

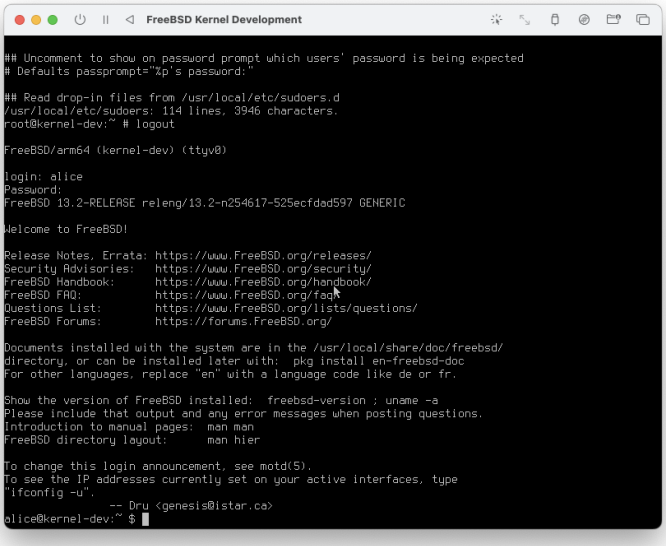

Figure 48: Log into the system as a normal user

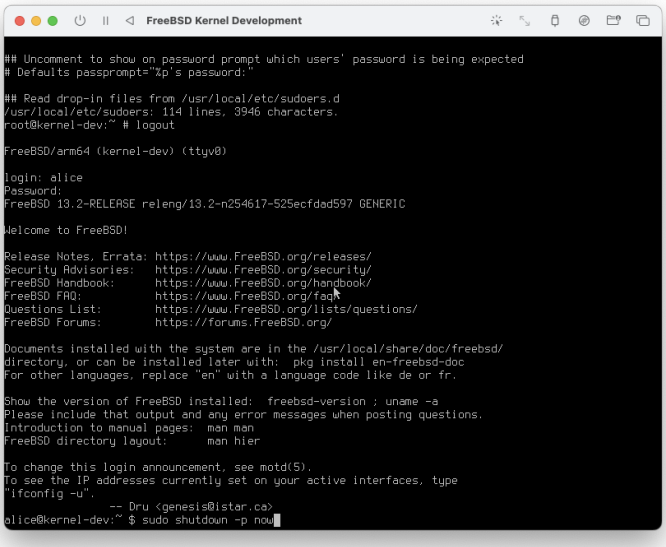

Figure 49: Issue a shutdown with sudo as a normal user

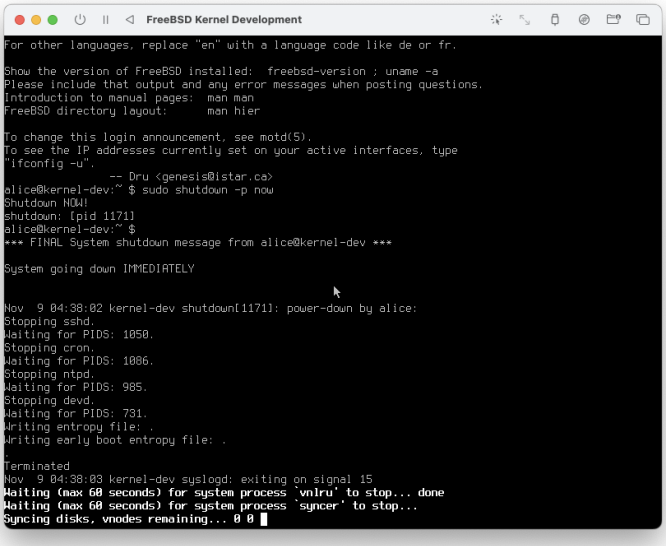

Figure 50: System starts shutting down

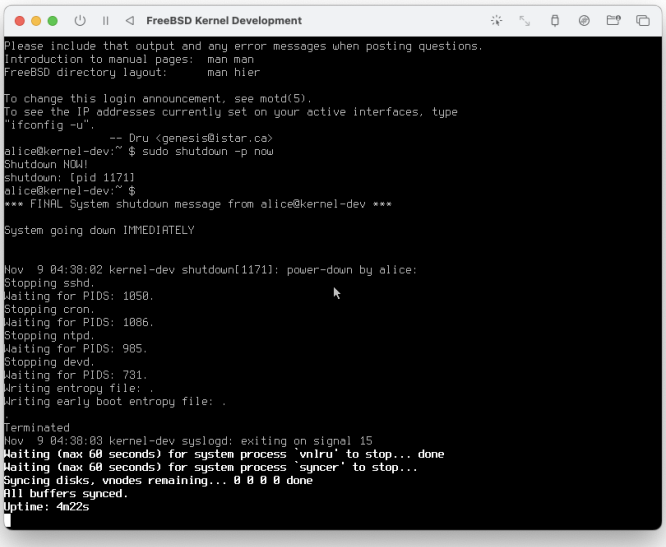

Figure 51: System shutdown complete, VM powers off

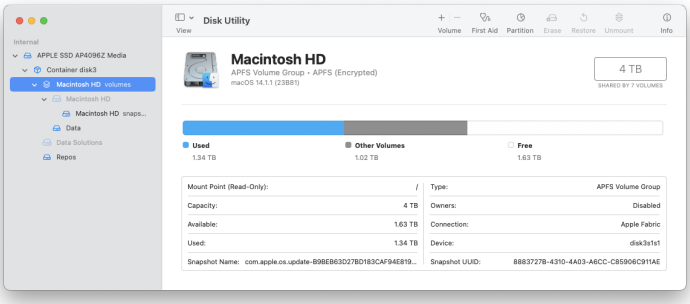

Figure 52: Open Disk Utility and Select the Macintosh Volume

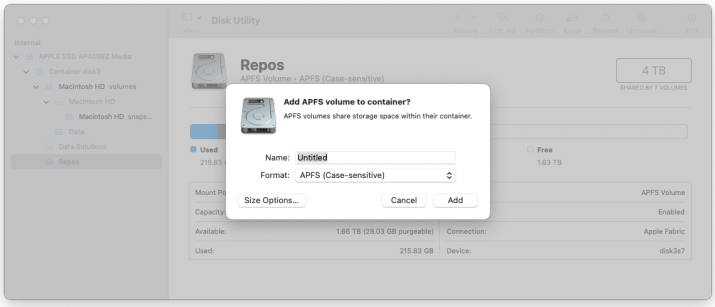

Figure 53: Add a new, case sensitive, volume

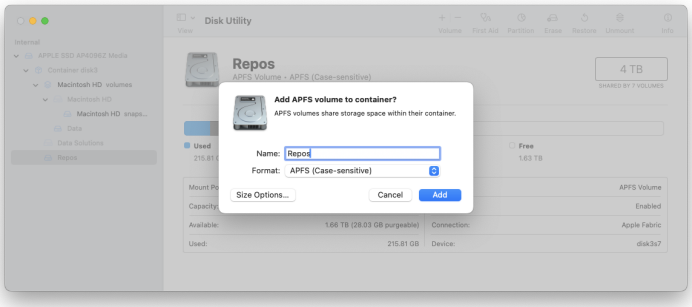

Figure 54: Give the volume a useful name

```
ifconfig
vtnet0: flags=1008843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST,LOWER_UP> metric 0 mtu 1500
options=80028<VLAN_MTU,JUMBO_MTU,LINKSTATE>
ether f6:d6:33:68:f7:0d
inet 192.168.68.2 netmask 0xffffff00 broadcast 192.168.68.255
media: Ethernet autoselect (10Gbase-T <full-duplex>)
status: active
nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
lo0: flags=1008049<UP,LOOPBACK,RUNNING,MULTICAST,LOWER_UP> metric 0 mtu 16384
options=680003<RXCSUM,TXCSUM,LINKSTATE,RXCSUM_IPV6,TXCSUM_IPV6>
inet 127.0.0.1 netmask 0xff000000
inet6 ::1 prefixlen 128
inet6 fe80::1%lo0 prefixlen 64 scopeid 0x2
groups: lo
nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>
```

Figure 55: Check the virtual machine's IP address

```
cat /etc/exports
/Volumes/Repos -alldirs -maproot=0:0
```

Figure 56: Setting up the exports

```
> sudo nfsd start
> sudo nfsd status
Password:
nfsd service is enabled
nfsd is running (pid 704, 8 threads)
```

Figure 57: Start NFS daemon

```
> sudo mount -v -t nfs 192.168.68.1:/Volumes/Repos/Yale/freebsd-src /usr/src
> df /usr/src
Filesystem                                        1K-blocks       Used      Avail Capacity  Mounte
192.168.68.1:/Volumes/Repos/Yale/freebsd-src 3902665360 2305897348 1596768012    59%    /usr/s
> ls /usr/src
ls /usr/src
CONTRIBUTING.md bin sbin
COPYRIGHT cddl secure
LOCKS contrib share
MAINTAINERS crypto stand
Makefile etc sys
Makefile.inc1 gnu targets
Makefile.libcompat include tests
Makefile.sys.inc kerberos5 tools
ObsoleteFiles.inc lib usr.bin
README.md libexec usr.sbin
RELNOTES release
UPDATING rescue
```

Figure 58: Mount the source volume

Finally we mount the Repos volume into the virtual machine.
You can now build a new kernel.

```
> cd /usr/src
> sudo make -j 8 buildworld >& /tmp/bw.out

Check the output log in /tmp.bw.out

> sudo make -j 8 buildkernel >& /tmp/bk.out

Check the kernel build log in /tmp/bk.out
```

Figure 59: Build the world and the kernel