

Getting Started with Kernel-based Virtual Machine (KVM)

Abstract (Part 1/5)

- Want to get started with Kernel-based Virtual Machine (KVM)?
- Want to run a virtual machine on your system using open source technologies?
- Want to interact with KVM virtual machines from command line interface (CLI)?

Abstract (Part 2/5)

In this tutorial, Leonard will be teaching people to familiarize themselves with KVM technologies, which allows virtual machines to run with near native performance.

Abstract (Part 3/5)

Participants must have a basic knowledge of how the Linux operating system works, and must have a recent Linux based operating system running on a portable computer to join this tutorial.

Abstract (Part 4/5)

We will be focusing on tasks such as creating, accessing, modifying, and deleting KVMs, primarily using CLI and if time permits, using a graphical user interface (GUI) too.

Abstract (Part 5/5)

At the end of this tutorial, participants are expected to know how to check if KVM is supported on their computer hardware and manage KVMs with confidence.

Agenda / Tasks / 90 Minutes

- Setup Kernel-based Virtual Machine (KVM)
- Manage KVM Using:
 - Command Line Interface (CLI)
 - Graphical User Interface (GUI)



Overview of Kernel-based Virtual Machine (KVM) (Part 1/2)

- An open source virtualization technology built into Linux®. Turn Linux into a hypervisor.
- Allows a host machine to run multiple, isolated virtual environments called guests or virtual machines (VMs).
- Available from Linux 2.6.20 or newer.

Overview of Kernel-based Virtual Machine (KVM) (Part 2/2)

- KVM converts Linux into a type-1 (bare-metal) hypervisor.
- Every VM is implemented as a regular Linux process, scheduled by the standard Linux scheduler, with dedicated virtual hardware components.

Overview of Hypervisor

- A hypervisor is software that creates and runs virtual machines (VMs).
- It is also called a virtual machine monitor (VMM).
- It isolates the hypervisor operating system and resources from the virtual machines and enables the creation and management of those VMs.

Overview of QEMU (Quick Emulator)

- QEMU (Quick Emulator) is part of the KVM experience being the userspace backend for it, but it also can be used for hardware without virtualization extensions by using its Tiny Code Generator (TCG) mode.

Why Virtual Machines Called Domain? (Part 1/2)

- Domain Θ is the first domain started by the hypervisor at boot, and will be running a Linux OS. This domain is privileged: it may access the hardware and can run the XenControlTools that manage other domains.

Why Virtual Machines Called Domain? (Part 2/2)

- These other domains are referred to as DomU's with the U standing for "user". They are unprivileged, and the equivalent to a guest system or guest virtual machine.

Task 1

Setup Kernel-based Virtual Machine (KVM)

Hardware Virtualization Support (Part 1/2)

- KVM requires a CPU with virtualization extensions.
 - Intel® Virtualization Technology (Intel® VT)
 - CPU flag is **vmx** (*Virtual Machine Extensions*).
 - AMD virtualization (AMD-V)
 - CPU flag is **svm** (*Secure Virtual Machine*).

Hardware Virtualization Support (Part 2/2)

```
egrep --count '^flags.*(vmx|svm)' /proc/cpuinfo
```

- If output is 0, your system does not support the relevant virtualization extensions or disabled on BIOS. You can still use QEMU/KVM, but the emulator will fall back to software virtualization, which is much slower.

Installing Virtualization Packages (Fedora)

```
dnf groupinfo virtualization
```

```
dnf group install \
    virtualization \
    --with-optional \
    --assumeyes
```

- See Fedora's [Installation Documentation](#).

Installing Virtualization Packages (Ubuntu)

```
# apt-get install \
bridge-utils \
qemu-kvm \
virt-manager
```

Installing Virtualization Packages (CentOS)

```
# yum install \
    libvirt \
    qemu-kvm \
    virt-install \
    virt-install \
    virt-manager
```

Enable `libvirtd` Service

- The `libvirtd` service is a server side daemon and driver required to manage the virtualization capabilities of the KVM hypervisor.
- Start `libvirtd` service and enable it on boot.

```
systemctl start libvirtd
```

```
systemctl enable libvirtd
```

Verify KVM Kernel Modules

- Verify that the KVM kernel modules are properly loaded.

```
lsmod | egrep 'kvm_(amd|intel)'
```

- If output contains **kvm_intel** or **kvm_amd**, KVM is properly configured.

Append Groups to Manage KVM

- Append current user to `kvm` and `libvirt` groups to create and manage virtual machines.

```
usermod --append --groups=kvm,libvirt ${USER}
```

```
cat /etc/group | egrep "^(kvm|libvirt).*${USER}"
```

- Log out and log in again to apply this modification.

Update QEMU Configuration

```
# cp /etc/libvirt/qemu.conf /etc/libvirt/qemu.conf.original
# sed --in-place \
  "s,\#user = \"root\",#user = \"${USER}\",g" \
  /etc/libvirt/qemu.conf
# sed --in-place \
  "s,\#group = \"root\",#group = \"libvirt\",g" \
  /etc/libvirt/qemu.conf
# diff --unified \
  /etc/libvirt/qemu.conf.original \
  /etc/libvirt/qemu.conf
systemctl restart libvirtd
```

Task 2

Manage KVM using Command Line Interface (CLI)

- Install Debian from Network.

Install Debian from Network

```
$ virt-install \
  --name Debian11 --os-variant debian11 --description 'Debian11' \
  --vcpus 2 --ram 2048 \
  --location \
  https://ftp.debian.org/debian/dists/stable/main/installer-amd64 \
  --network bridge=virbr0 \
  --graphics vnc,listen=127.0.0.1,port=5901 \
  --noreboot --noautoconsole \
  --extra-args 'console=ttyS0,115200n8 serial'
$ virt-viewer --connect qemu:///session --wait Debian11
```

View Serial Console Message

```
$ virsh console Fedora  
Connected to domain 'Fedora'  
Escape character is ^] (Ctrl + ])
```

Guest Virtual Machine States and Types (Part 1/2)

Several `virsh` commands are affected by the type of the guest virtual machine:

Transient or **Persistent**.

Guest Virtual Machine States and Types (Part 2/2)

During the life cycle of a virtual machine, libvirt will classify the guest as any of the following states:

Undefined , Shut off , Running , Paused , Saved

Display virsh Version

```
virsh version
```

```
virsh version --daemon
```

Connect to Hypervisor (Part 1/2)

```
virsh connect [hostname-or-URI] [--readonly]
```

The most commonly used URIs are:

qemu:///system , qemu:///session , lxc:///

Connect to Hypervisor (Part 2/2)

For example, establish a session to connect to your set of guest virtual machines (VMs), with you as the local user:

```
virsh connect qemu:///session
```

List Guest VM Connected to Hypervisor

```
virsh list --all
```

```
virsh list --inactive
```

Display Information about Hypervisor

```
virsh hostname
```

```
virsh sysinfo
```

Take Screenshot of Virtual Machine

```
virsh screenshot $<Domain-{Id, Name, Uuid}>  
[imagefilepath] [--screen screenID]
```

Example: virsh screenshot Debian11

Extra: Start Guest Virtual Machine

```
virsh start $<Domain-{Id, Name, Uuid}> [--  
console] [--paused] [--autodestroy] [--bypass-  
cache] [--force-boot]
```

Starts the \$<Domain-{Id, Name, Uuid}> that you already created and is currently in the inactive state.

Configuring a Virtual Machine to be Started Automatically at Boot

```
virsh autostart [--disable] $<Domain-  
{Id, Name, Uuid}>
```

Example: virsh autostart Debian11

Extra: Rebooting a Guest Virtual Machine

```
virsh reboot $<Domain-{Id, Name, Uuid}> [--mode  
    <RebootModeName>]
```

Example: virsh reboot Debian11 --mode initctl

Extra: Save Guest Virtual Machine's Configuration

```
virsh save [--bypass-cache] domain file [--xml  
string] [--running] [--paused] [--verbose]
```

Example: `virsh save Debian11 Debian11-
Configuration.xml --running`

Extra: Define Guest VM with XML File

```
virsh define $<Domain-{Id,Name,Uuid}>.xml
```

Example: `virsh define Debian11-Configuration.xml`

Extra: Extract Guest VM XML File

```
virsh save-image-dumpxml file --security-info
```

Example: `virsh save-image-dumpxml Debian11-Configuration.xml`

Extra: Edit Guest VM Configuration

```
virsh save-image-edit <file> [--running] [--  
paused]
```

Example: virsh save-image-edit Debian11-
Configuration.xml --running

Extra: Restore Guest Virtual Machine

```
virsh restore <file> [--bypass-cache] [--xml  
/path/to/file] [--running] [--paused]
```

Example: virsh restore Debian11-
Configuration.xml --running

Extra: Resuming a Guest Virtual Machine

```
virsh resume $<Domain-{Id, Name, Uuid}>
```

Display Host Physical Machine Name

```
virsh domhostname $<Domain-{Id,Name,Uuid}>
```

Display Guest VM General Information

```
virsh dominfo $<Domain-  
{Id,Name,Uuid}>d/Domain-Name/Uuid}
```

Display Guest VM's ID Number

```
virsh domid $<Domain-{Id, Name, Uuid}>
```

Extra: Abort Running Jobs on a Guest VM

```
virsh domjobabort $<Domain-{Id, Name, Uuid}>
```

List Statistic about Guest VM

```
virsh domjobinfo $<Domain-{Id, Name, Uuid}>
```

Display Guest Virtual Machine's Name

```
virsh domname $<Domain-{Id,Uuid}>
```

Display Virtual Machine's State

```
virsh domstate $<Domain-{Id, Name, Uuid}>
```

Display Connection State to the Virtual Machine

```
virsh domcontrol $<Domain-{Id, Name, Uuid}>
```

Shut Down Guest Virtual Machine

```
virsh shutdown $<Domain-{Id, Name, Uuid}> [ --  
mode modename]
```

Example: virsh shutdown Debian11 --mode acpi

Suspend Guest Virtual Machine

```
virsh suspend $<Domain-{Id, Name, Uuid}>
```

Example: virsh suspend Debian11

Reset Virtual Machine

```
virsh reset $<Domain-{Id, Name, Uuid}>
```

Example: virsh reset Debian11

Stop Running Guest Virtual Machine To Restart It Later

```
virsh managedsave $<Domain-{Id, Name, Uuid}> --  
bypass-cache --running | --paused | --verbose
```

Example: virsh managedsave Debian11 --running

Extra: Listing, Creating, Applying, and Deleting a Snapshot

```
qemu-img snapshot [ -l | -a snapshot | -c  
snapshot | -d snapshot ] filename
```

Remove and Delete a Virtual Machine

```
virsh undefine $<Domain-{Id,Name,Uuid}> [--  
managed-save] [storage] [--remove-all-storage]  
[--wipe-storage] [--snapshots-metadata] [--  
nvram]
```

Example: virsh undefine Debian11 --remove-all-storage

Force a Guest Virtual Machine to Stop

```
virsh destroy $<Domain-{Id,Name,Uuid}>
```

Example: virsh undefine Debian11 --remove-all-storage

Virtual Machine Termination

```
$ virsh shutdown Debian11 # Graceful Shutdown  
Domain 'Debian11' is being shutdown  
  
$ virsh destroy Debian11 # Force Shutdown  
Domain 'Debian11' destroyed  
  
$ virsh undefine Debian11  
Domain 'Debian11' has been undefined
```

Extra: Related Commands

```
$ virsh nodeinfo  
$ virsh edit  
  
$ virt-df  
$ virt-top  
$ virt-viewer  
  
$ virsh pool-list --all  
$ virsh pool-destroy  
$ virsh pool-undefine
```

Extra: List OS Variant

```
virt-install --os-variant list
```

Extra: Install Ubuntu from ISO Image

```
$ virt-install \
  --name Ubuntu --os-variant ubuntu22.04 --description 'Ubuntu' \
  --vcpus 2 --ram 2048 \
  --network bridge=virbr0,model=virtio \
  --graphics vnc,listen=127.0.0.1,port=5902 \
  --cdrom ~/Downloads/ubuntu-22.04-desktop-amd64.iso \
  --noreboot --noautoconsole
$ virt-viewer --connect qemu:///session --wait Ubuntu
```

Extra: Install Ubuntu from Network

```
$ virt-install \
  --name Ubuntu --os-variant ubuntu20.04 --description 'Ubuntu' \
  --vcpus 2 --ram 2048 \
  --location \
  http://archive.ubuntu.com/ubuntu/dists/focal/main/installer-amd64/ \
  --network bridge=virbr0,model=virtio \
  --graphics vnc,listen=127.0.0.1,port=5902 \
  --noreboot --noautoconsole \
  --extra-args='console=ttyS0,115200n8 serial edd=off'
$ virt-viewer --connect qemu:///session --wait Ubuntu
$ virsh console Ubuntu
```

Extra: Install Fedora from ISO Image

```
$ virt-install \
  --name Fedora --os-variant fedora36 --description 'Fedora' \
  --vcpus 2 --ram 2048 \
  --network bridge=virbr0,model=virtio \
  --graphics vnc,listen=127.0.0.1,port=5904 \
  --cdrom ~/Downloads/Fedora-Workstation-Live-x86_64-36-1.5.iso \
  --noreboot --noautoconsole
$ virt-viewer --connect qemu:///session --wait Fedora
```

Extra: Install Fedora from Network

```
$ virt-install \
  --name Fedora --os-variant fedora36 --description 'Fedora' \
  --vcpus 2 --ram 2048 \
  --location \
  https://download.fedoraproject.org/pub/fedora/linux/releases/36/Server/x86_64/os \
  --network bridge=virbr0,model=virtio \
  --graphics vnc,listen=127.0.0.1,port=5904 \
  --noreboot \
  --extra-args='console=ttyS0,115200n8 edd=off'
$ virsh console Fedora
```

Extra: Install AlmaLinux from ISO Image

```
$ virt-install \
  --name AlmaLinux --os-variant almalinux9 --description 'AlmaLinux' \
  --vcpus 2 --ram 3072 \
  --network bridge=virbr0,model=virtio \
  --graphics vnc,listen=127.0.0.1,port=5903 \
  --cdrom ~/Downloads/AlmaLinux-9.0-x86_64-dvd.iso \
  --noreboot --noautoconsole
$ virt-viewer --connect qemu:///session --wait AlmaLinux
```

Extra: Install AlmaLinux from Network

```
$ virt-install \
  --name AlmaLinux --os-variant almalinux9 --description 'AlmaLinux' \
  --vcpus 2 --ram 3072 \
  --location \
  https://almalinux.uib.no/9.0/BaseOS/x86_64/os/ \
  --network bridge=virbr0,model=virtio \
  --graphics vnc,listen=127.0.0.1,port=5905 \
  --noreboot \
  --extra-args='console=ttyS0,115200n8 edd=off'
$ virsh console AlmaLinux
```

Extra: Install CentOS from ISO Image

```
$ virt-install \
  --name CentOS --os-variant centos-stream9 --description 'CentOS' \
  --vcpus 2 --ram 3072 \
  --network bridge=virbr0,model=virtio \
  --graphics vnc,listen=127.0.0.1,port=5902 \
  --cdrom ~/Downloads/CentOS-Stream-9-latest-x86_64-dvd1.iso \
  --noreboot --noautoconsole
$ virt-viewer --connect qemu:///session --wait CentOS
```

Extra: Install CentOS from Network

```
$ virt-install \
  --name CentOS --os-variant centos-stream9 --description 'CentOS' \
  --vcpus 2 --ram 3072 \
  --location \
  https://mirror.netsite.dk/centos-stream/9-stream/BaseOS/x86_64/os/ \
  --network bridge=virbr0,model=virtio \
  --graphics vnc,listen=127.0.0.1,port=5904 \
  --noreboot \
  --extra-args='console=ttyS0,115200n8 edd=off'
$ virt-viewer --connect qemu:///session --wait CentOS
$ virsh console CentOS
```

Error: Refusing to Undefine

```
$ virsh undefine Ubuntu --remove-all-storage  
error: Refusing to undefine while domain managed save image exists
```

```
$ virsh managedsave-remove Ubuntu  
Removed managedsave image for domain 'Ubuntu'  
$ virsh undefine Ubuntu  
Domain 'Ubuntu' has been undefined
```

Error: Failed to Get MTU of Bridge

```
stderr=failed to get mtu of bridge `virbr0': No such device
```

```
# systemctl restart libvird  
$ brctl show  
bridge name      bridge id          STP enabled    interfaces  
virbr0           8000.525400a87247    yes
```

Error: Hangs on Probing EDD

```
Booting from Hard disk....  
Probing EDD (edd=off to disable)... ok
```

```
$ virt-install \  
...  
--extra-args='... edd=off'
```

Error: Failed to Get Domain

- Ensure that specified storage pool has correct permissions and path.

```
$ virsh pool-list --all  
$ virsh pool-info default  
$ virsh pool-dumpxml default  
$ virsh pool-dumpxml default \  
  | xmlstarlet sel --template --copy-of "/pool/target"  
$ virsh pool-dumpxml default \  
  | xmlstarlet sel --template --value-of "/pool/target/path"
```

Error: Cannot Access Storage File (UID:107, GID:107)

```
# cp /etc/libvirt/qemu.conf /etc/libvirt/qemu.conf.original
# sed --in-place \
  "s,\#user = \"root\",#user = \"${USER}\",g" \
  /etc/libvirt/qemu.conf
# sed --in-place \
  "s,\#group = \"root\",#group = \"libvirt\",g" \
  /etc/libvirt/qemu.conf
# systemctl restart libvirtd
```

Error: Missing 'Default' Network?

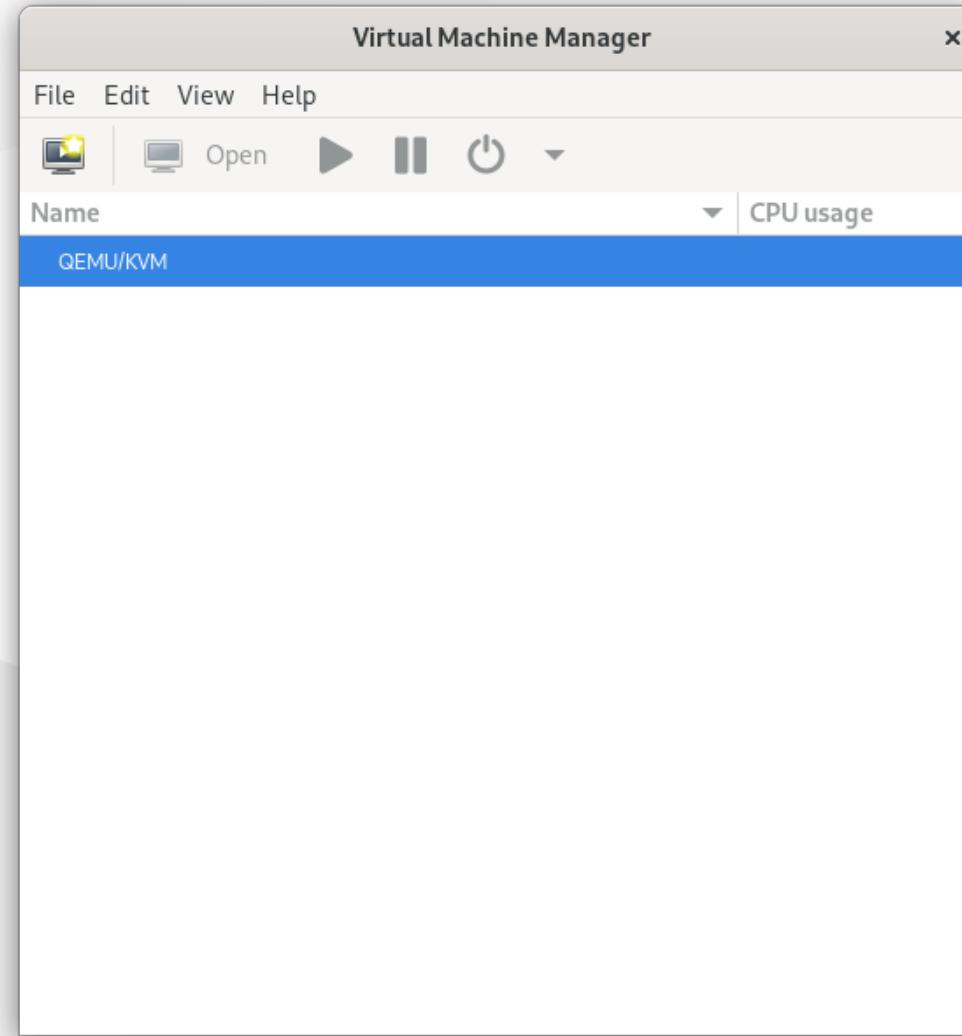
```
$ virsh net-list --all
Name      State      Autostart      Persistent
-----
$ sudo virsh net-list --all
Name      State      Autostart      Persistent
-----
default    active    yes            yes
```

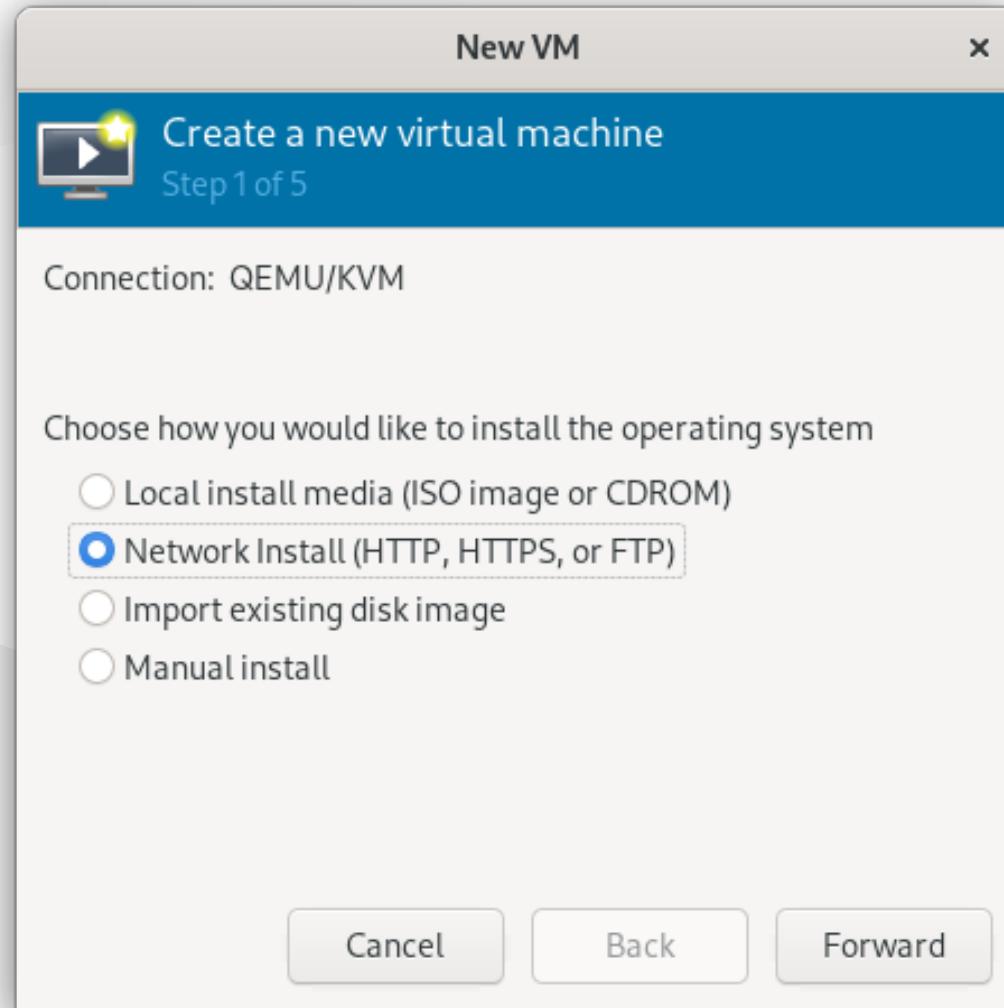
Read this post if default network is still missing.

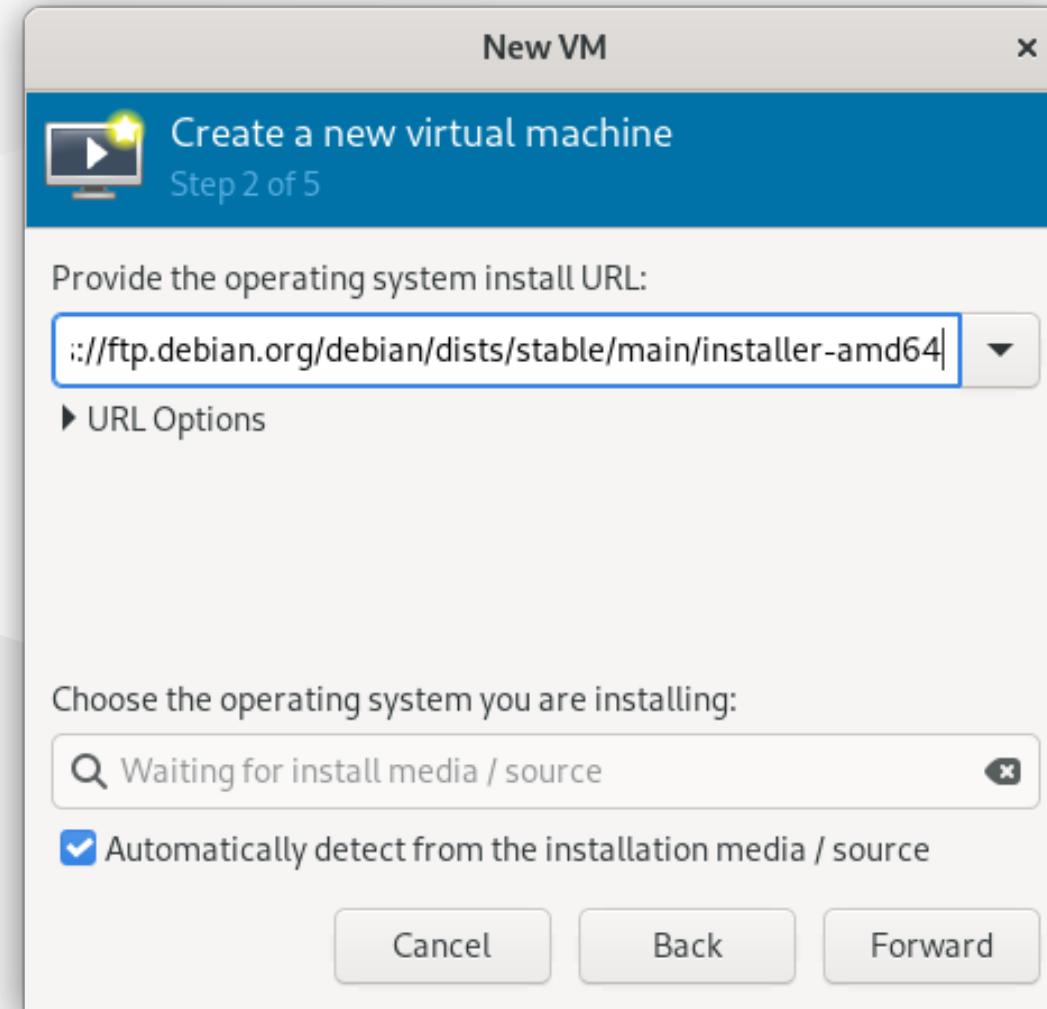
Task 3

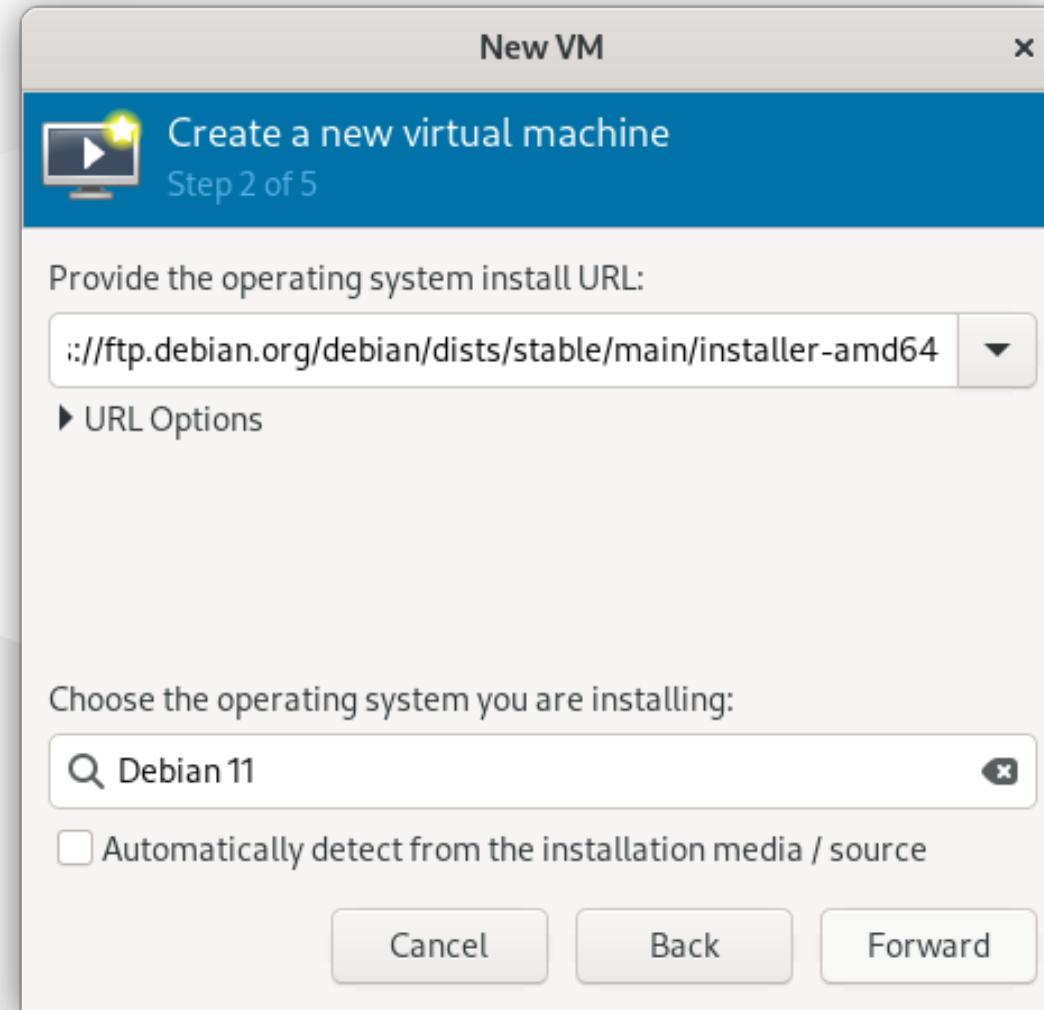
Manage KVM using Graphical User Interface (GUI)* (**if time permits*)

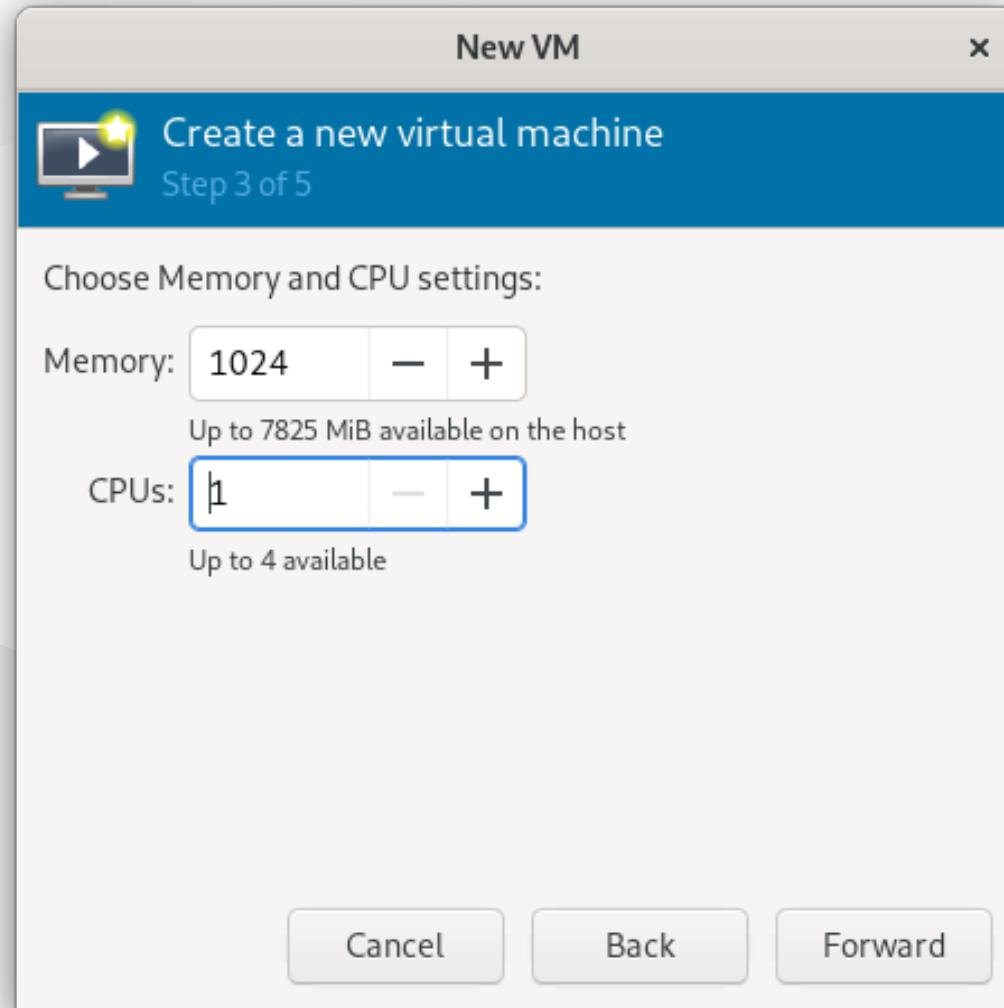
Use `virt-manager` to create, manage, & delete KVMs.

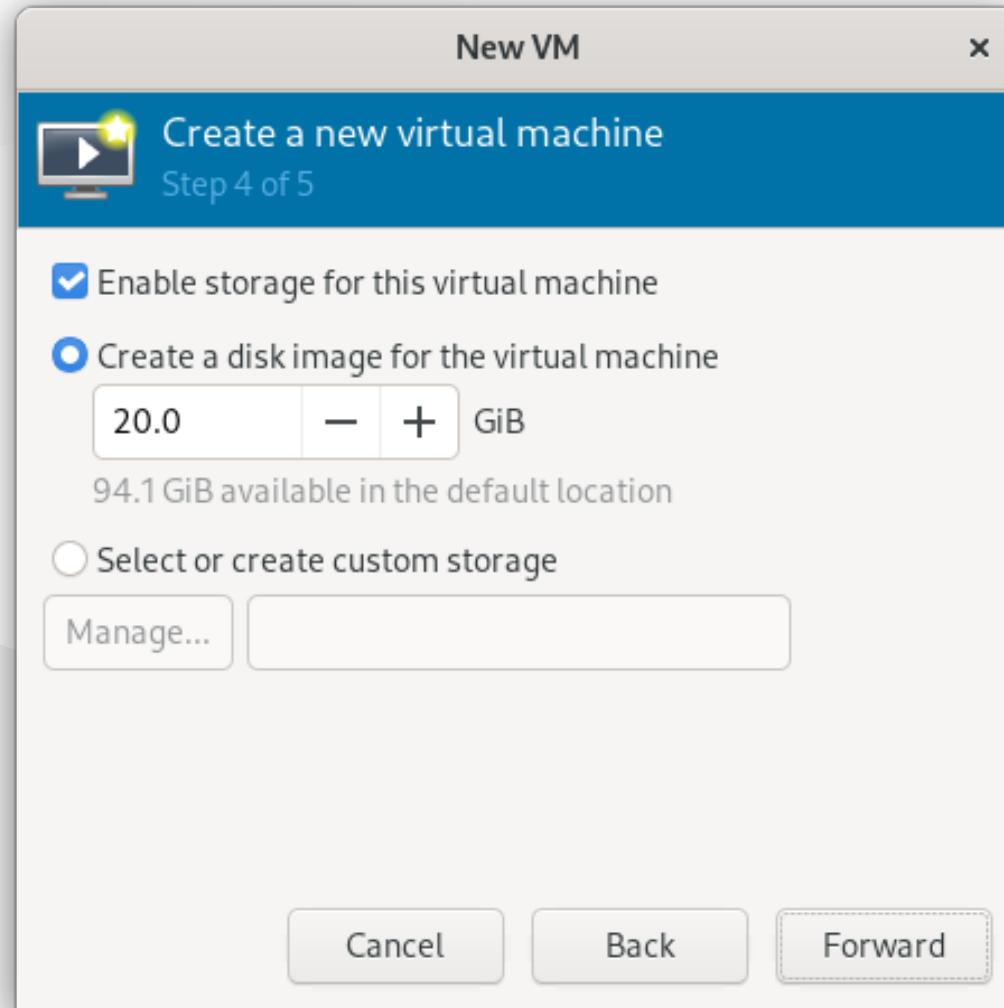


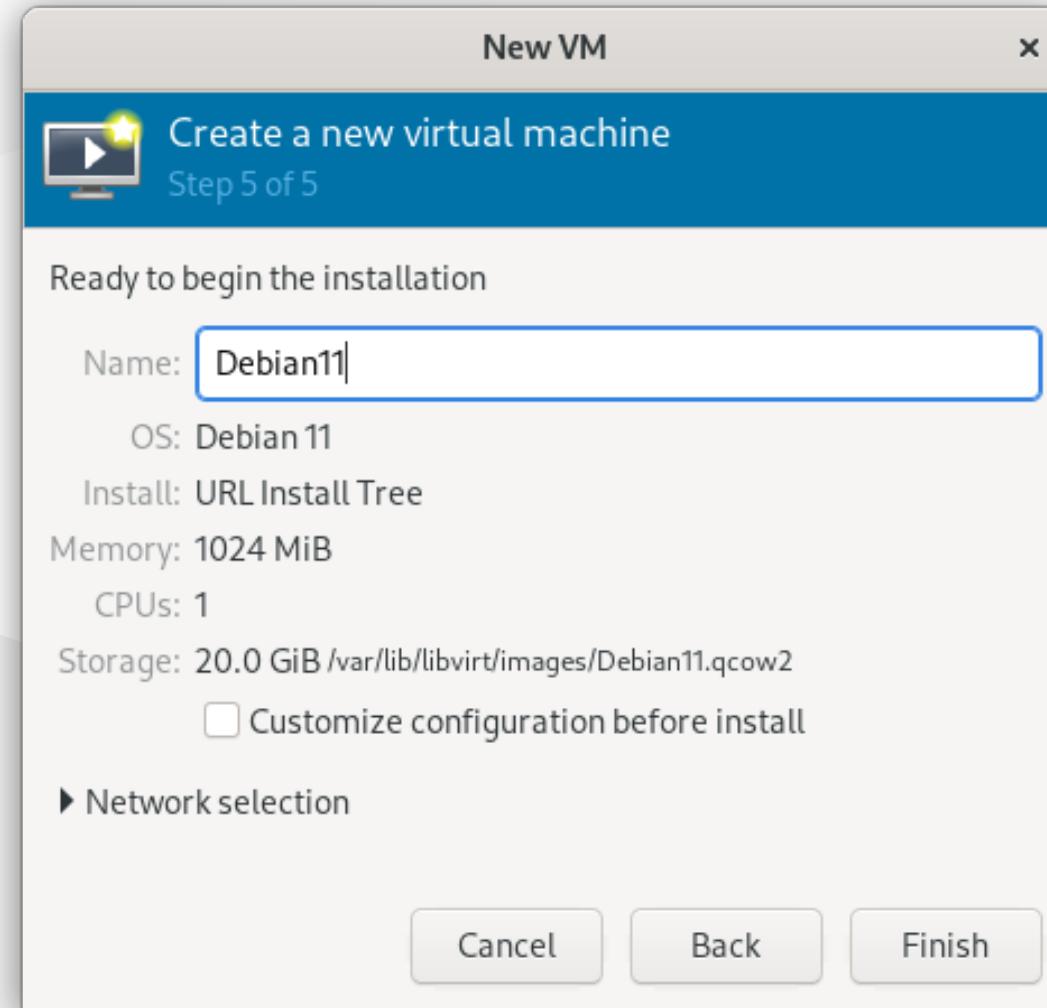












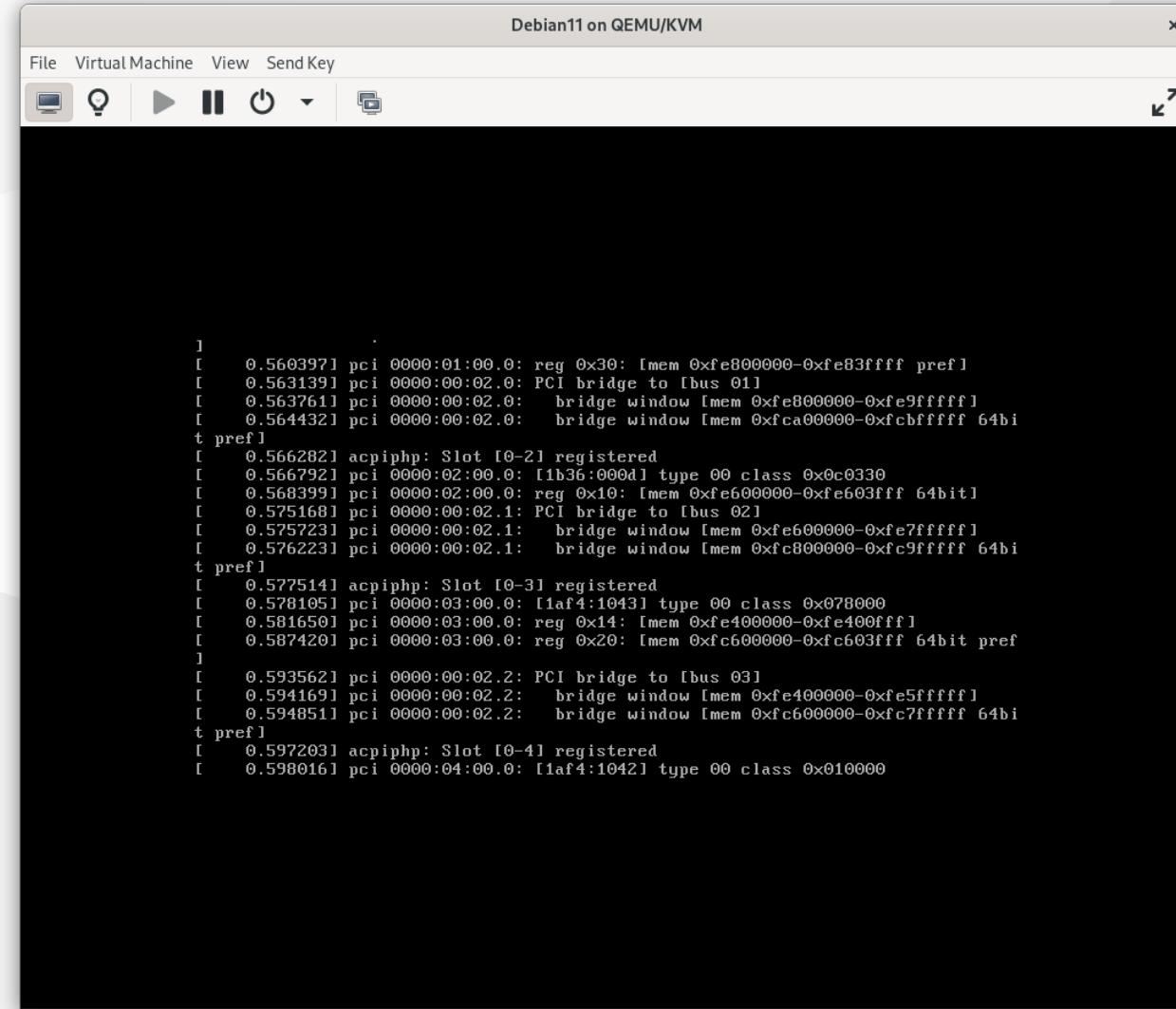
Creating Virtual Machine

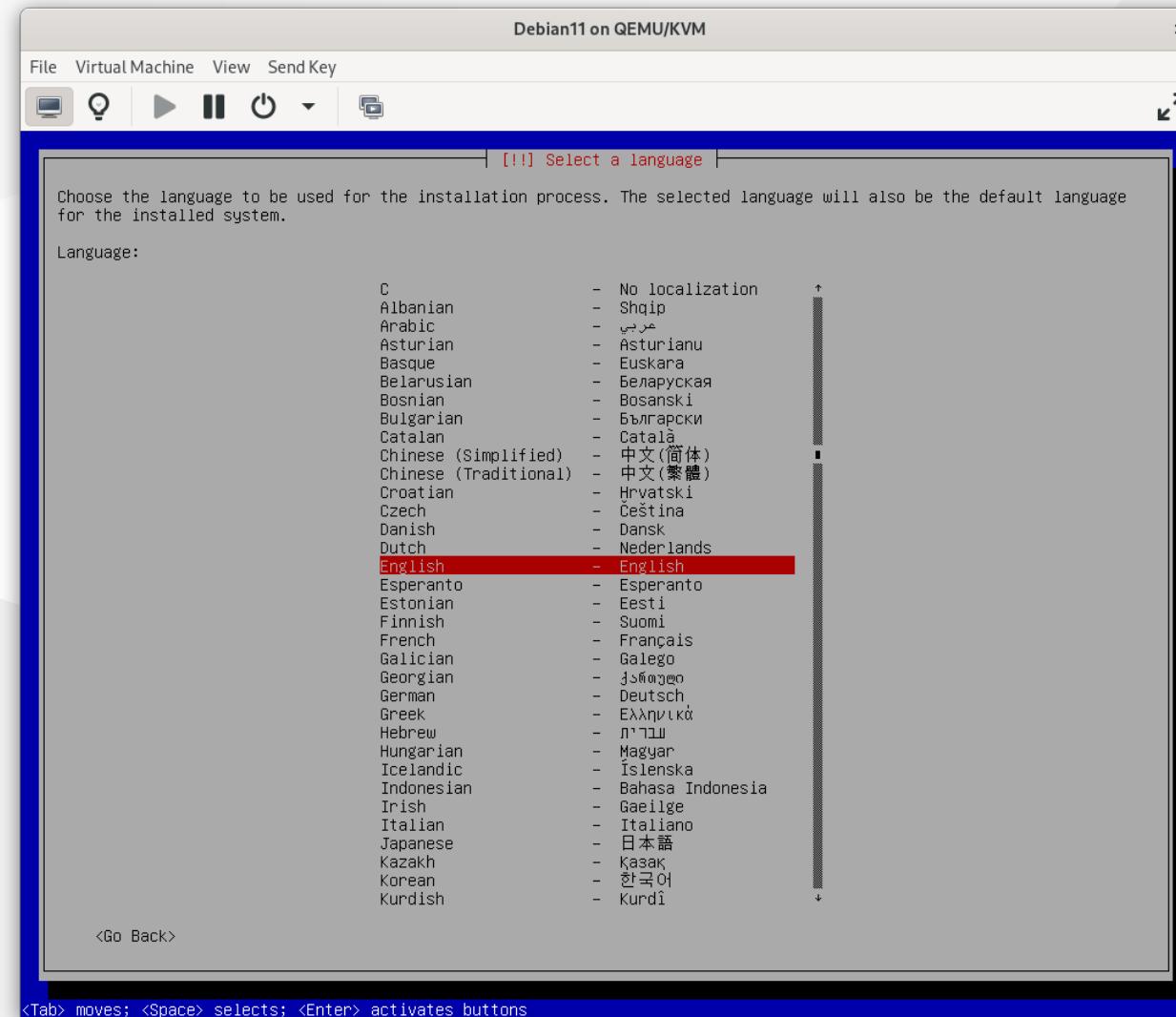


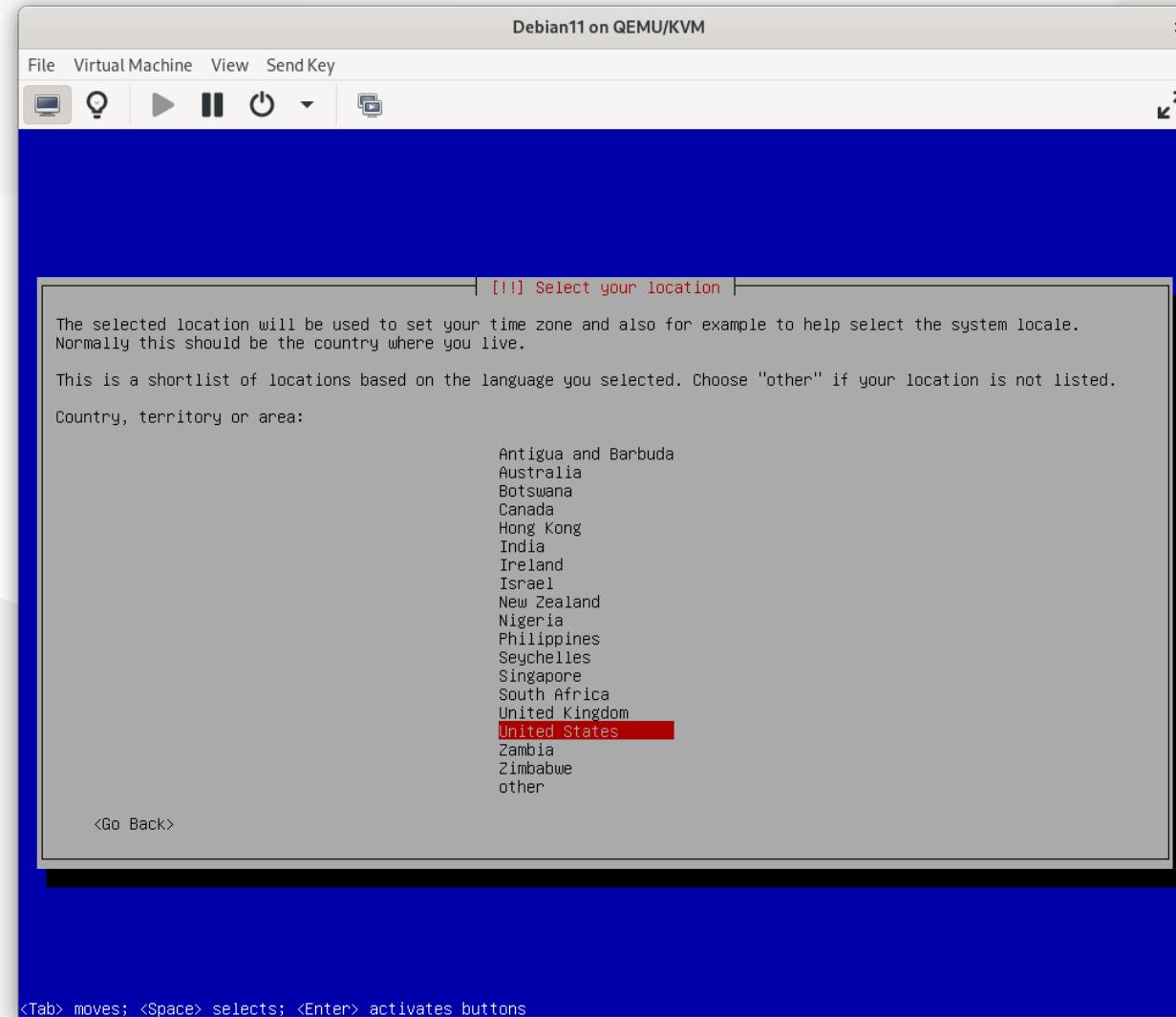
The virtual machine is now being created. Allocation of disk storage and retrieval of the installation images may take a few minutes to complete.

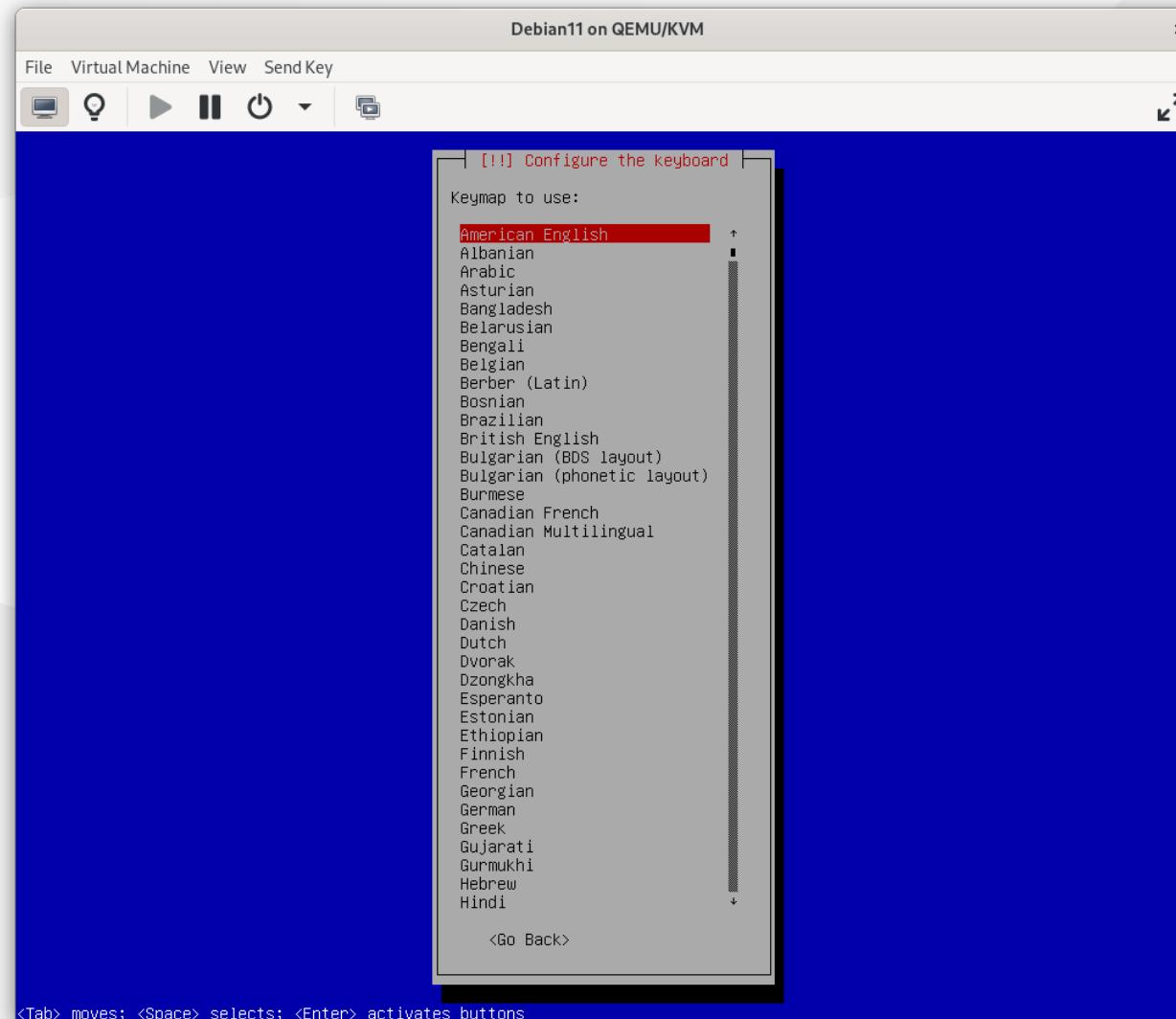
Allocating 'Debian11.qcow2'

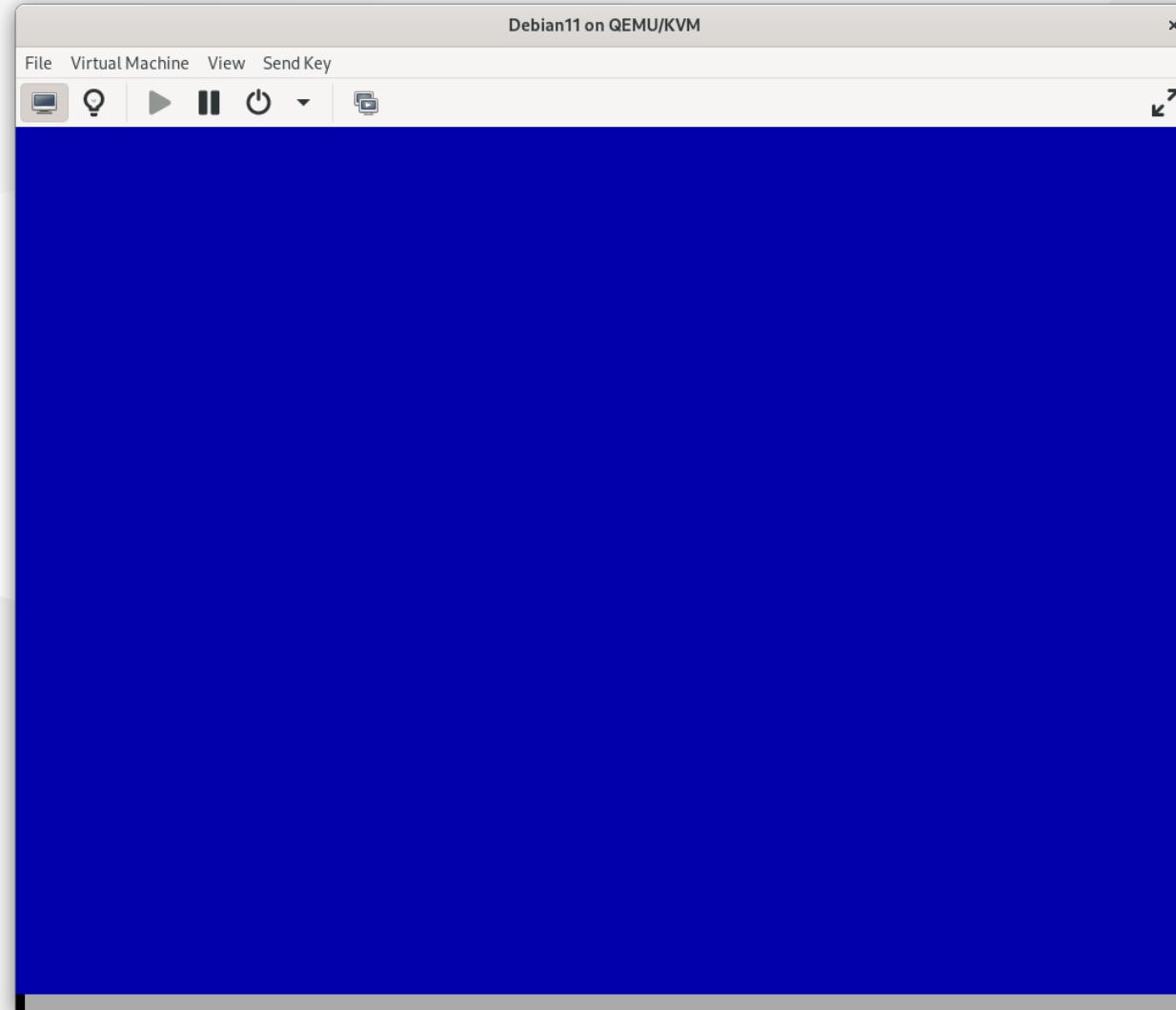
0% 0 B --- ETA

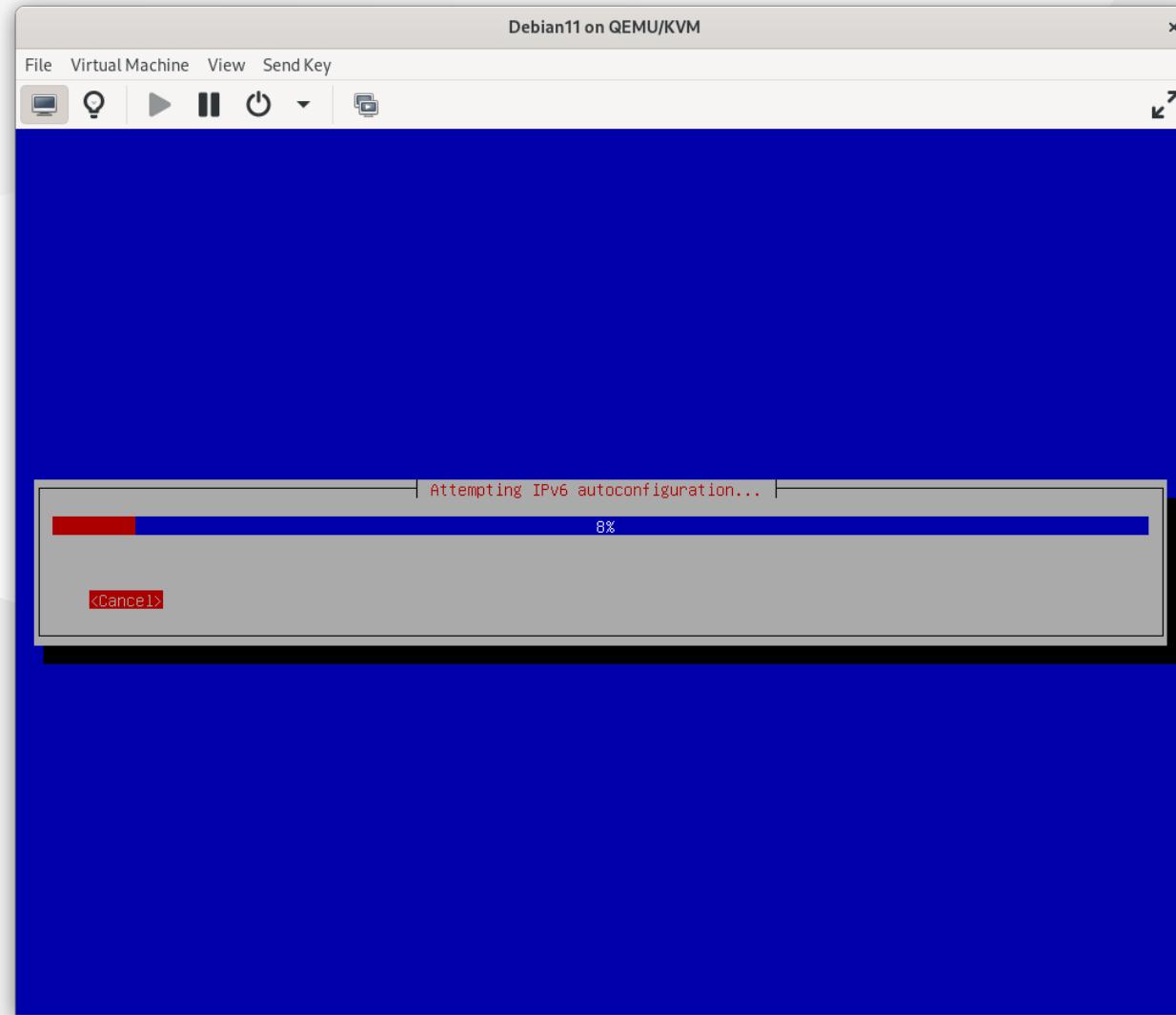


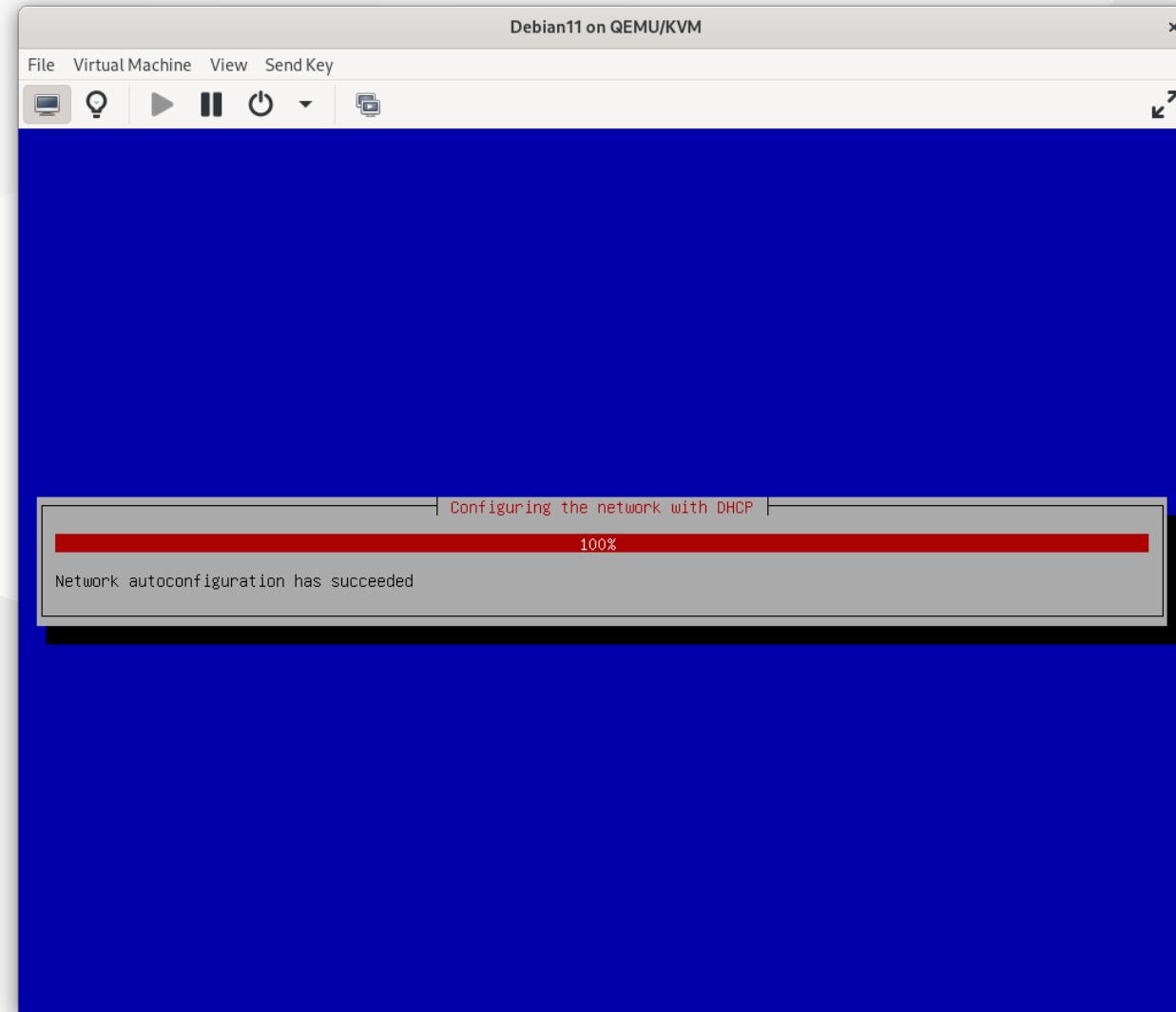


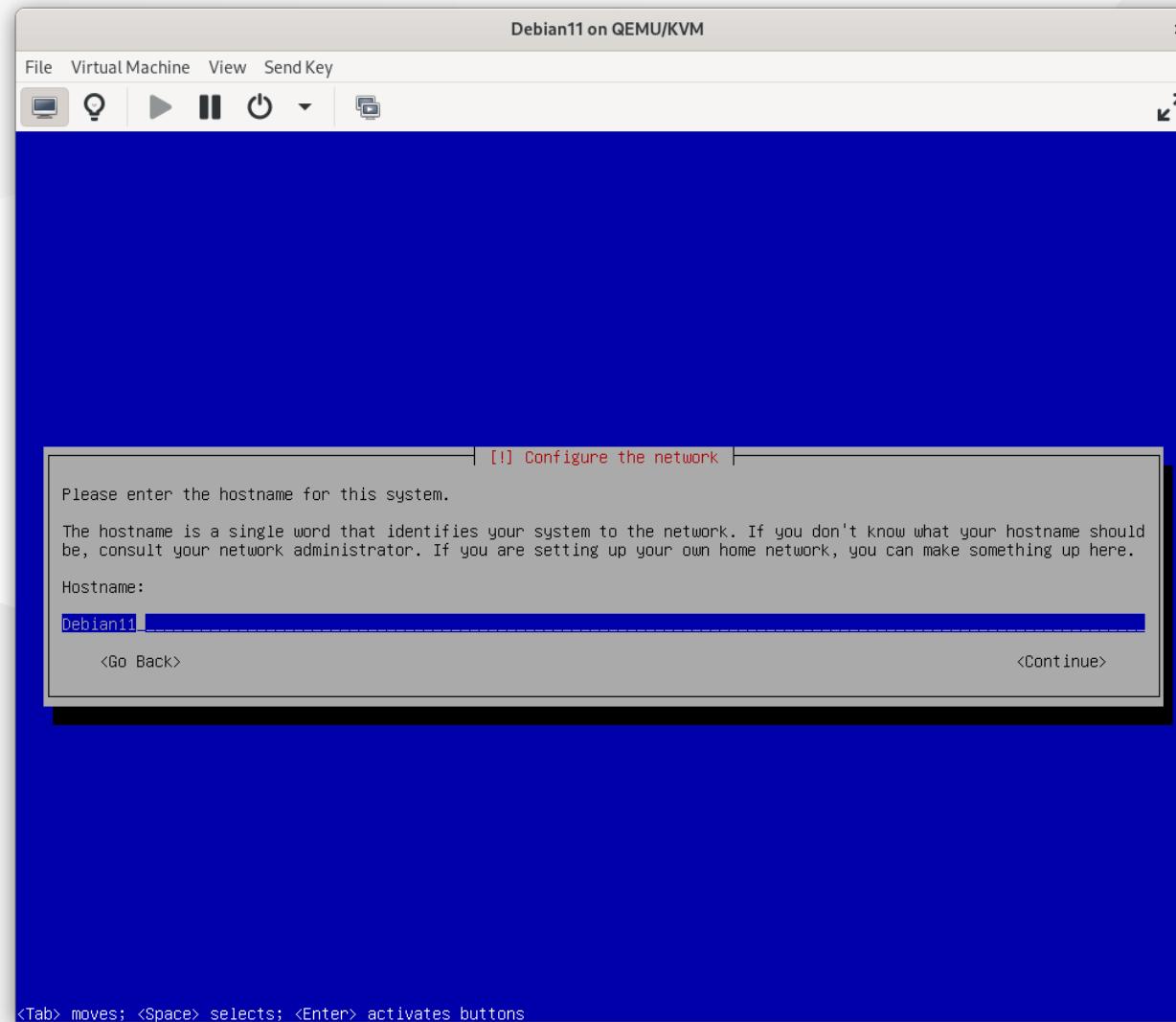


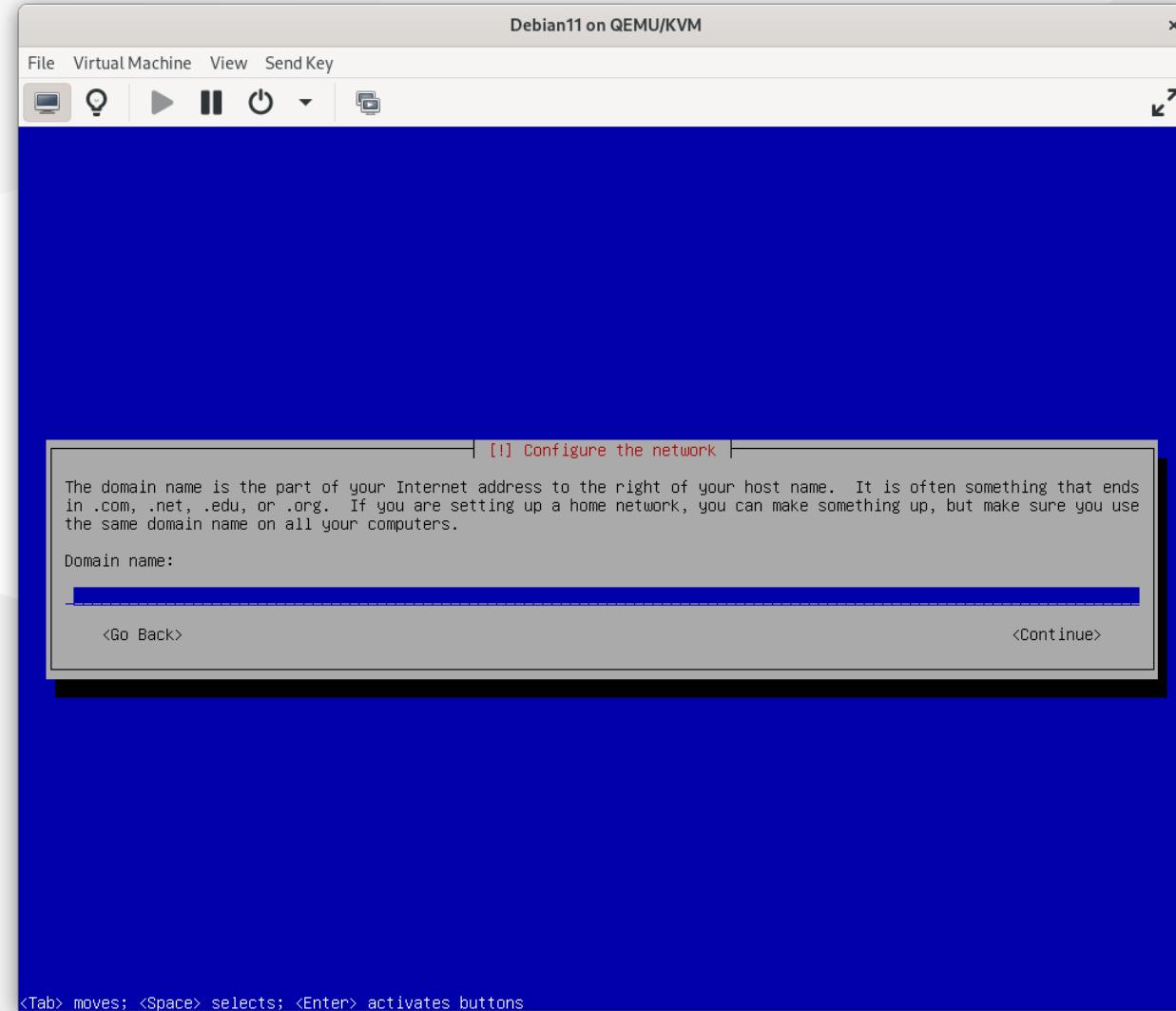


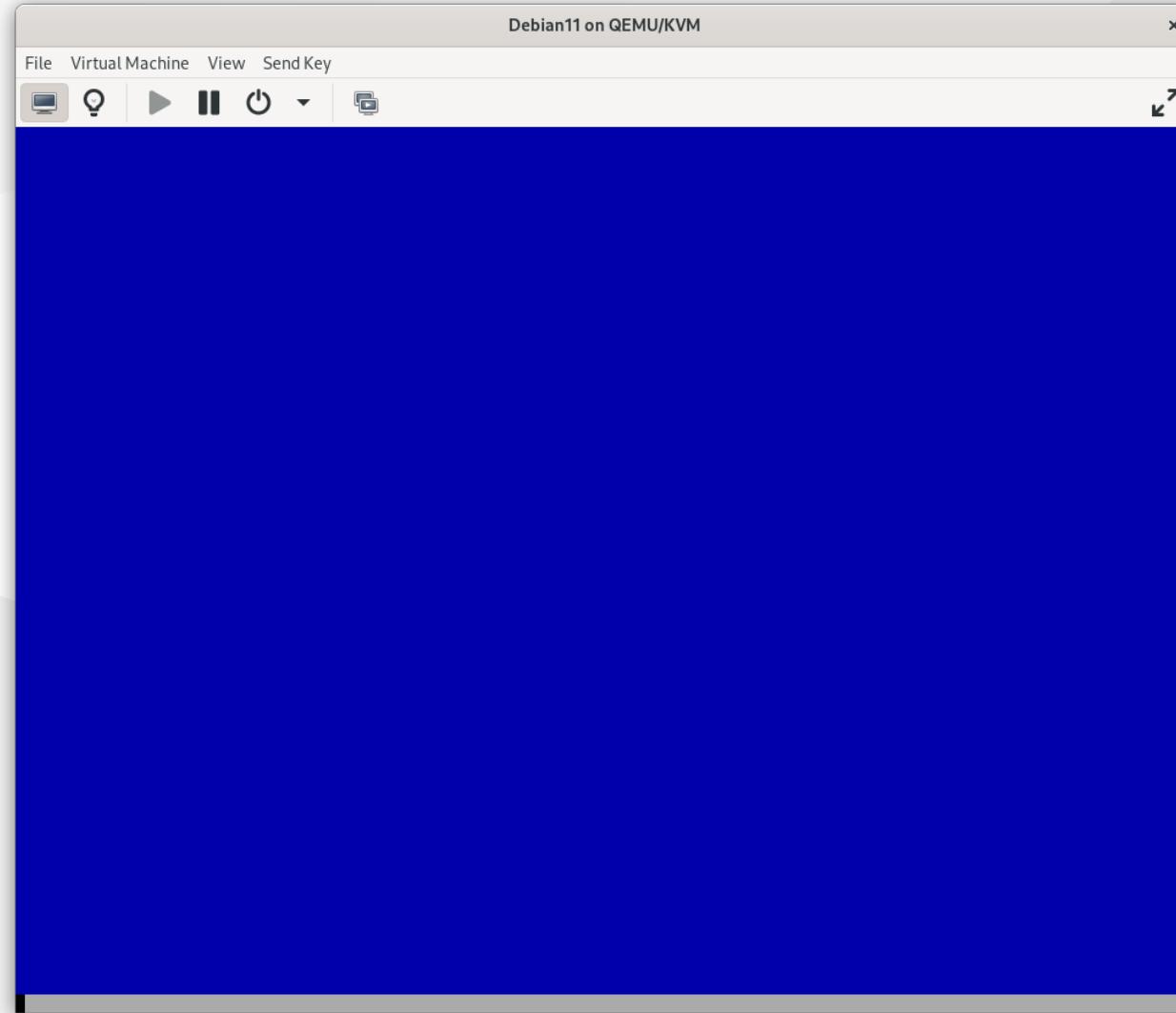


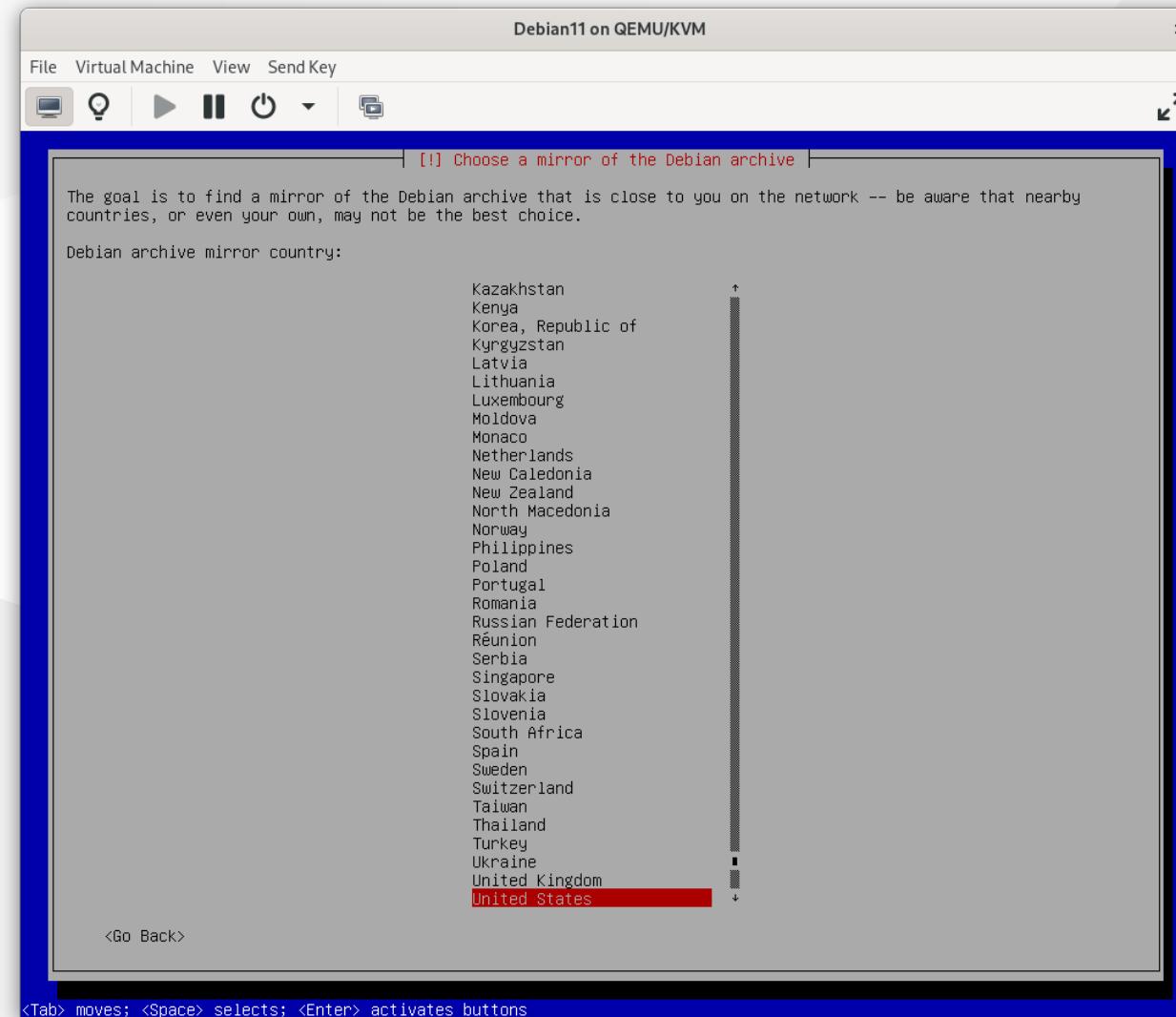


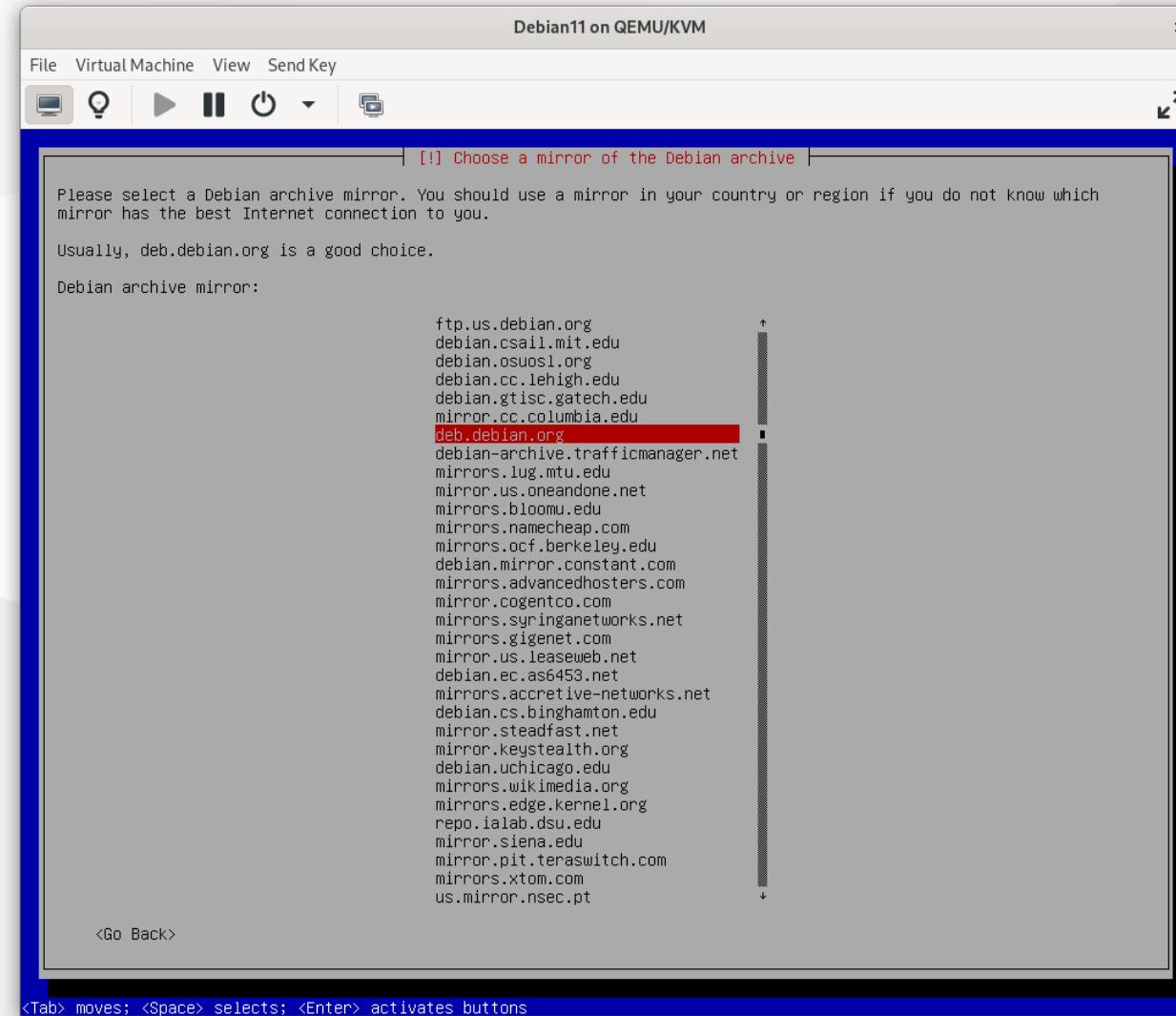


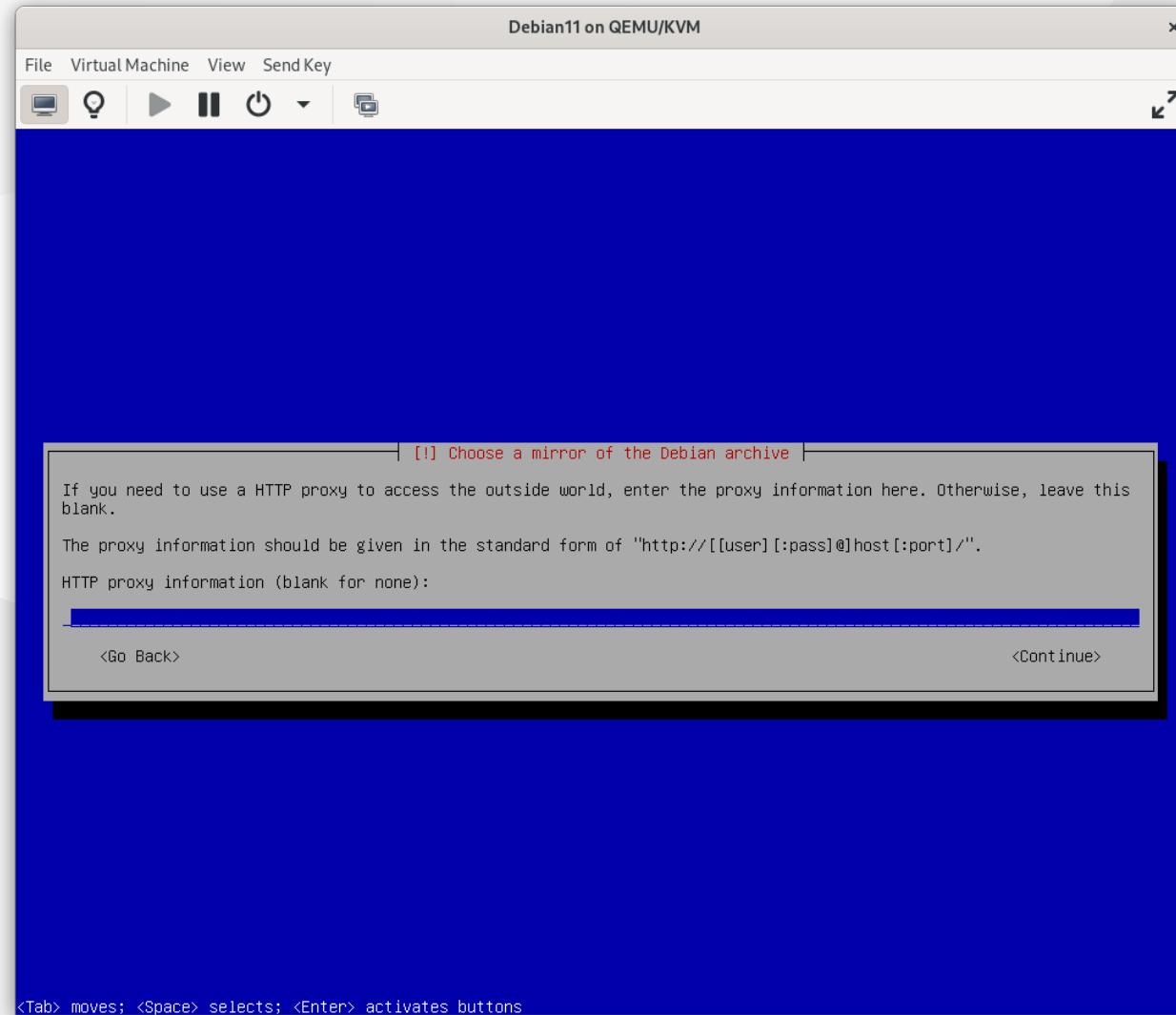


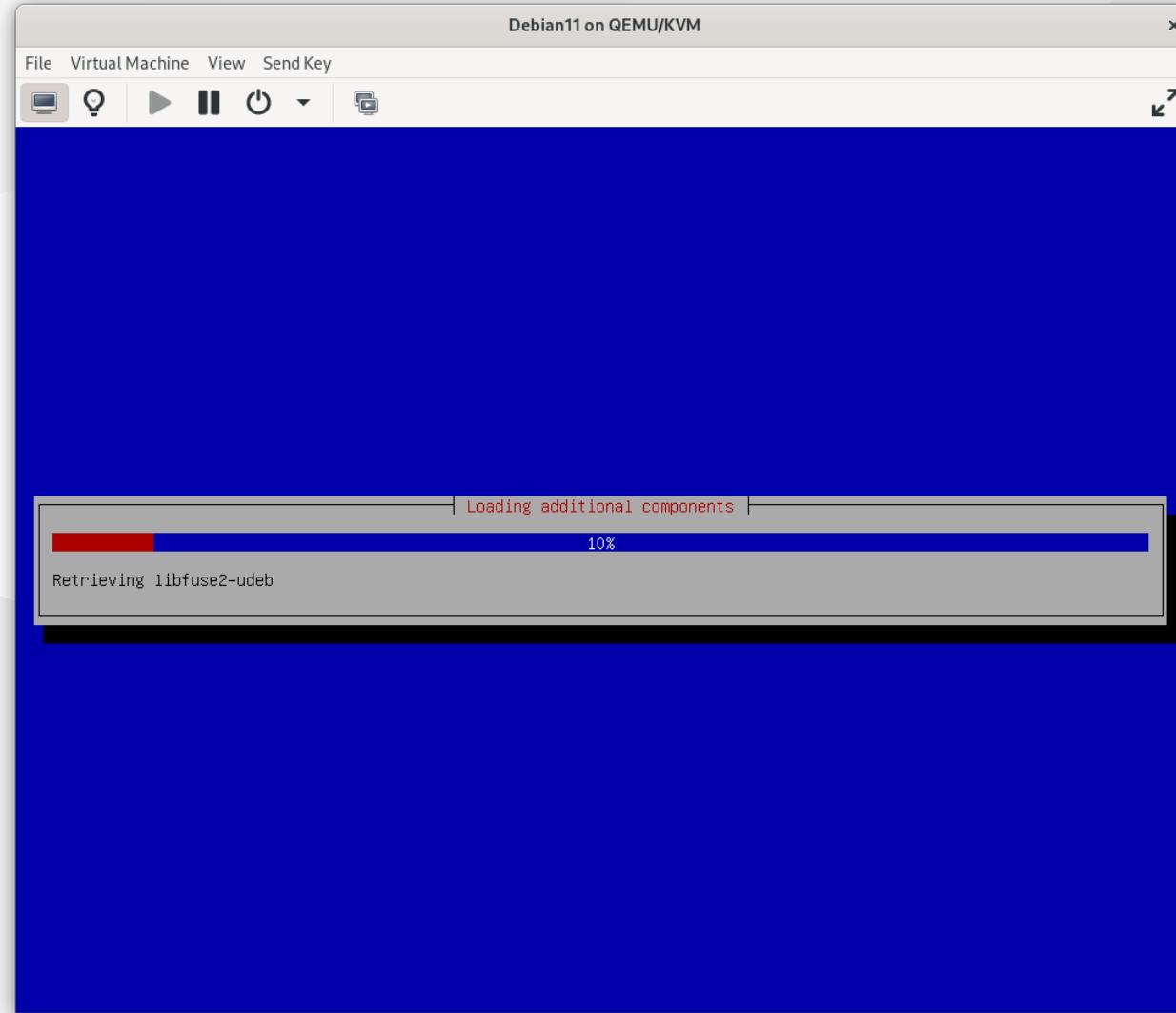


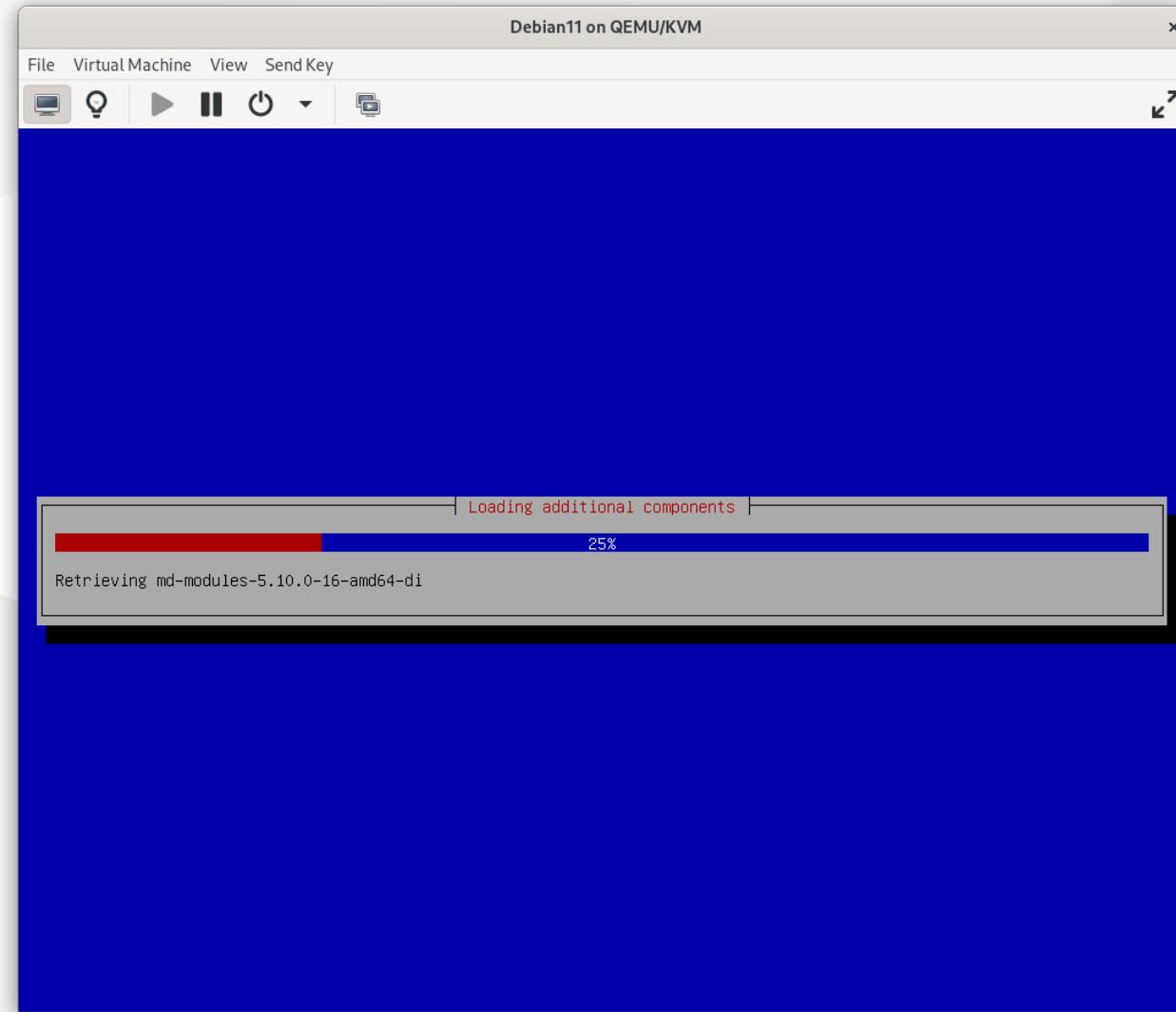


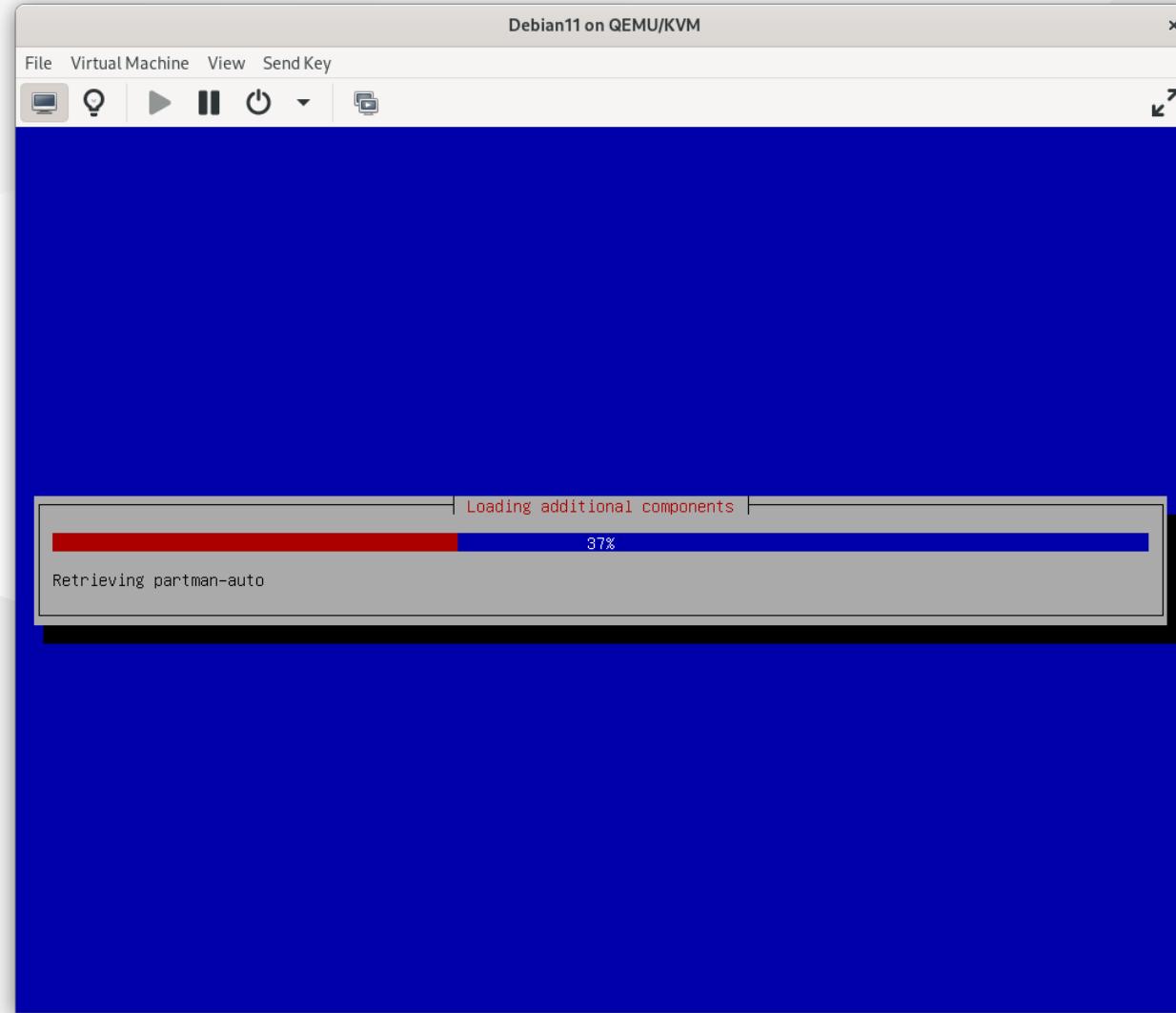


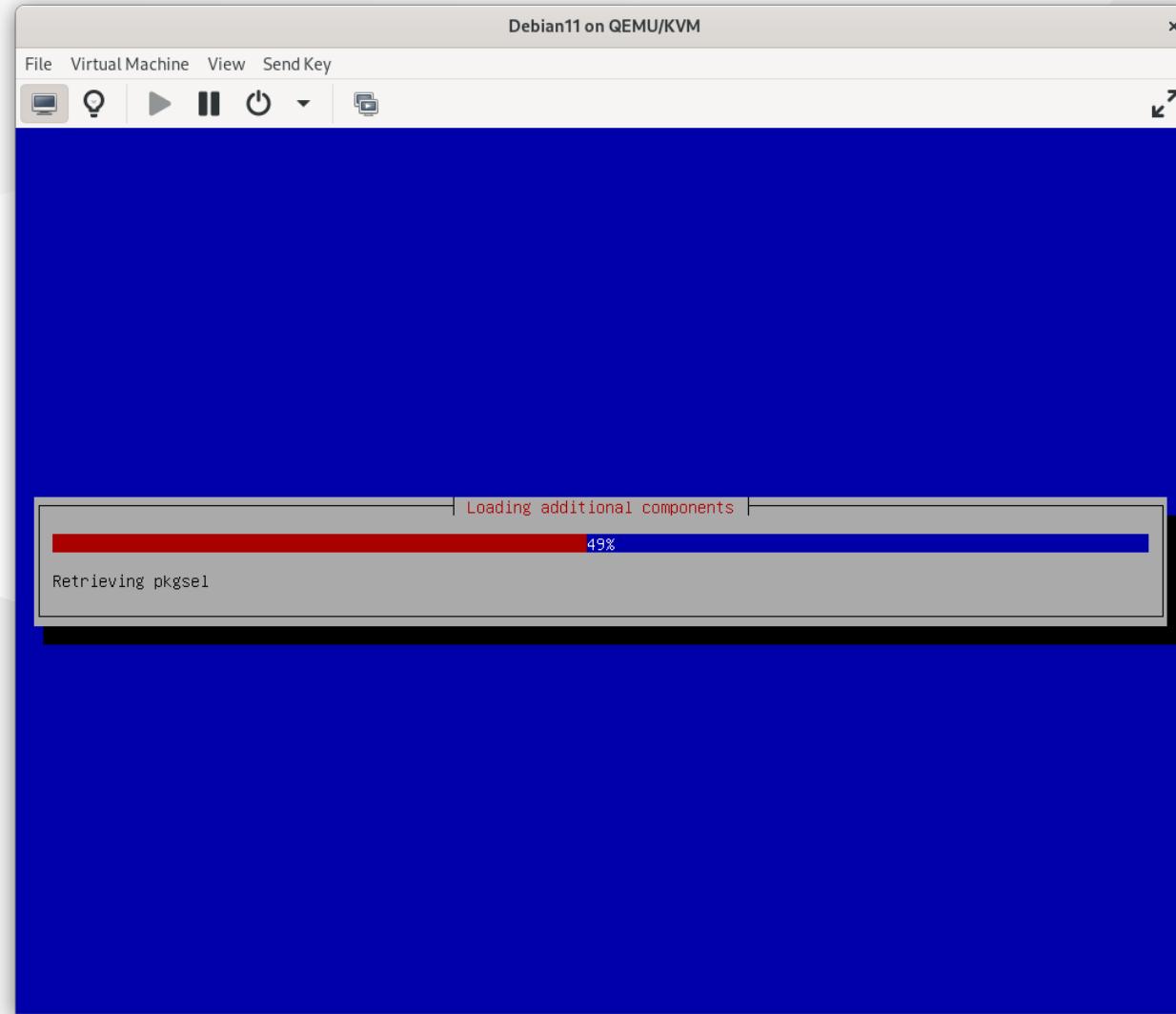


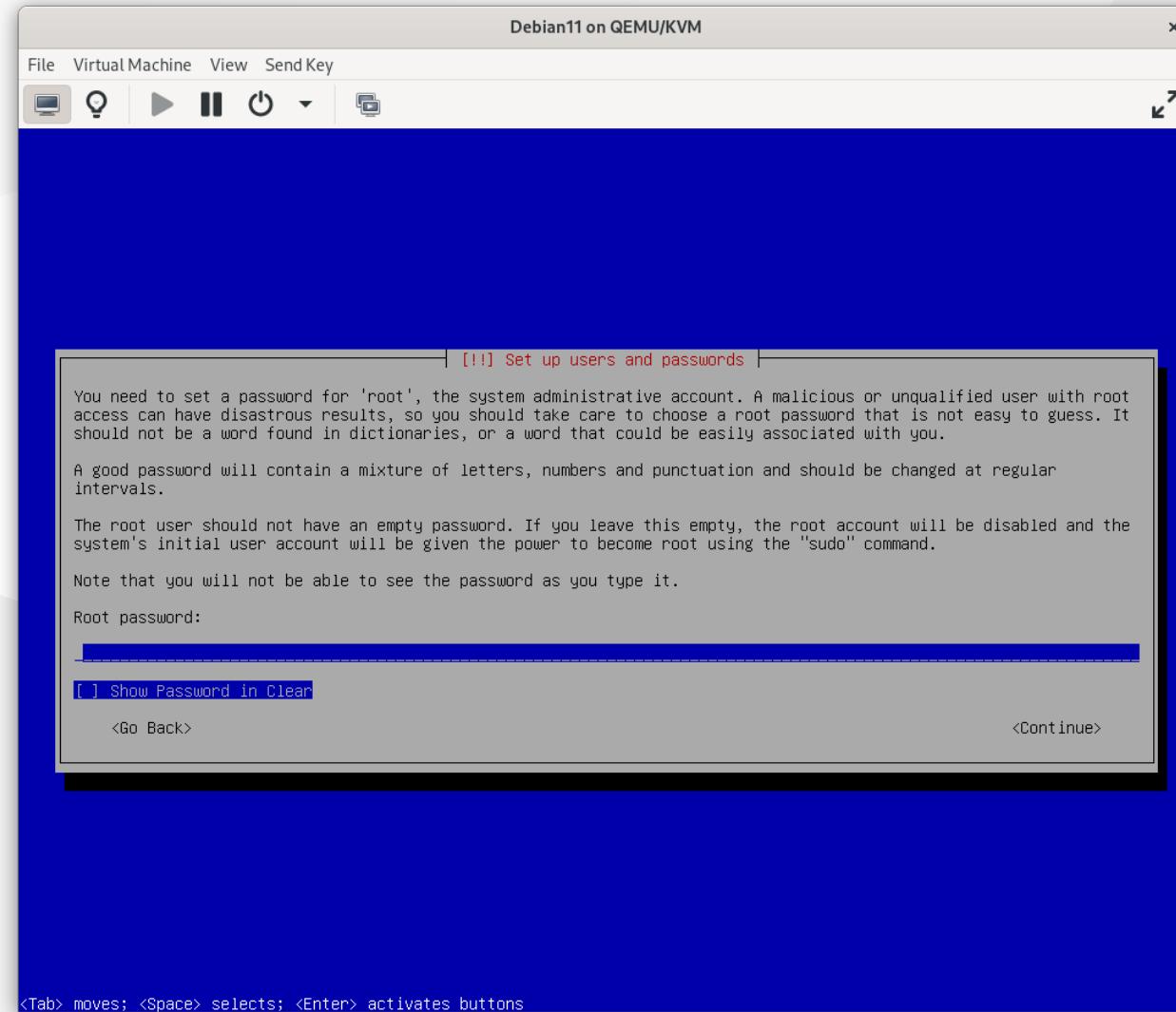


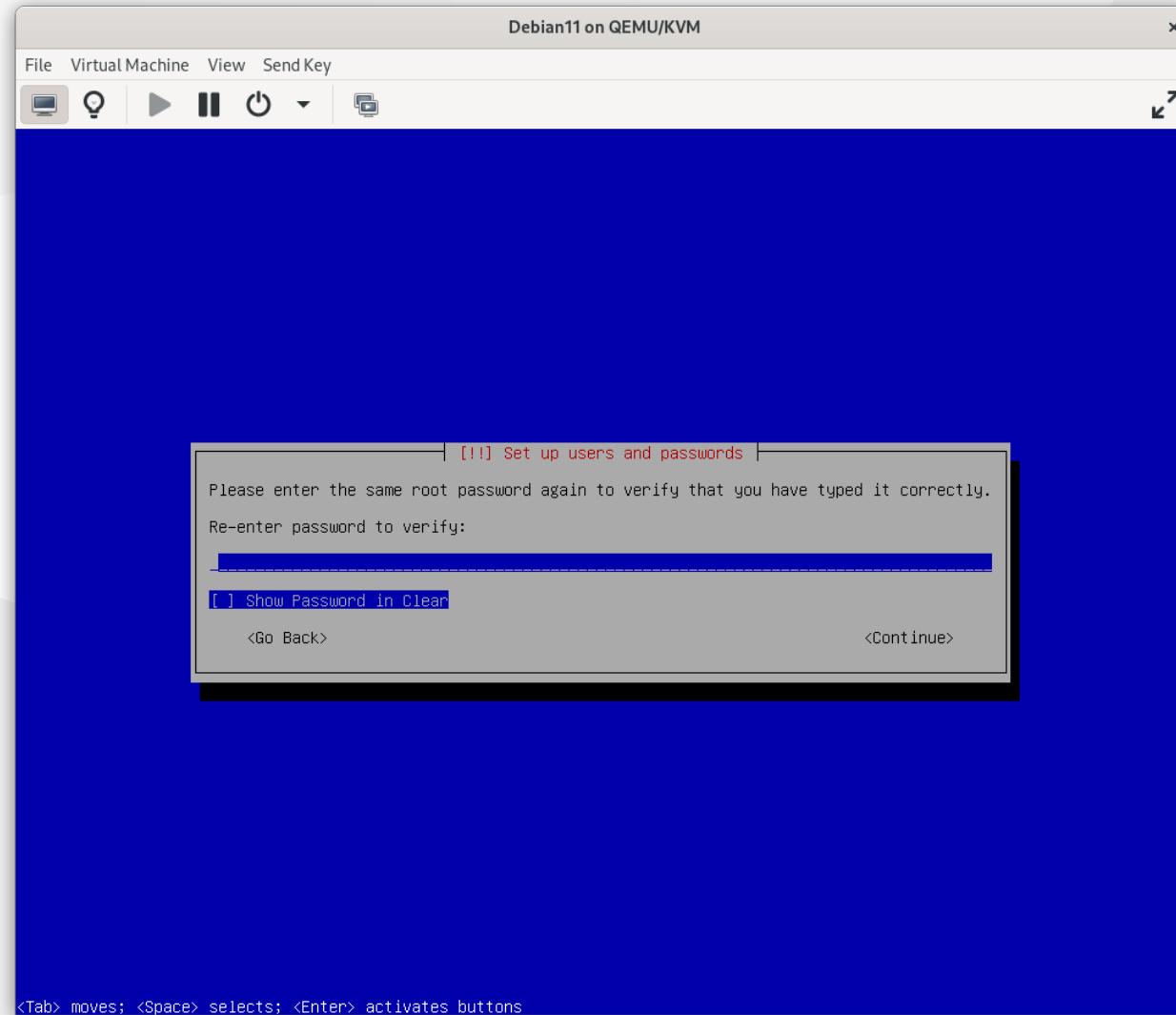


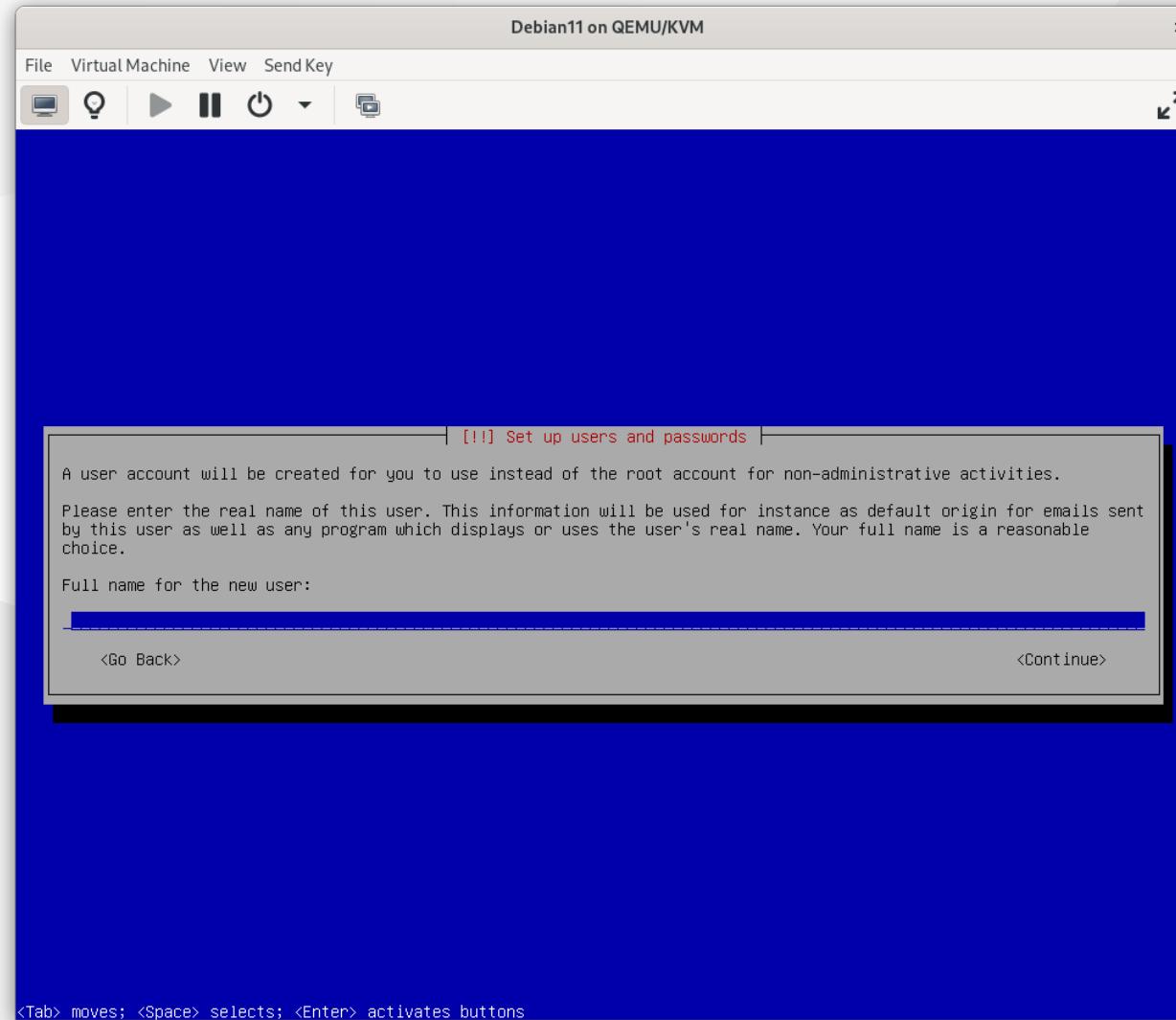


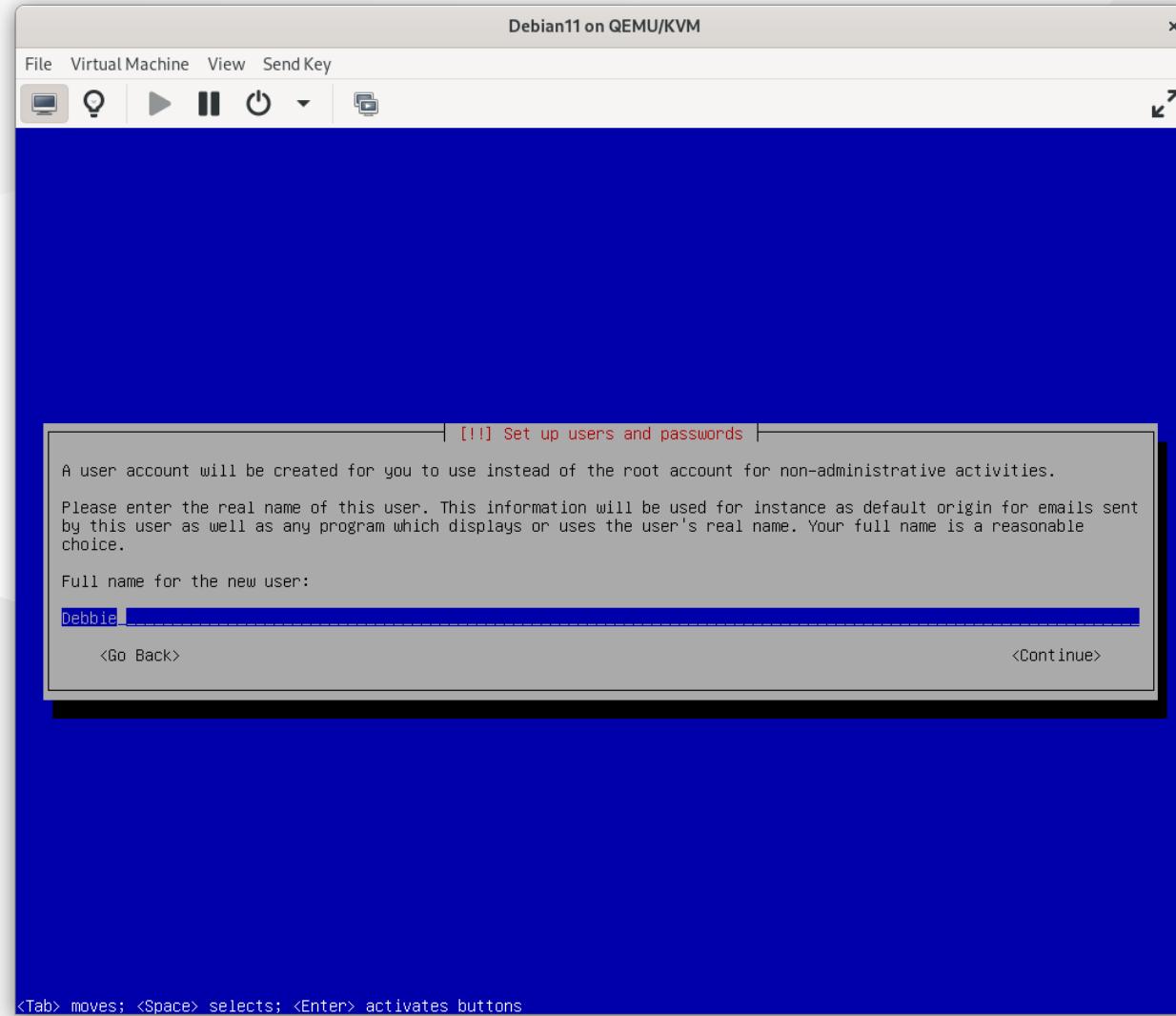


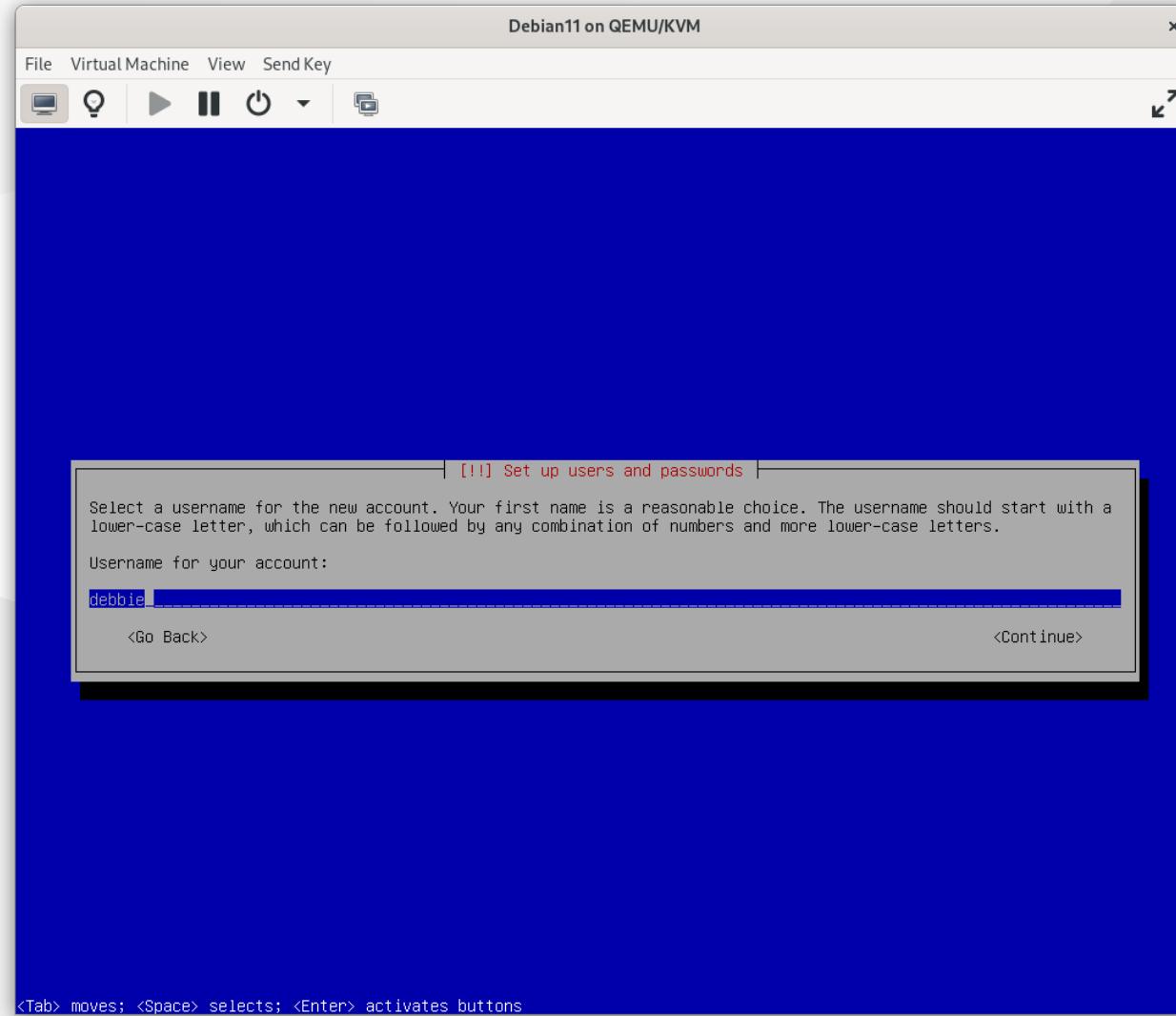


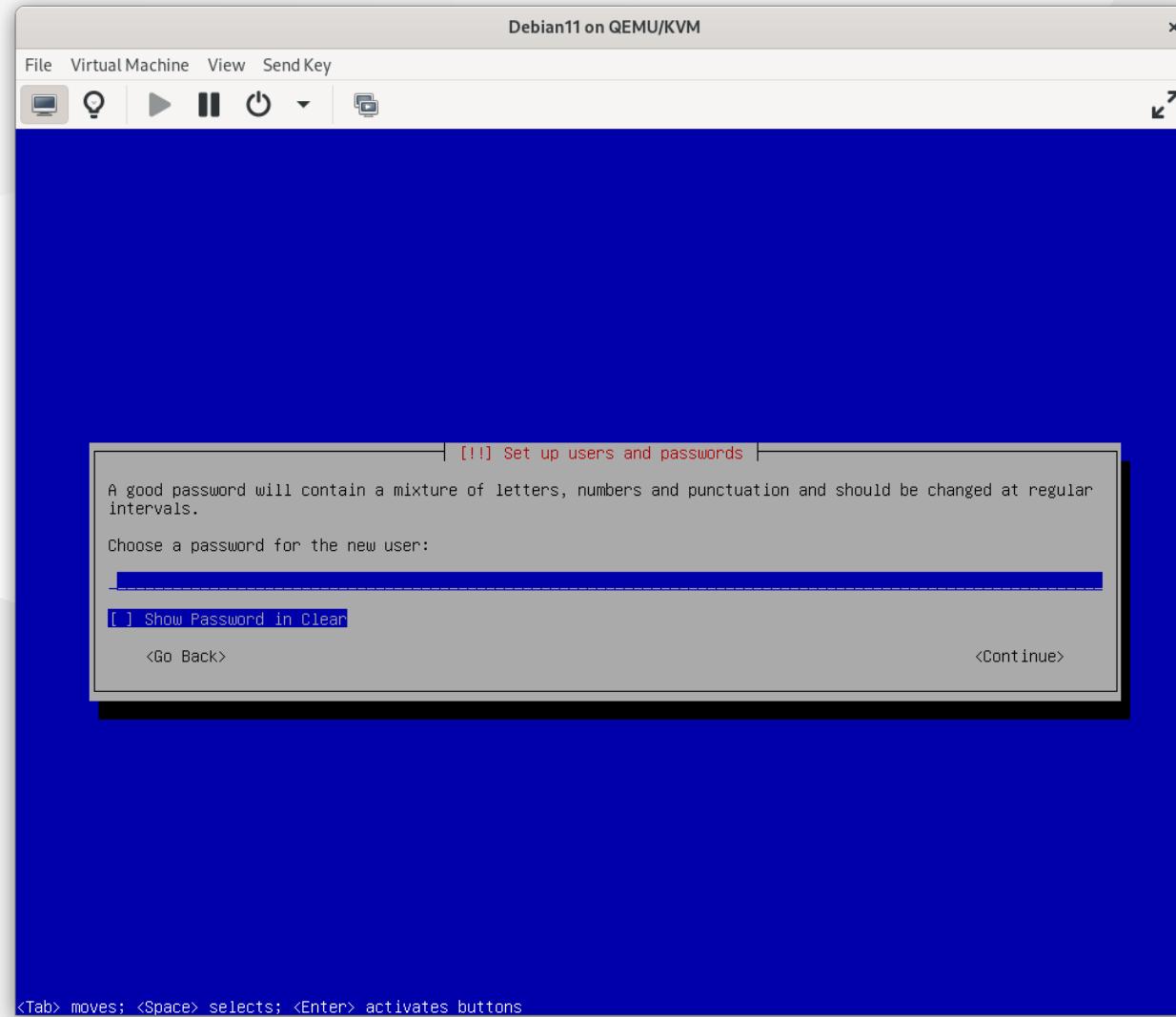


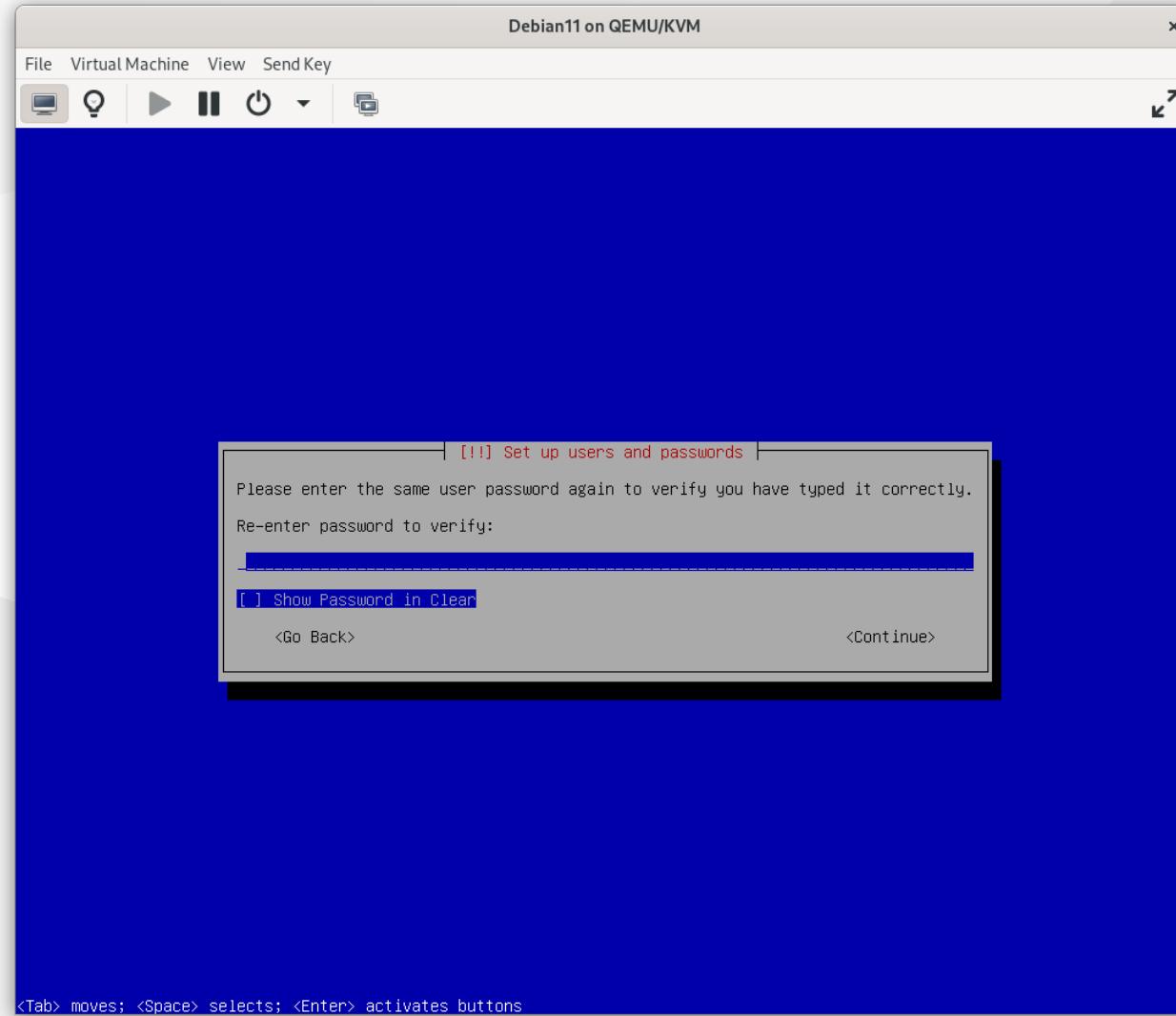


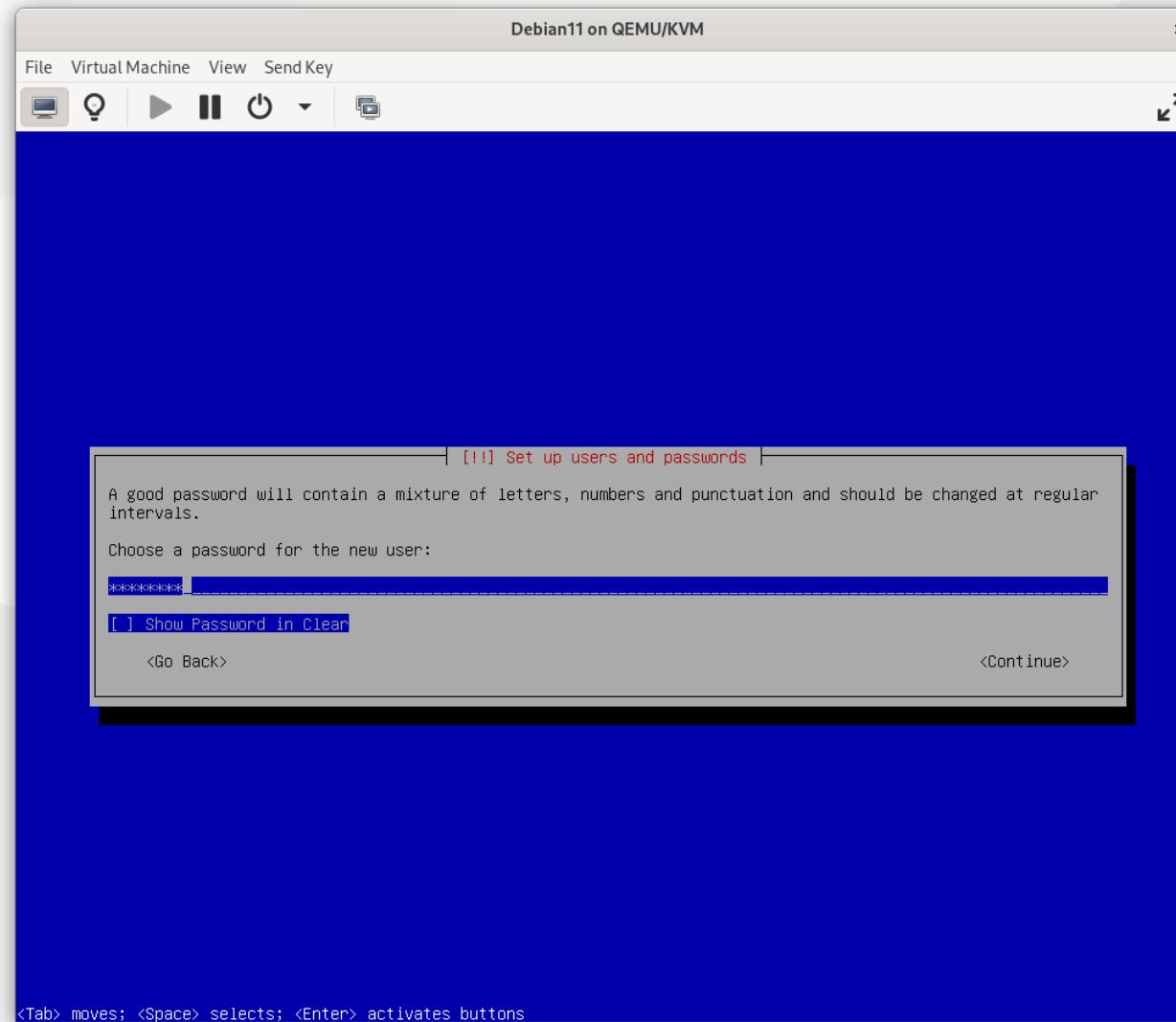


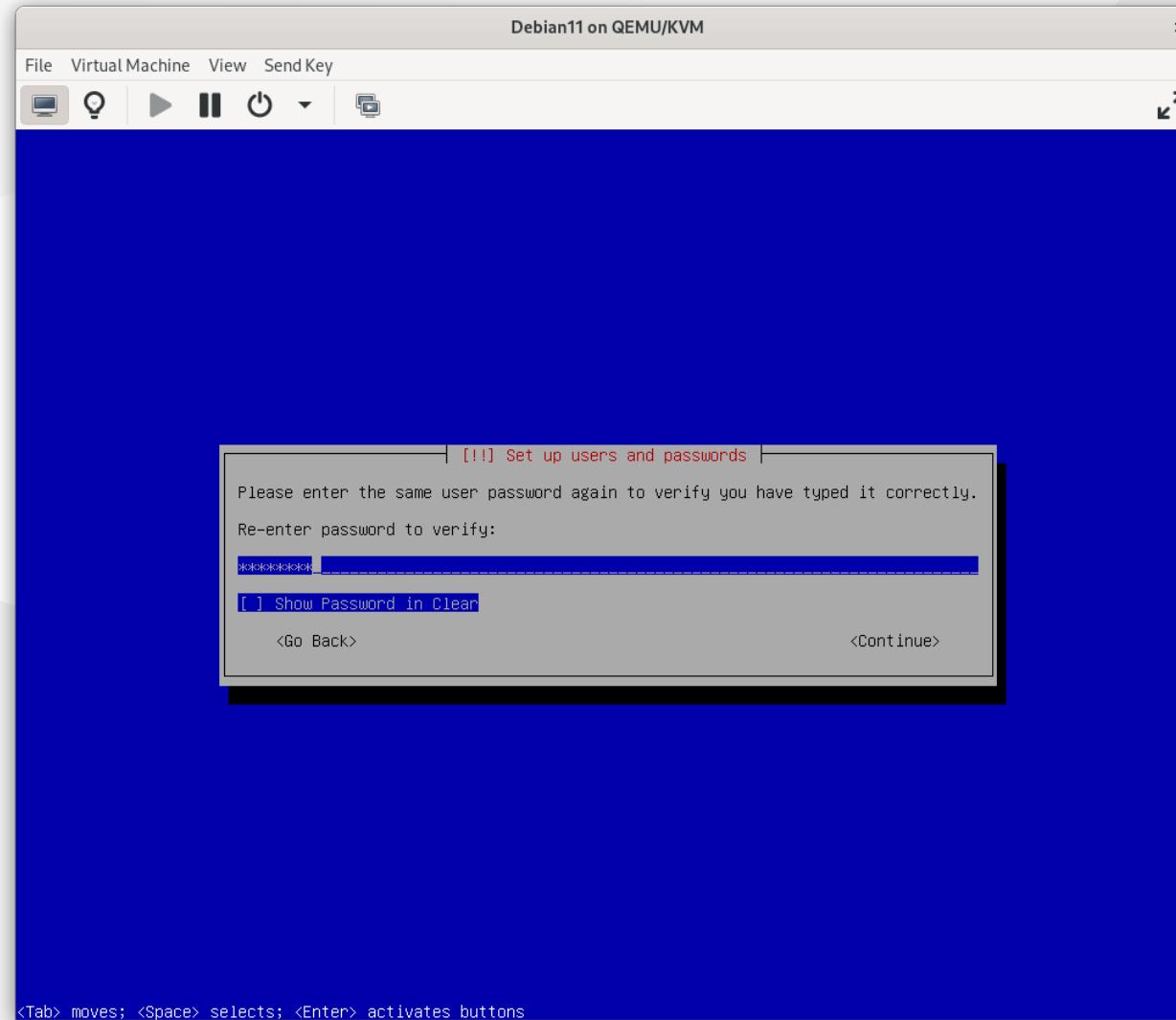


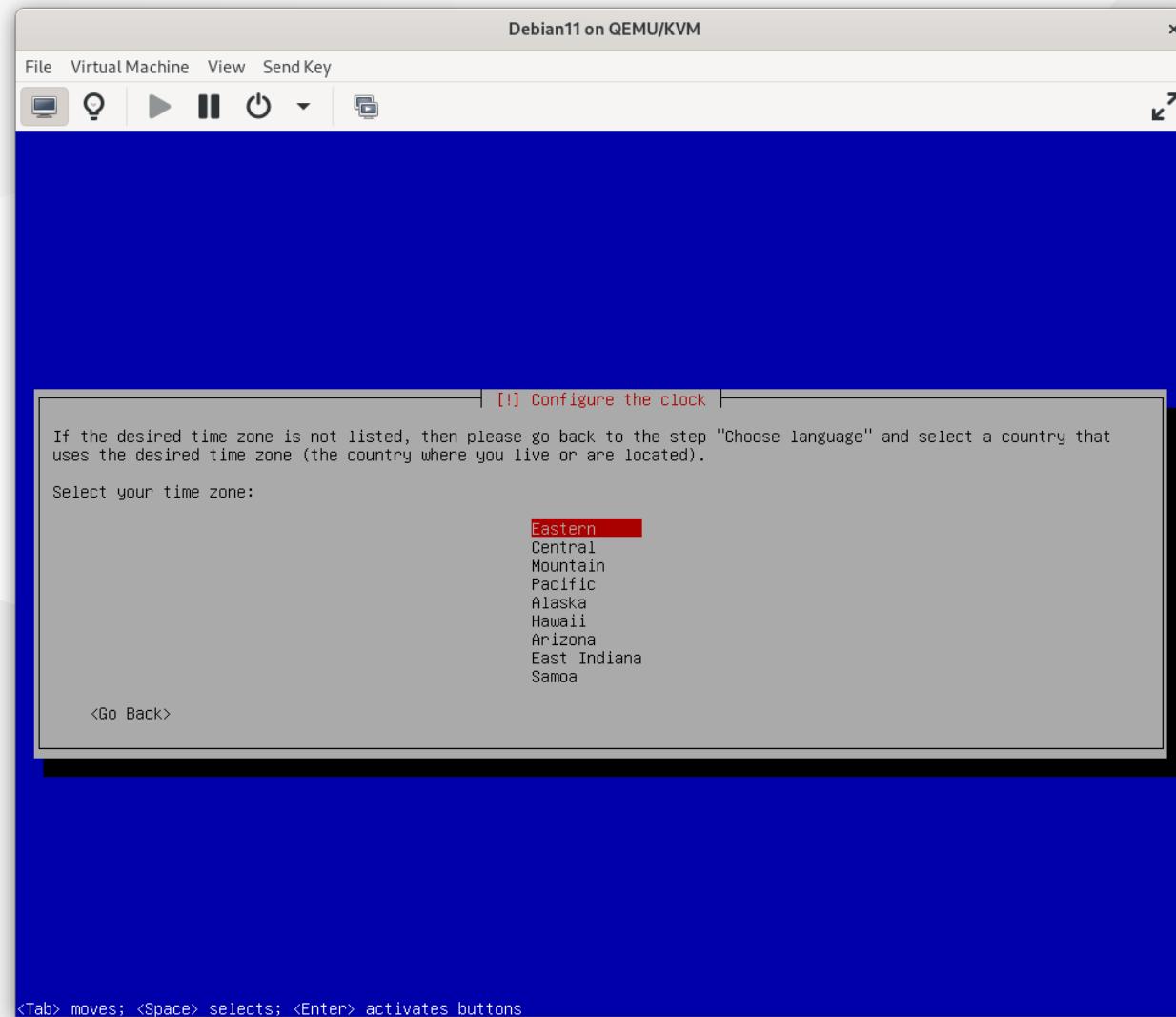


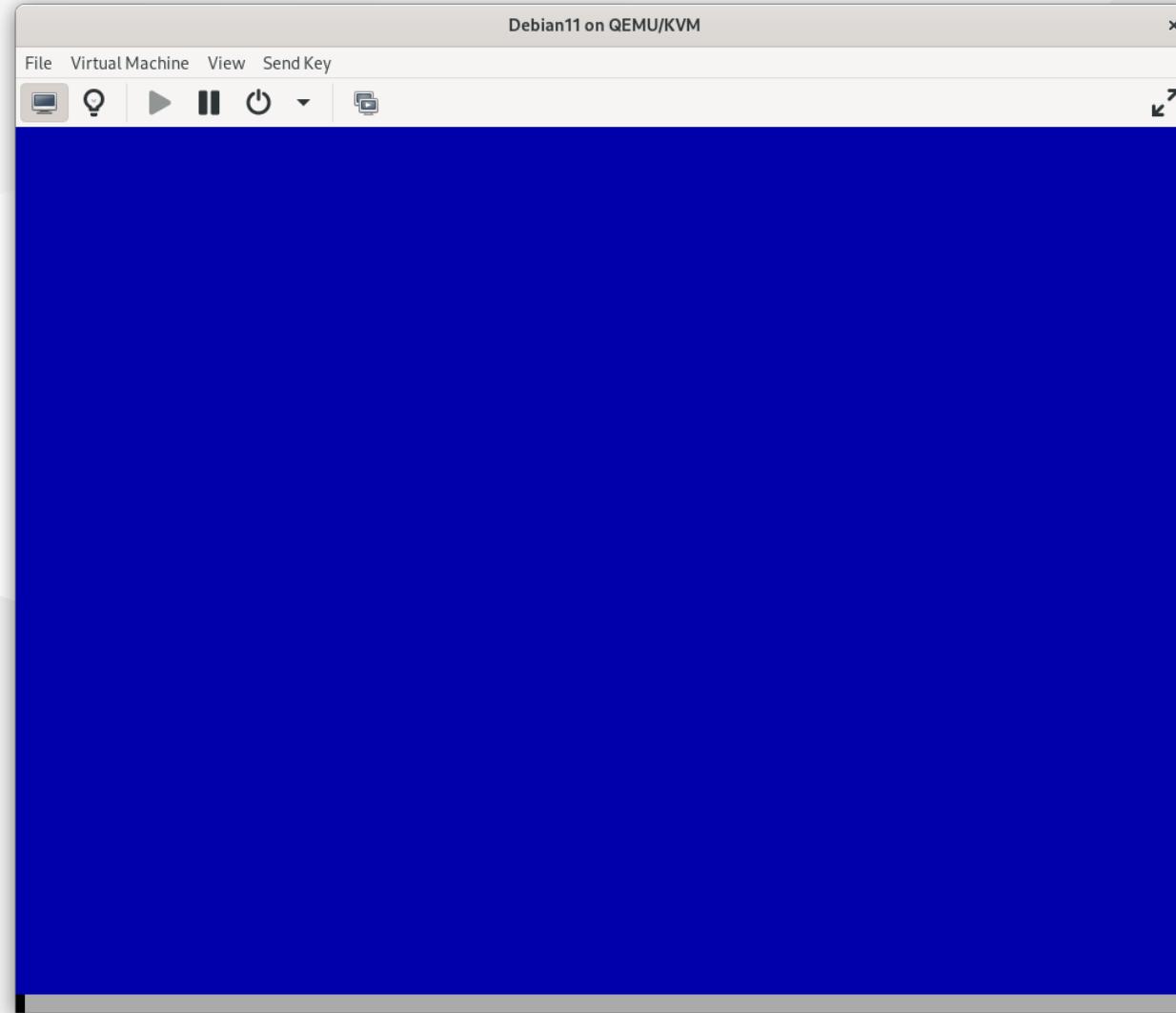


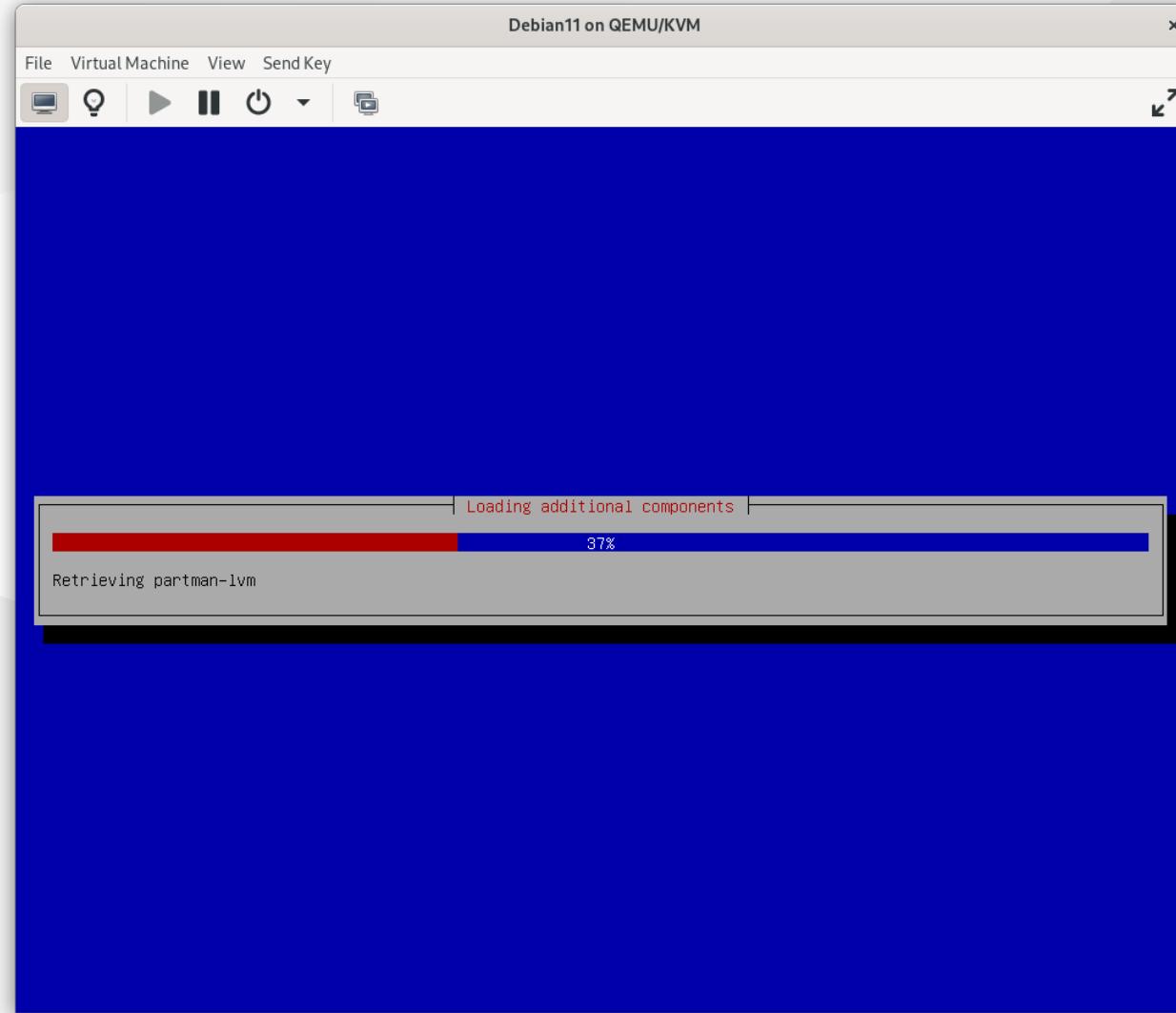


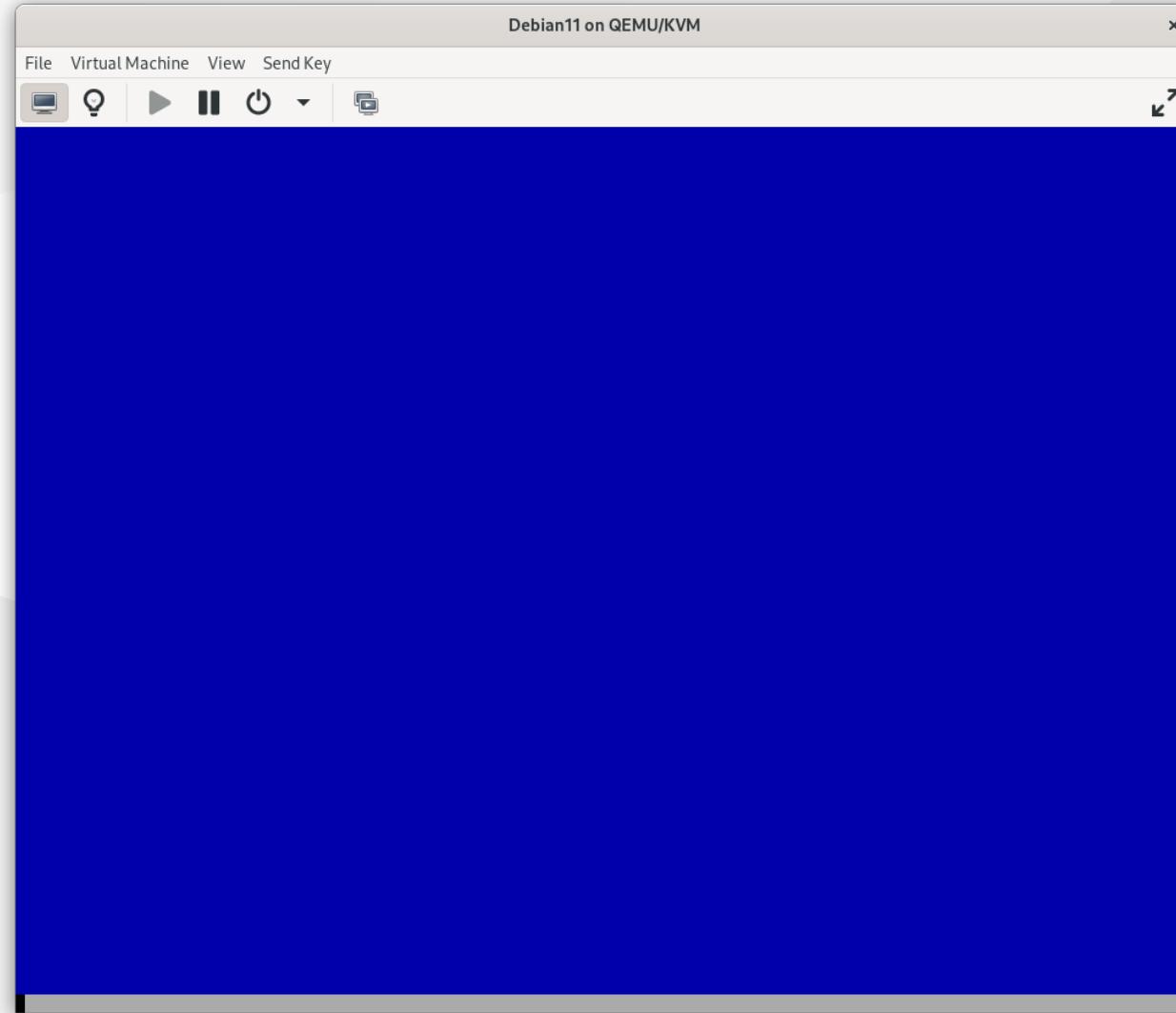


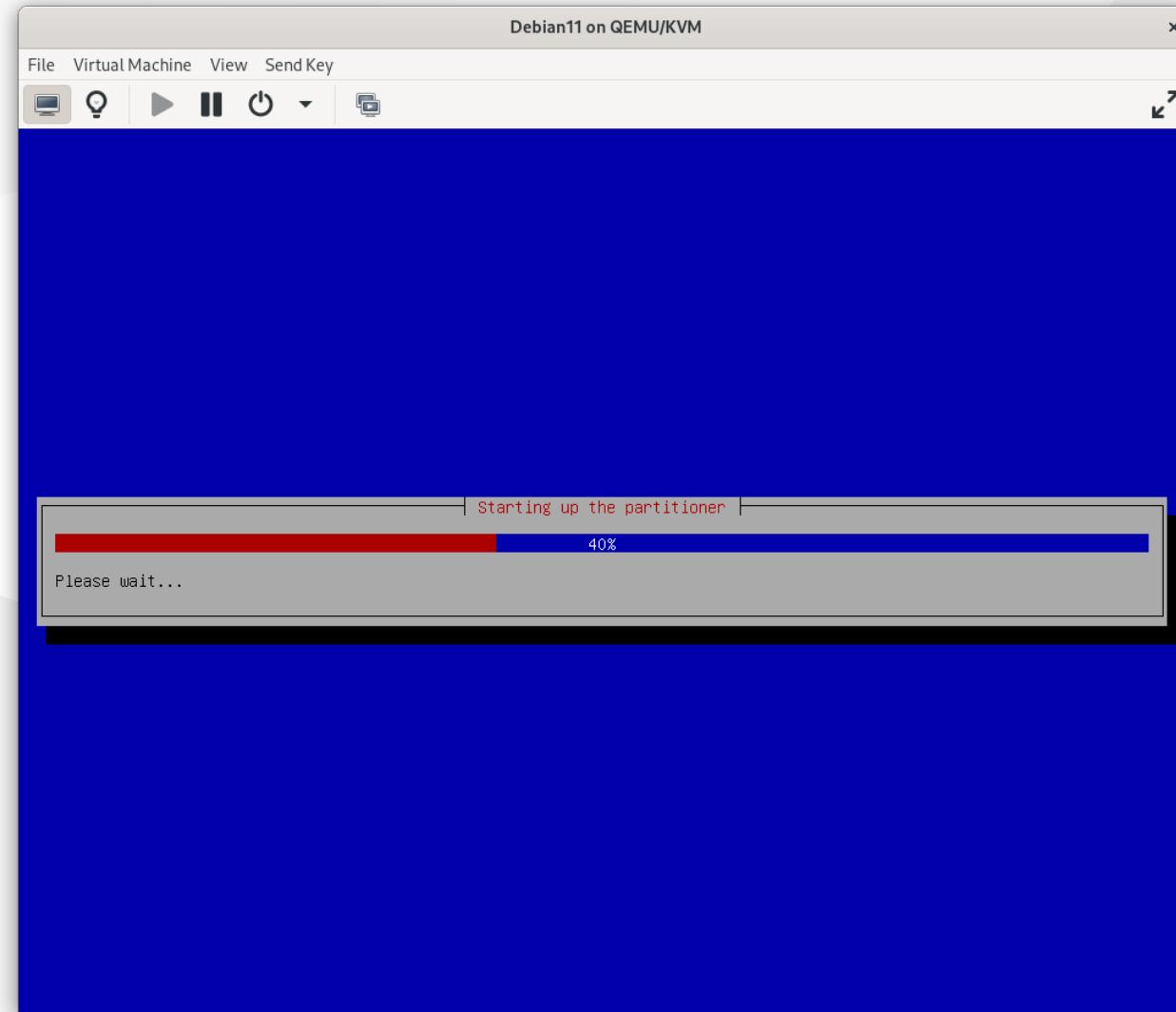


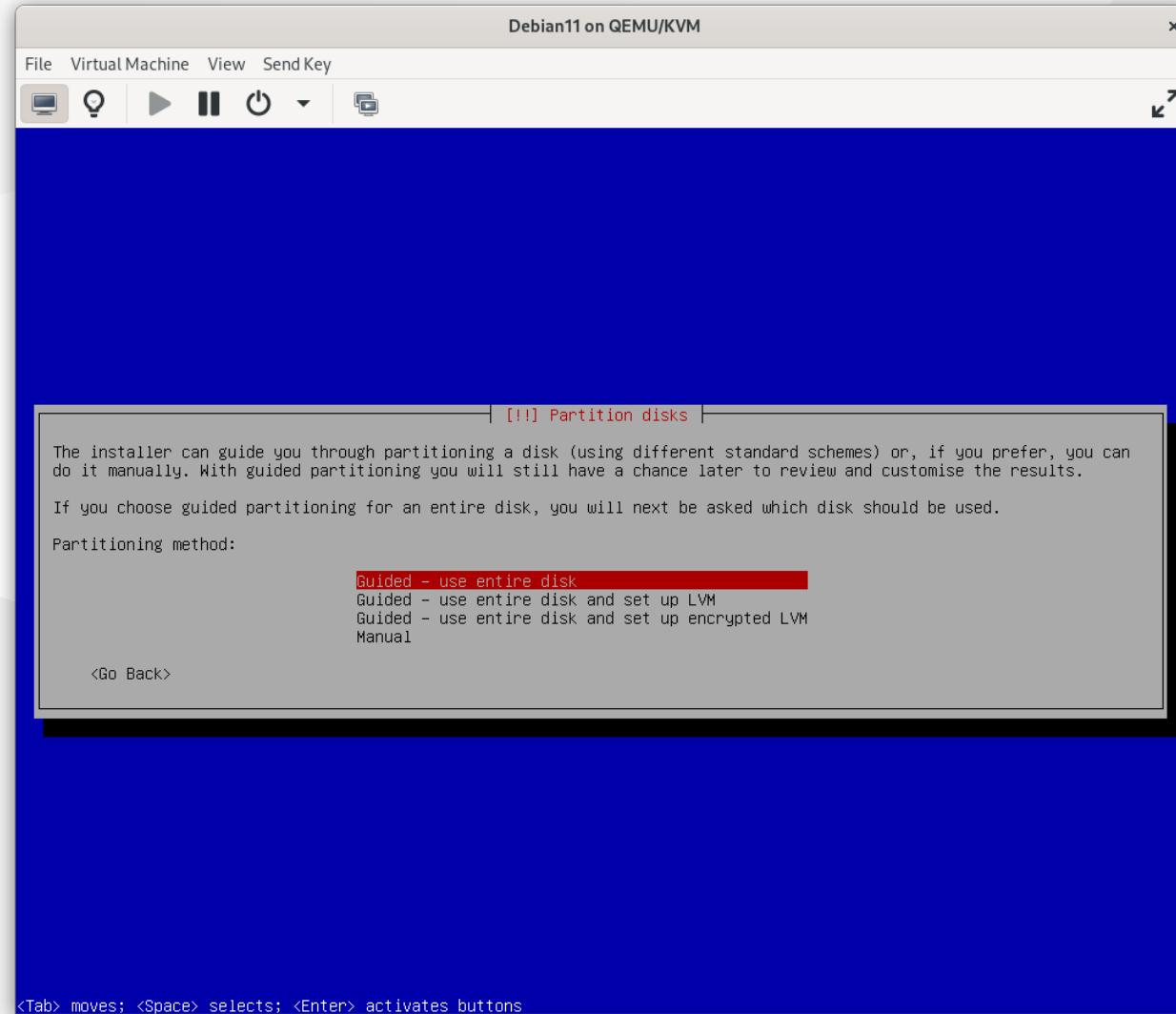


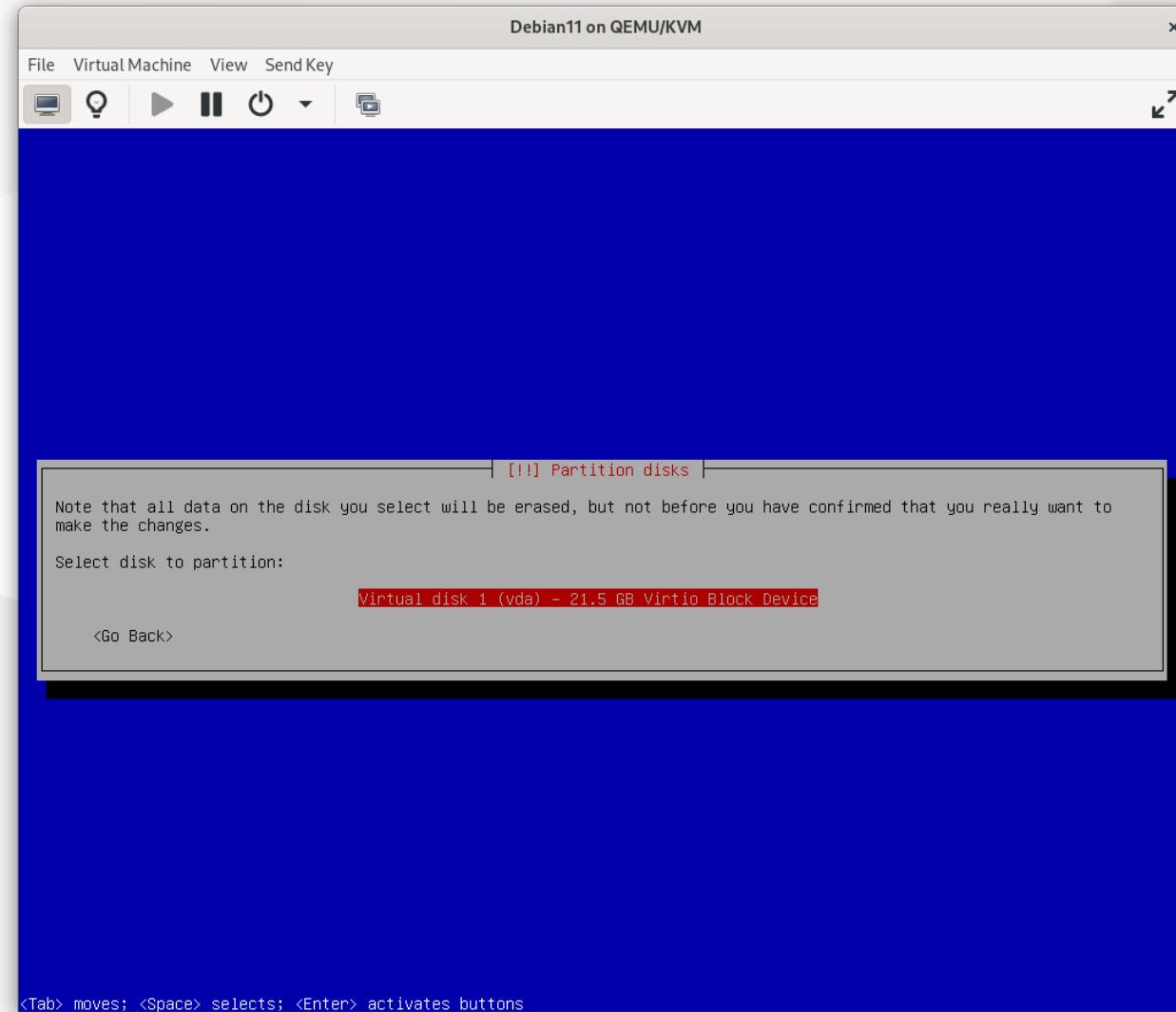


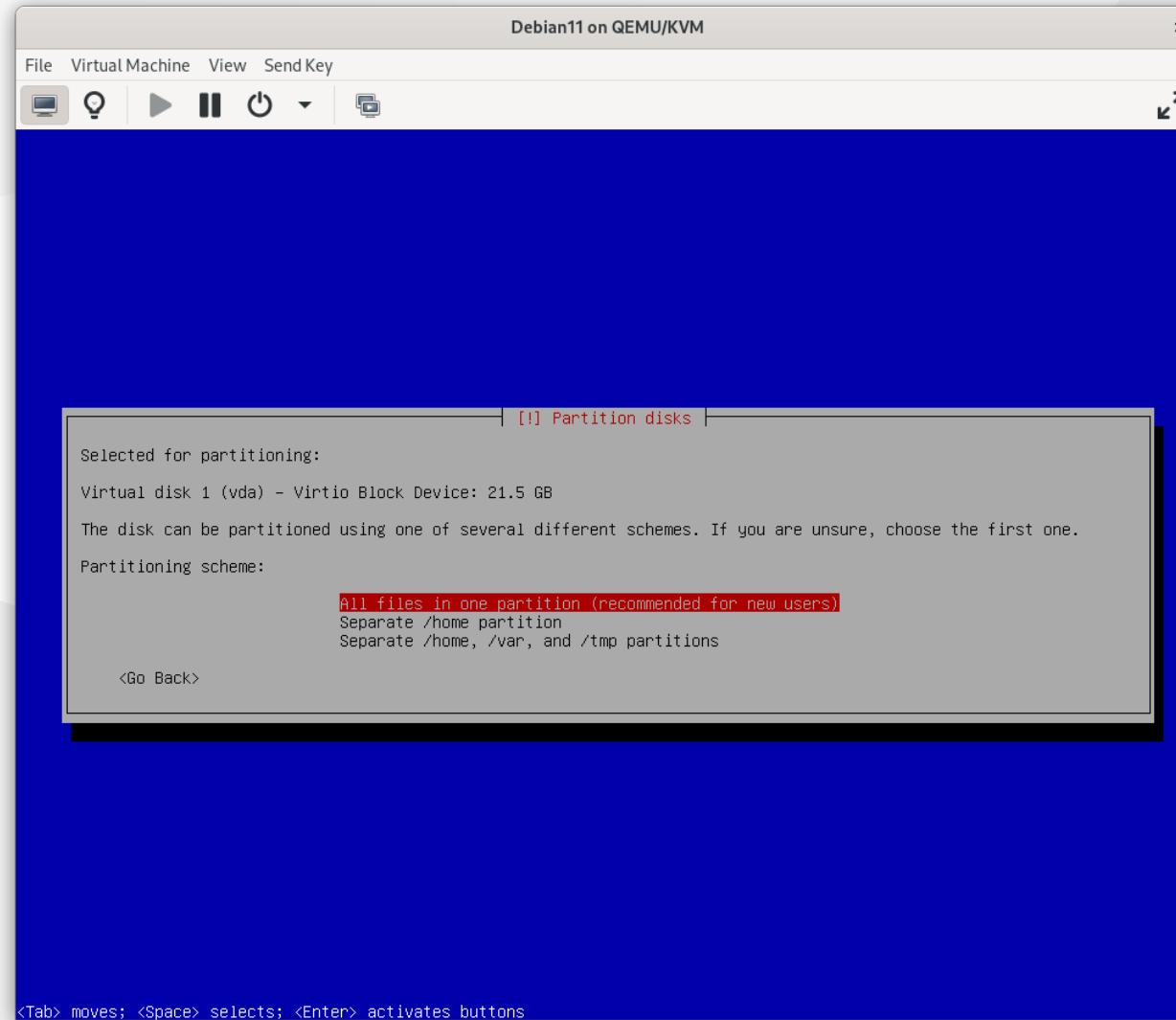


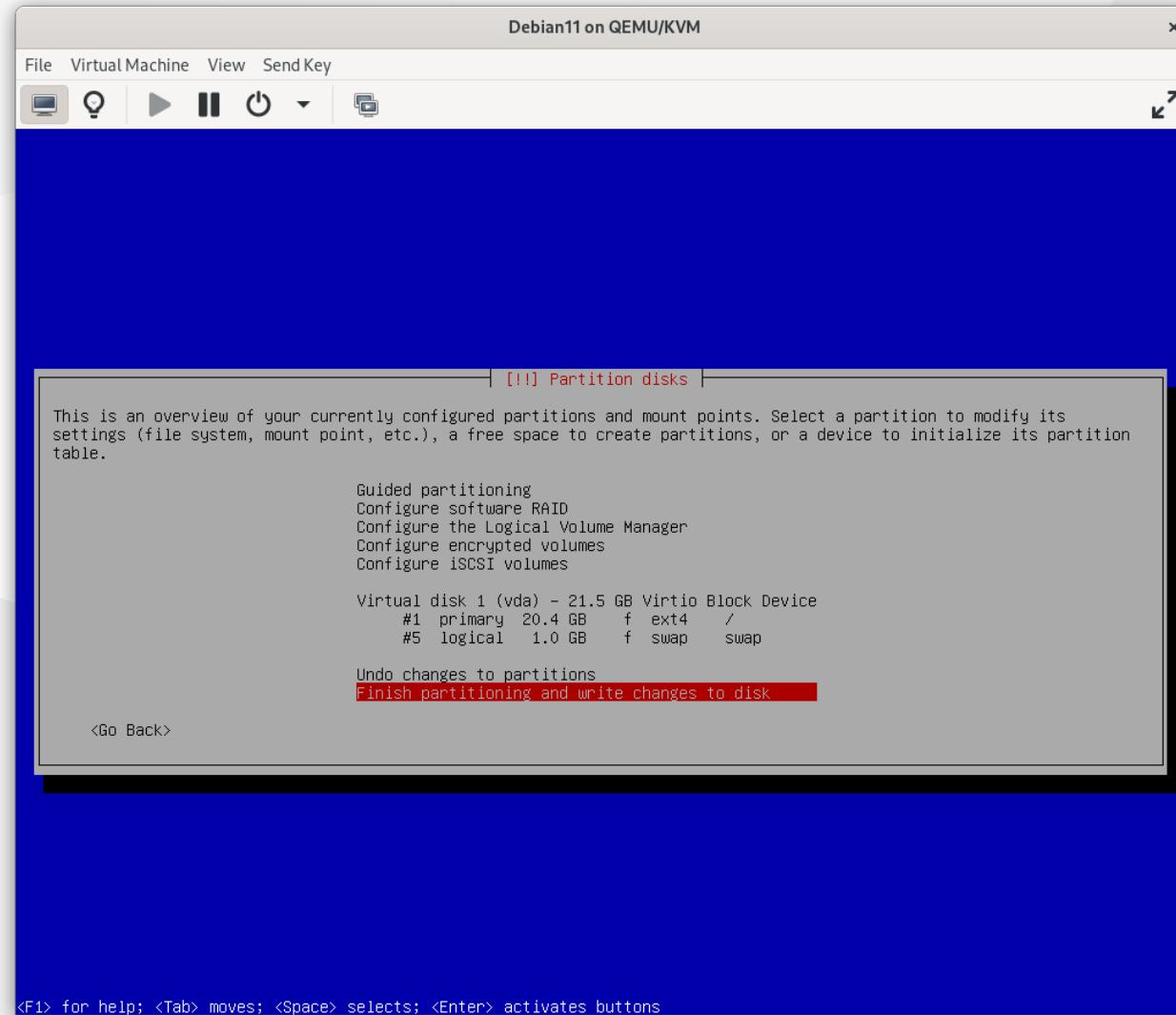


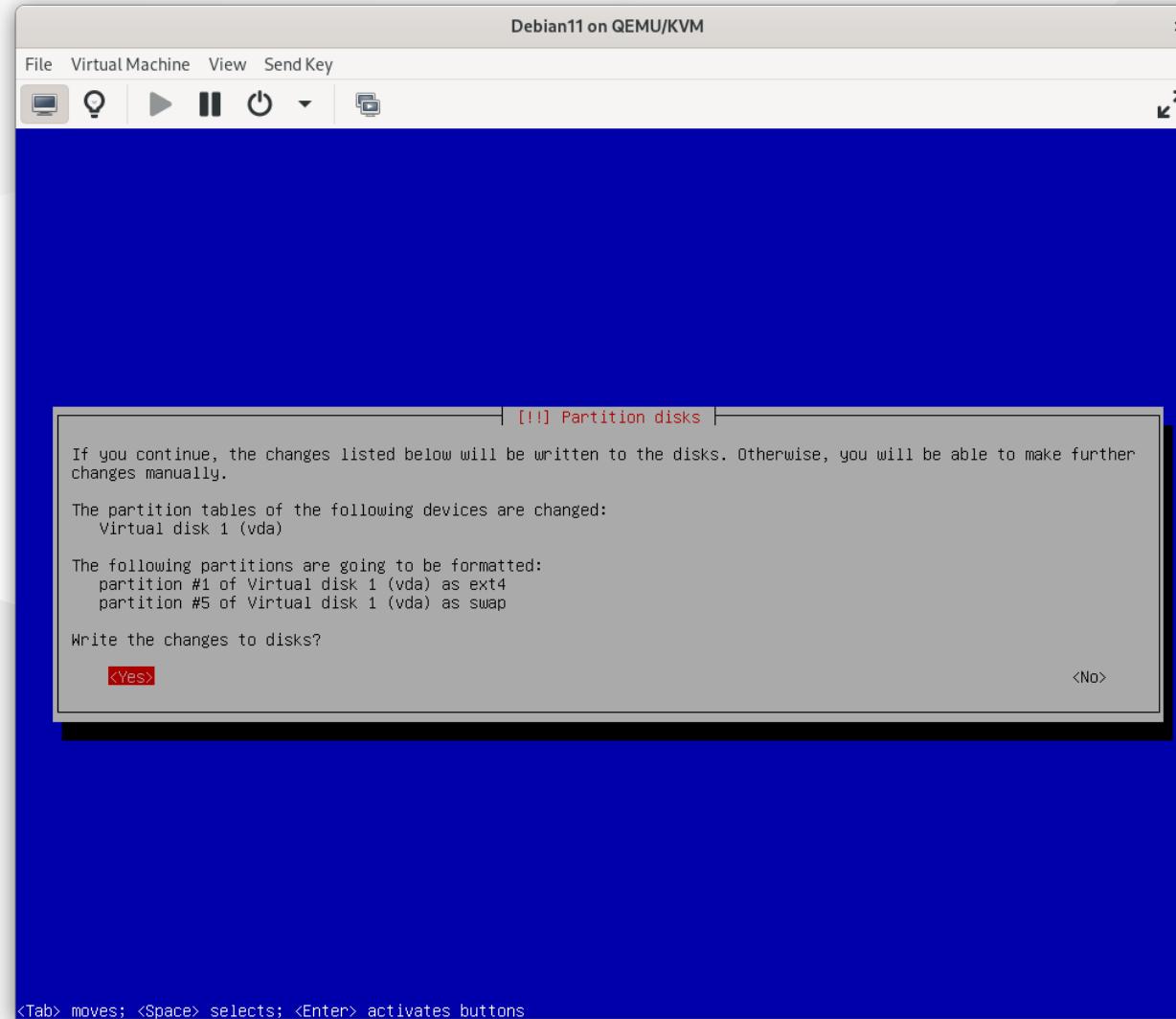


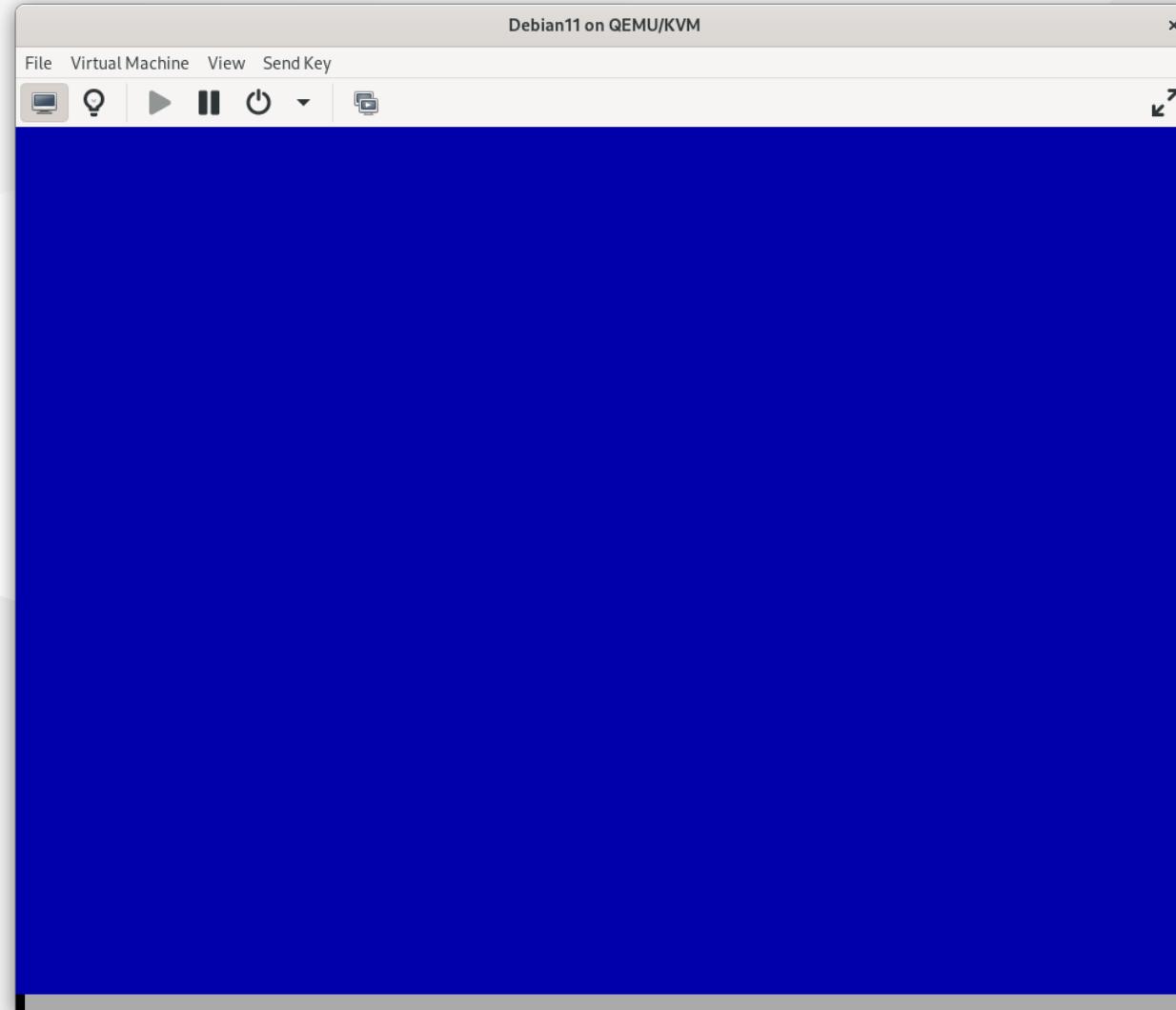


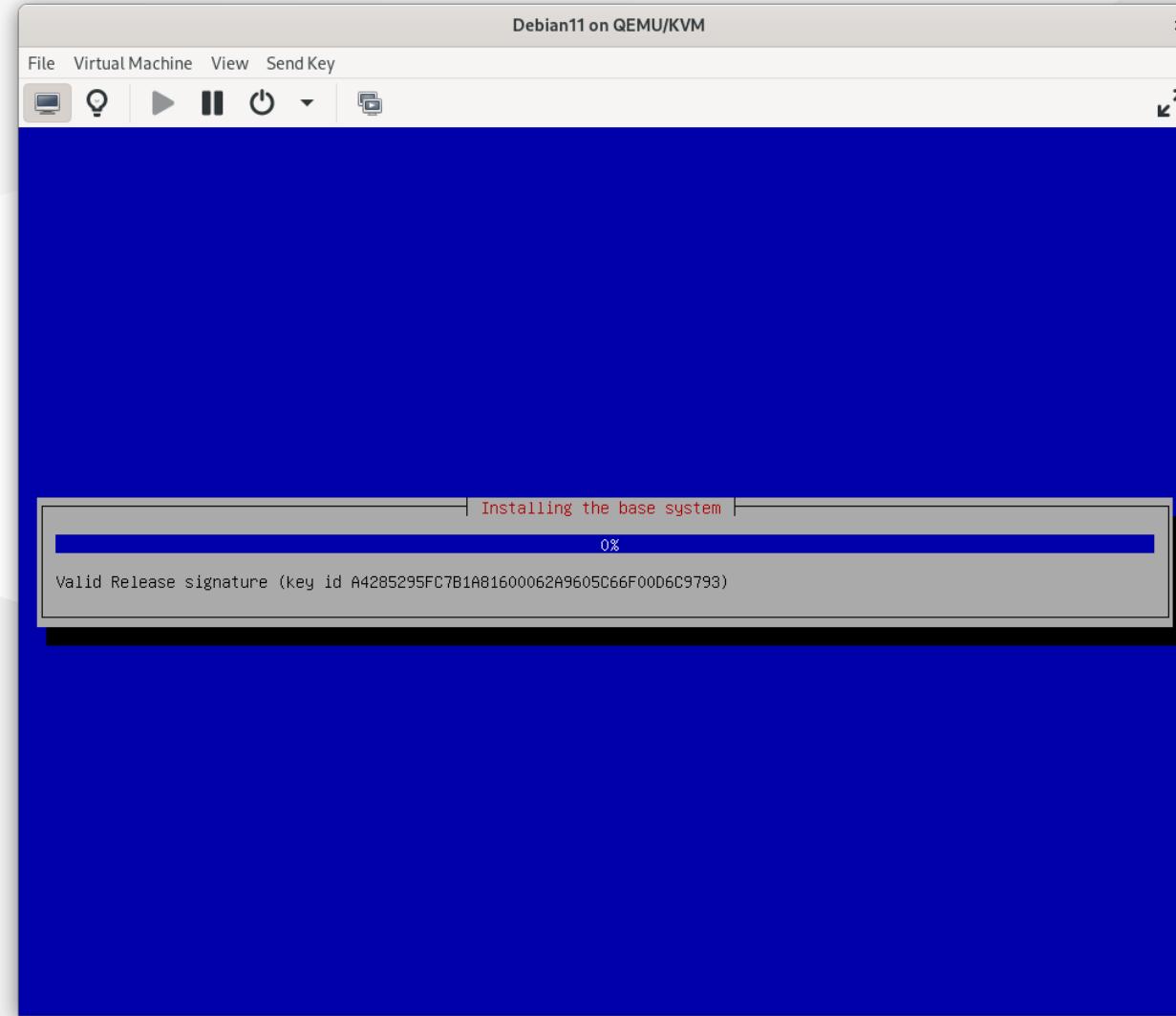


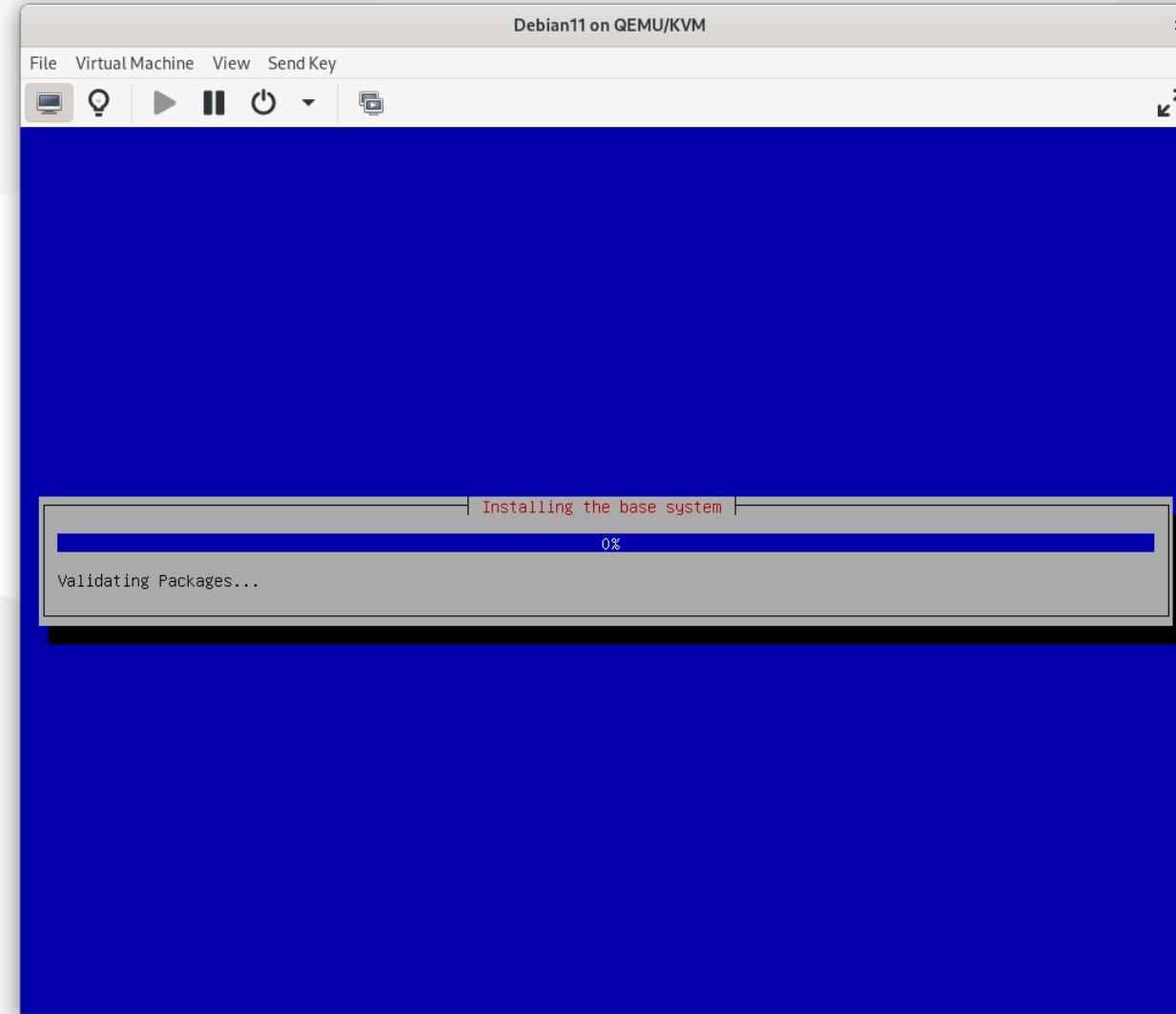


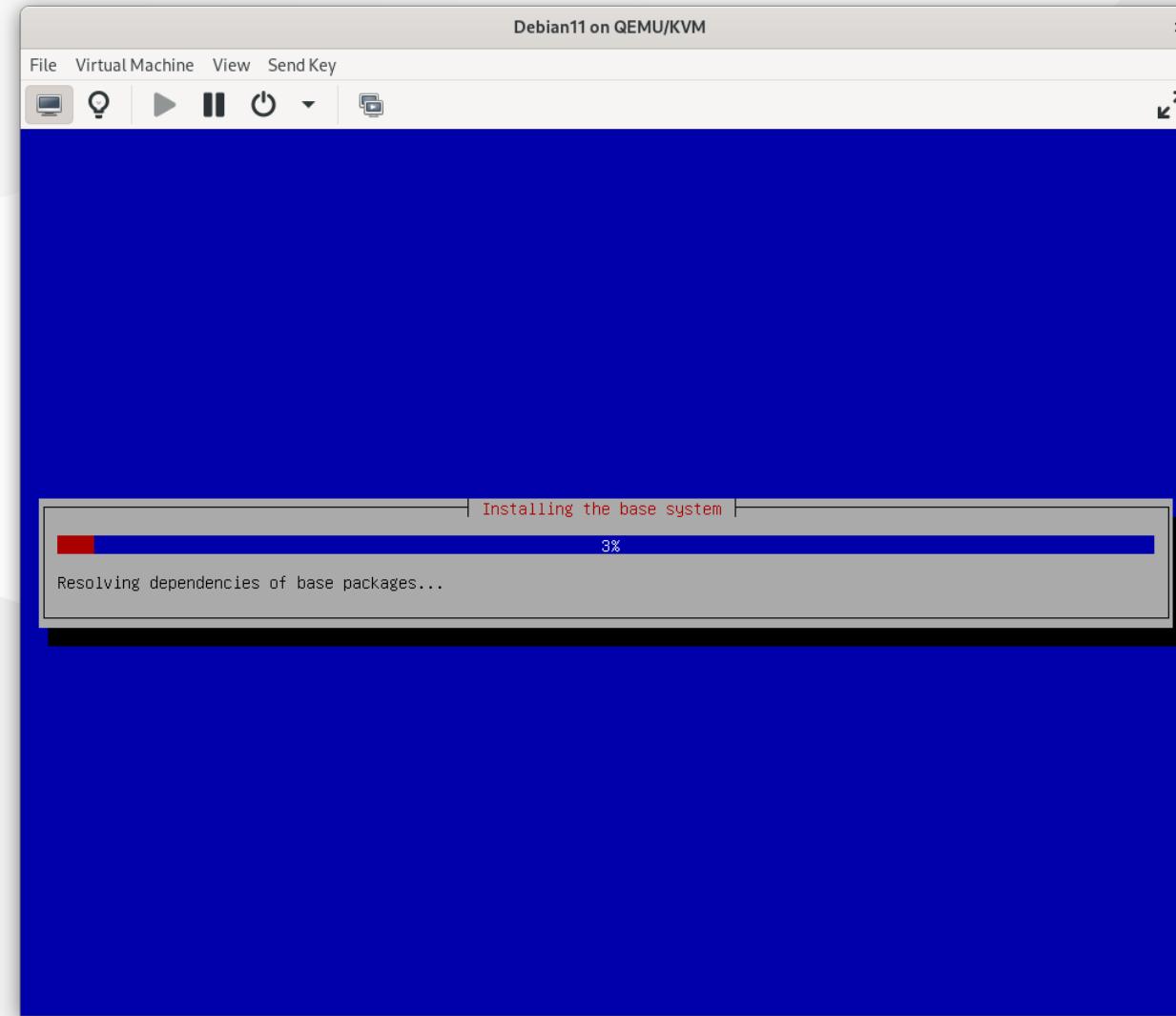


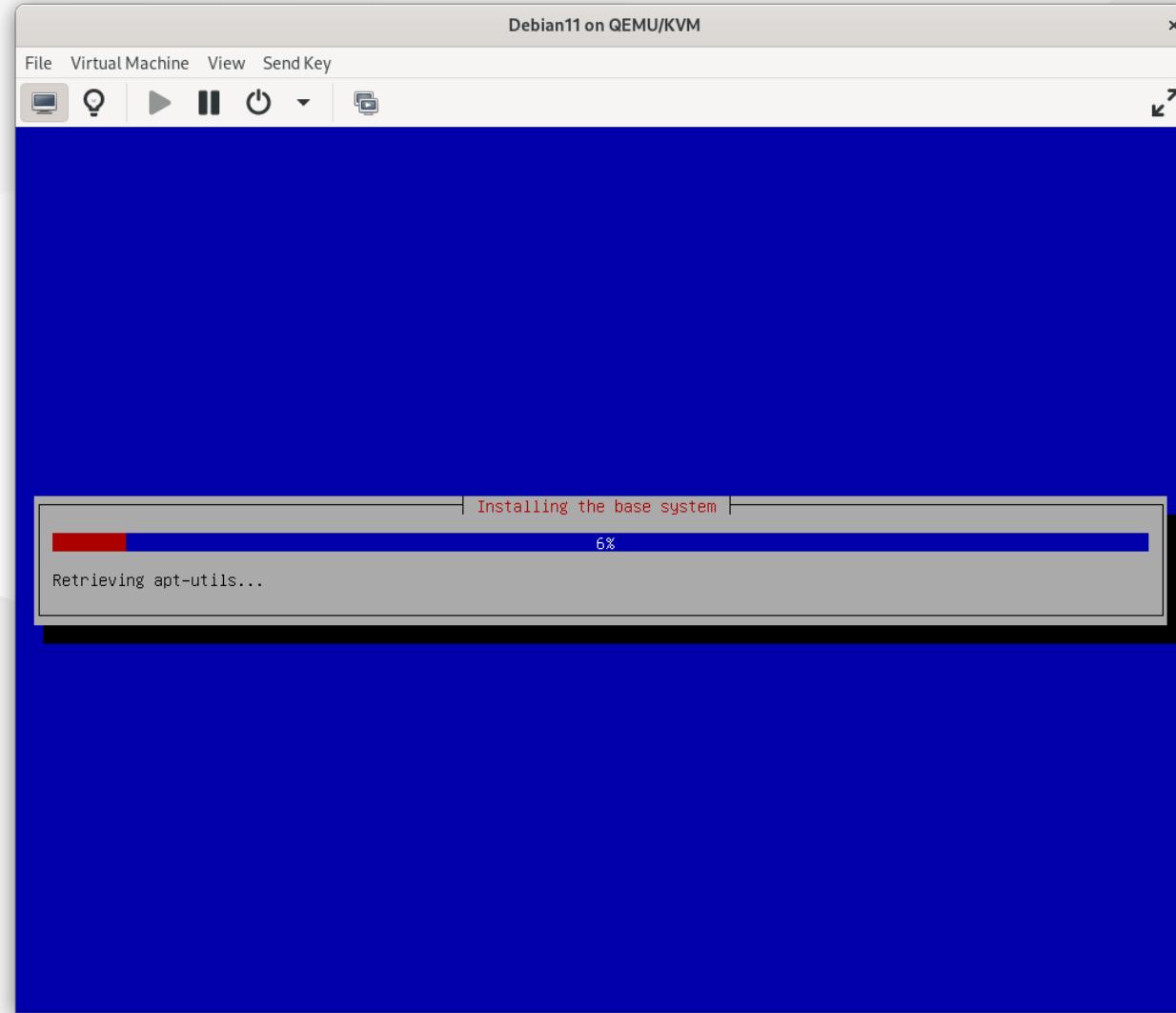


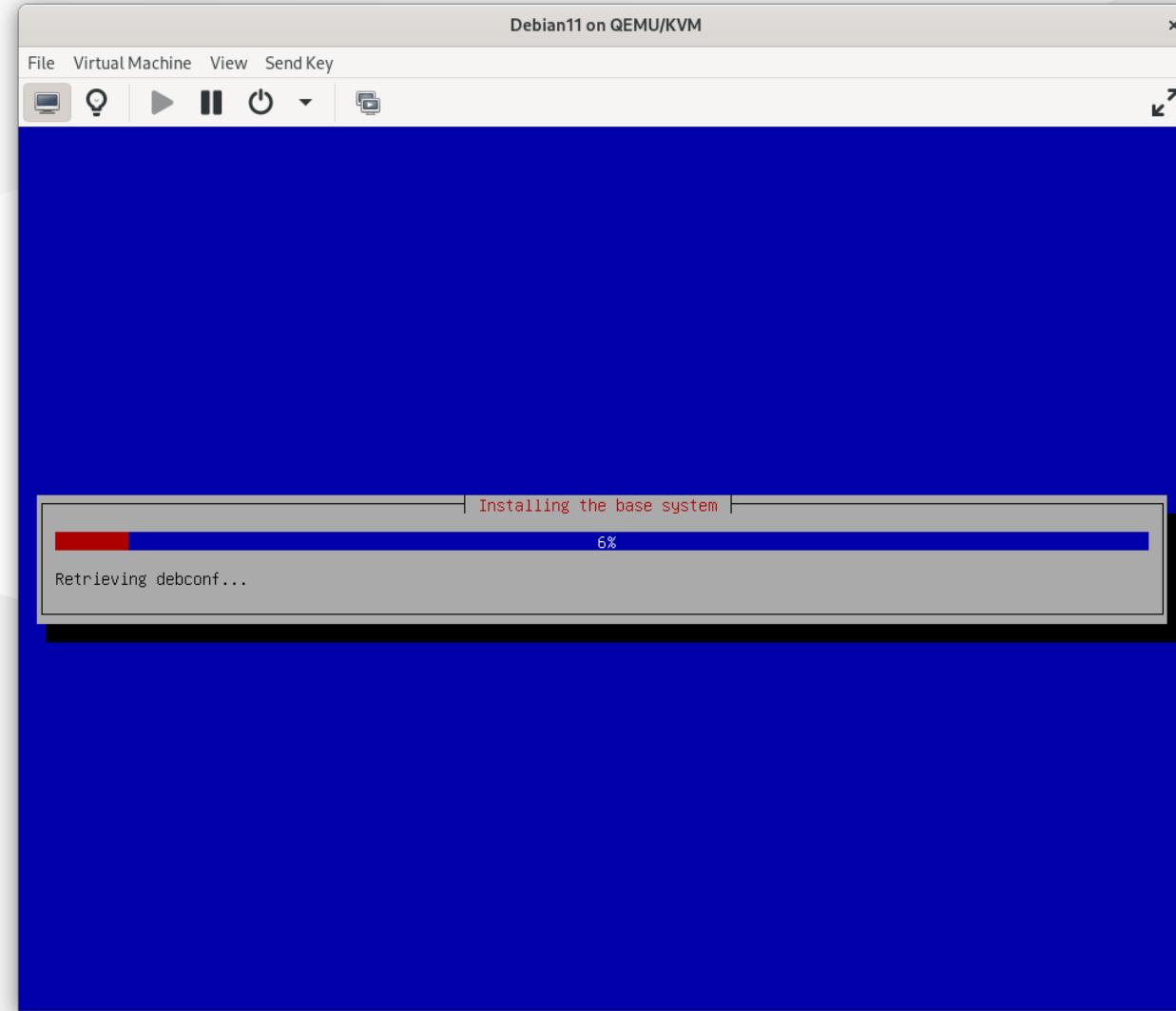


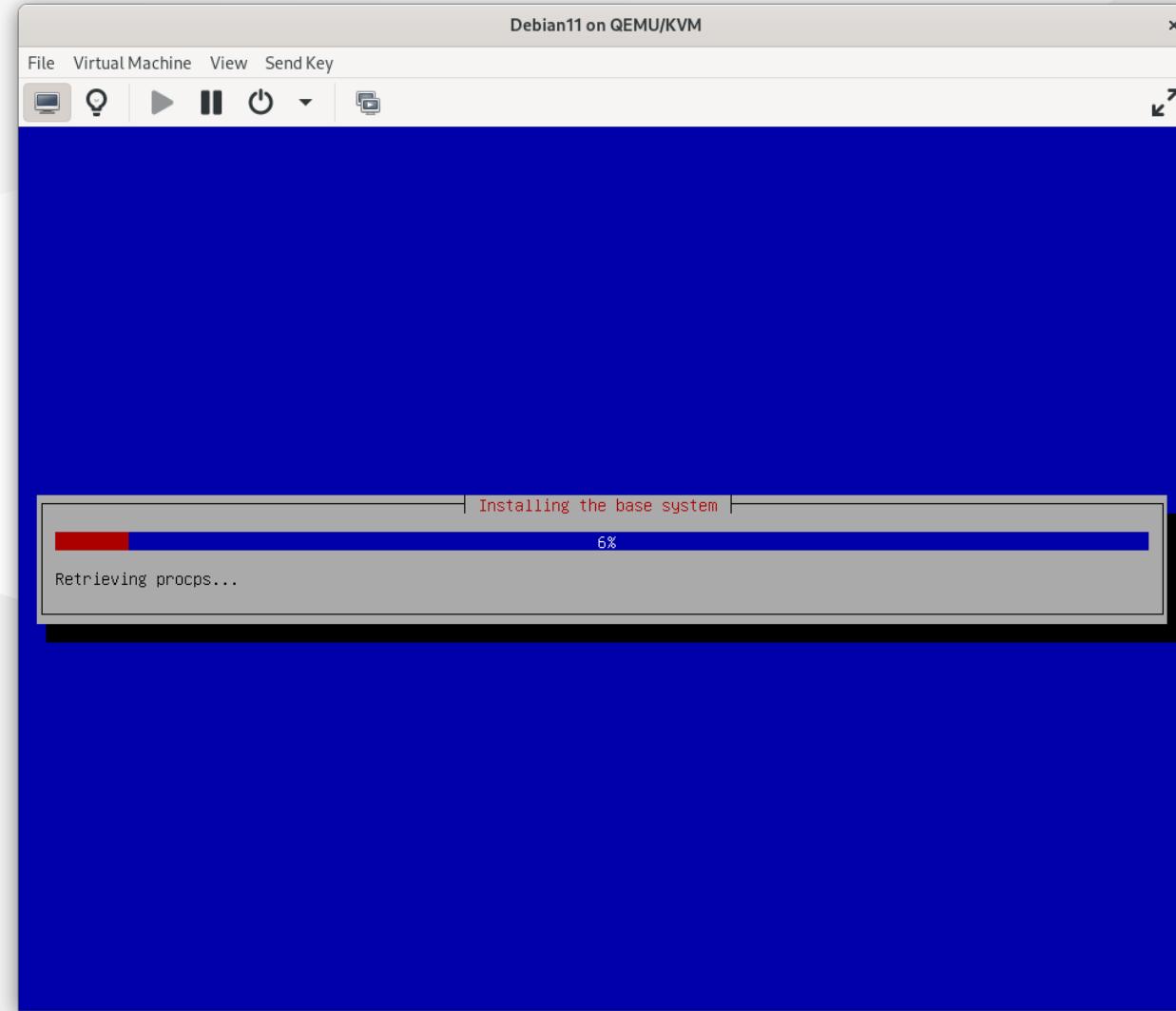


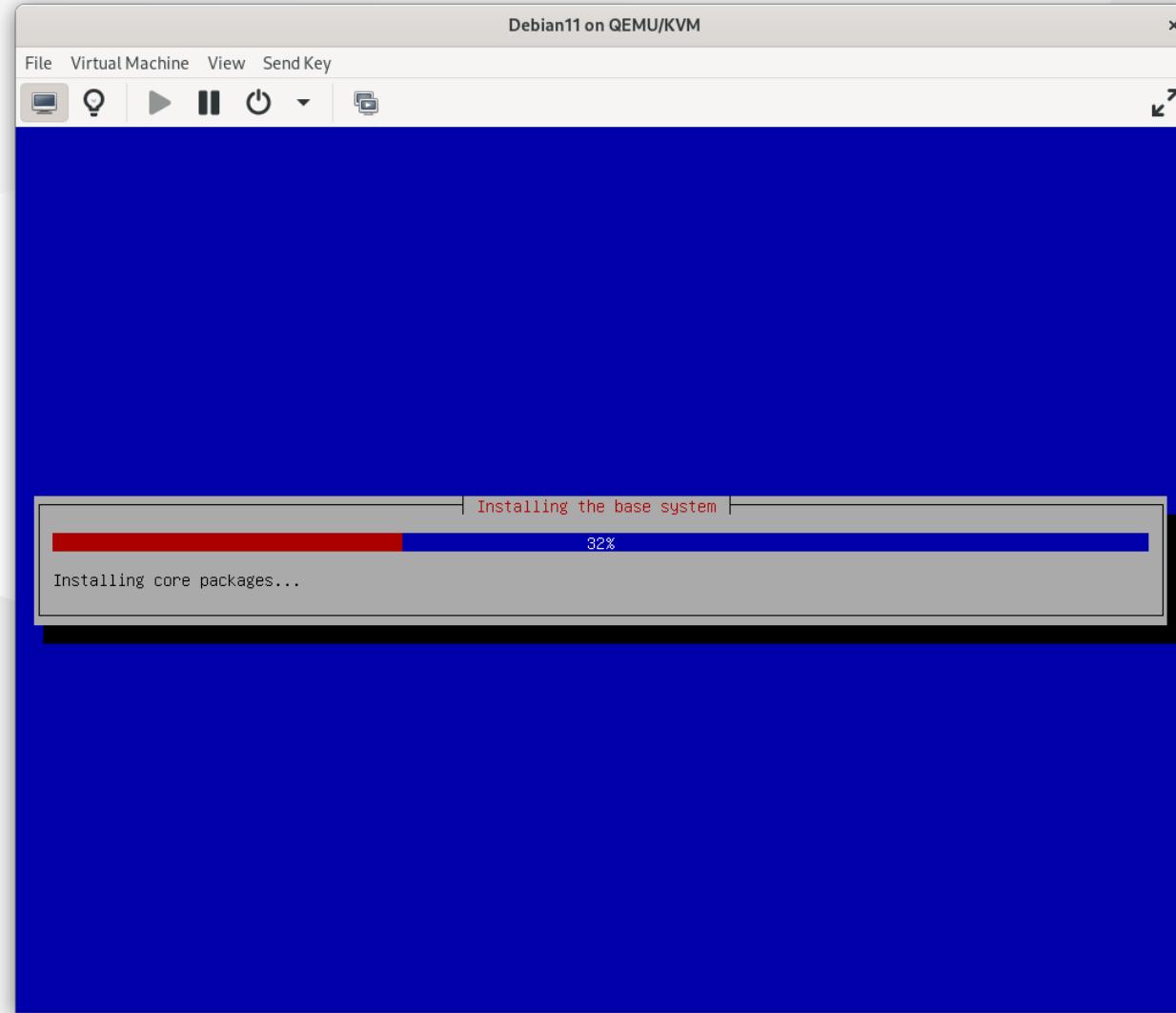


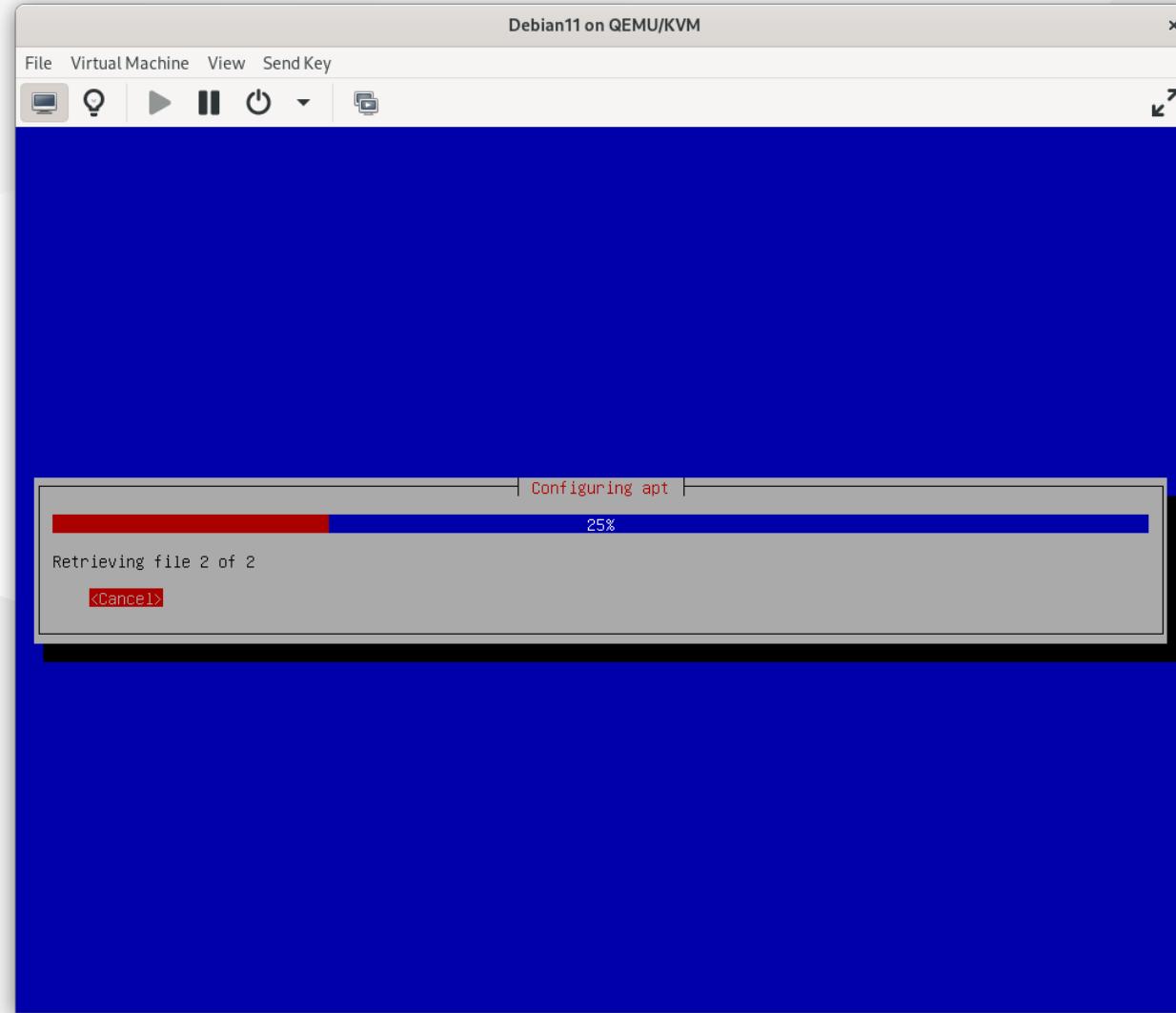


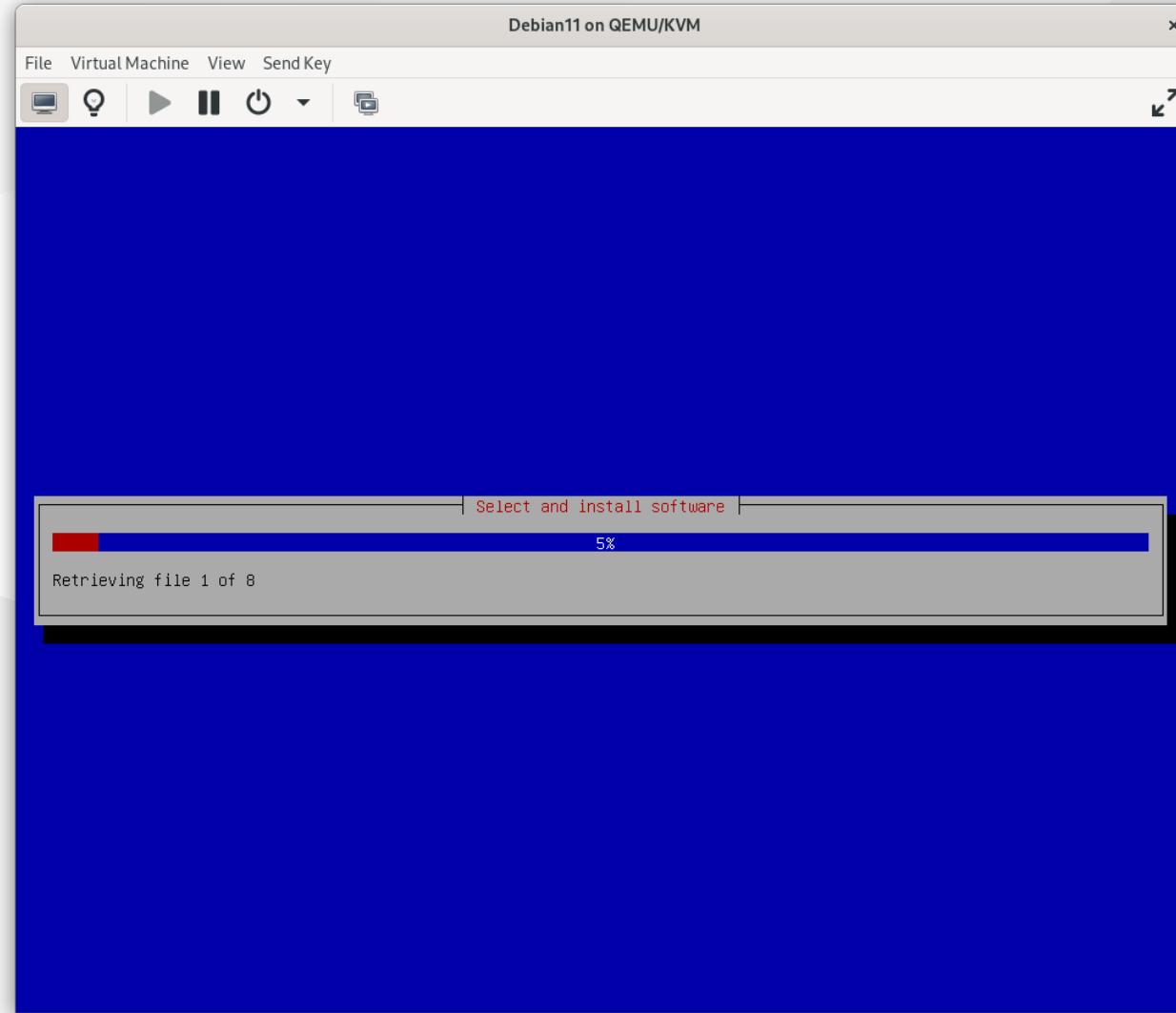


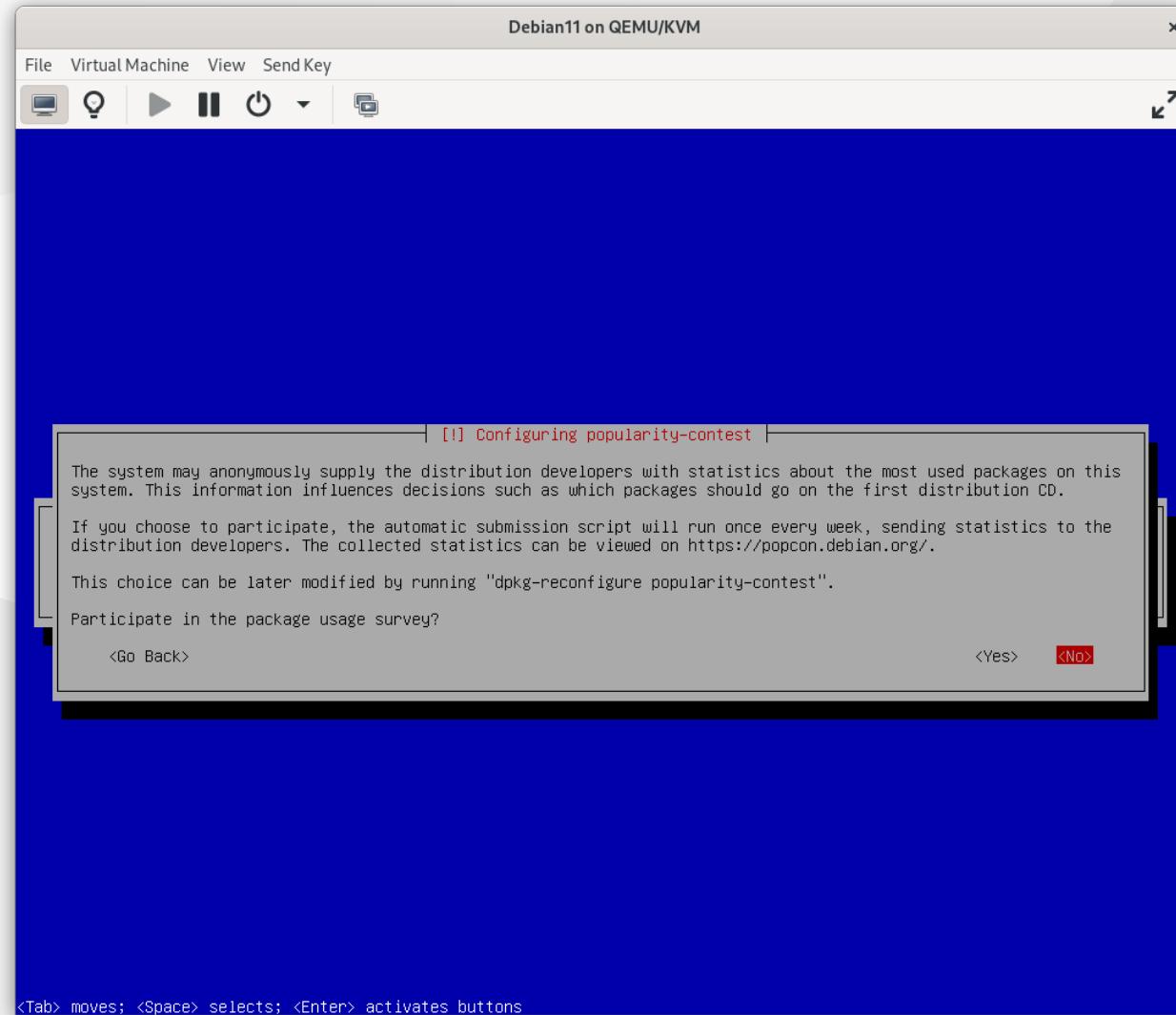


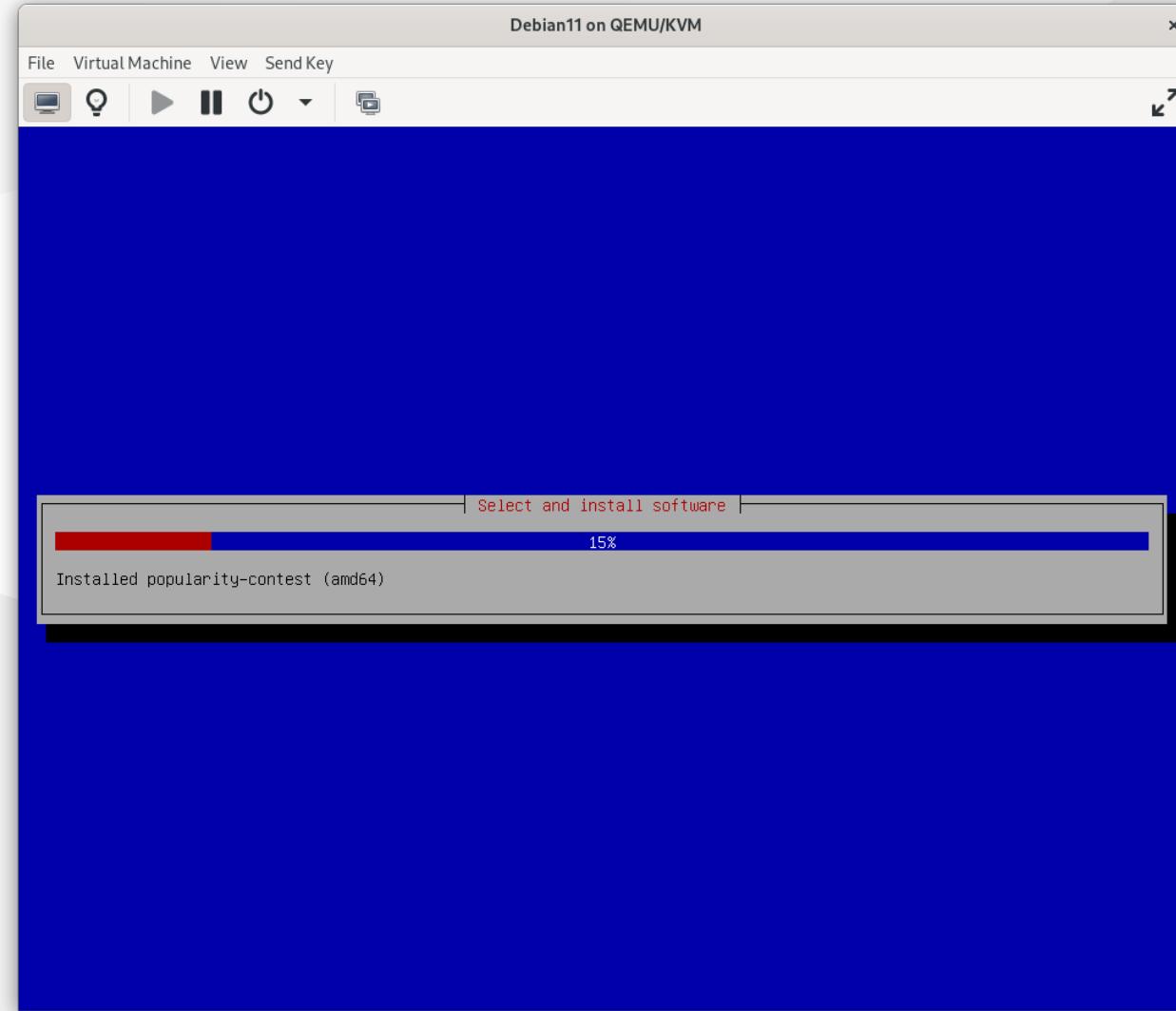


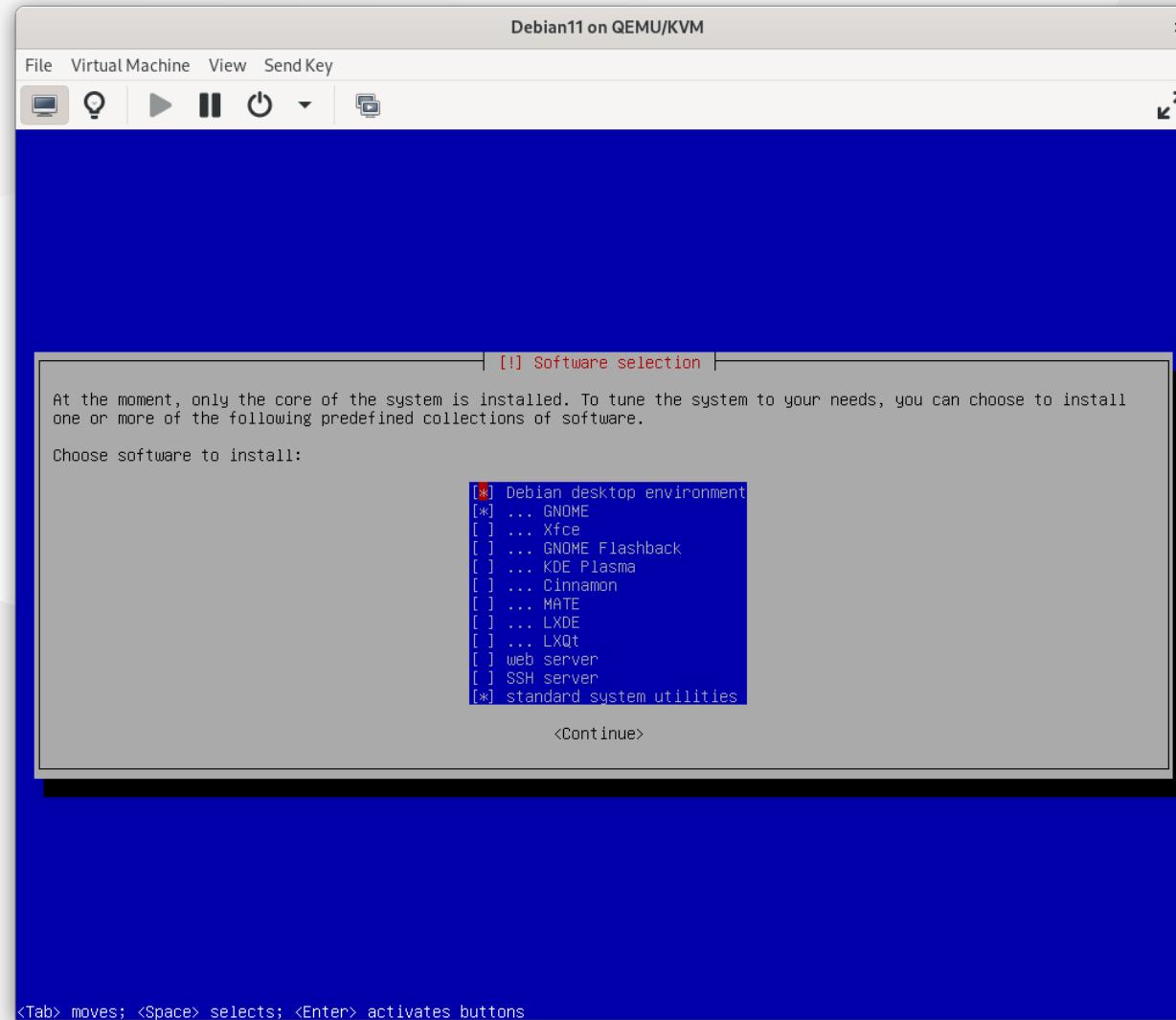


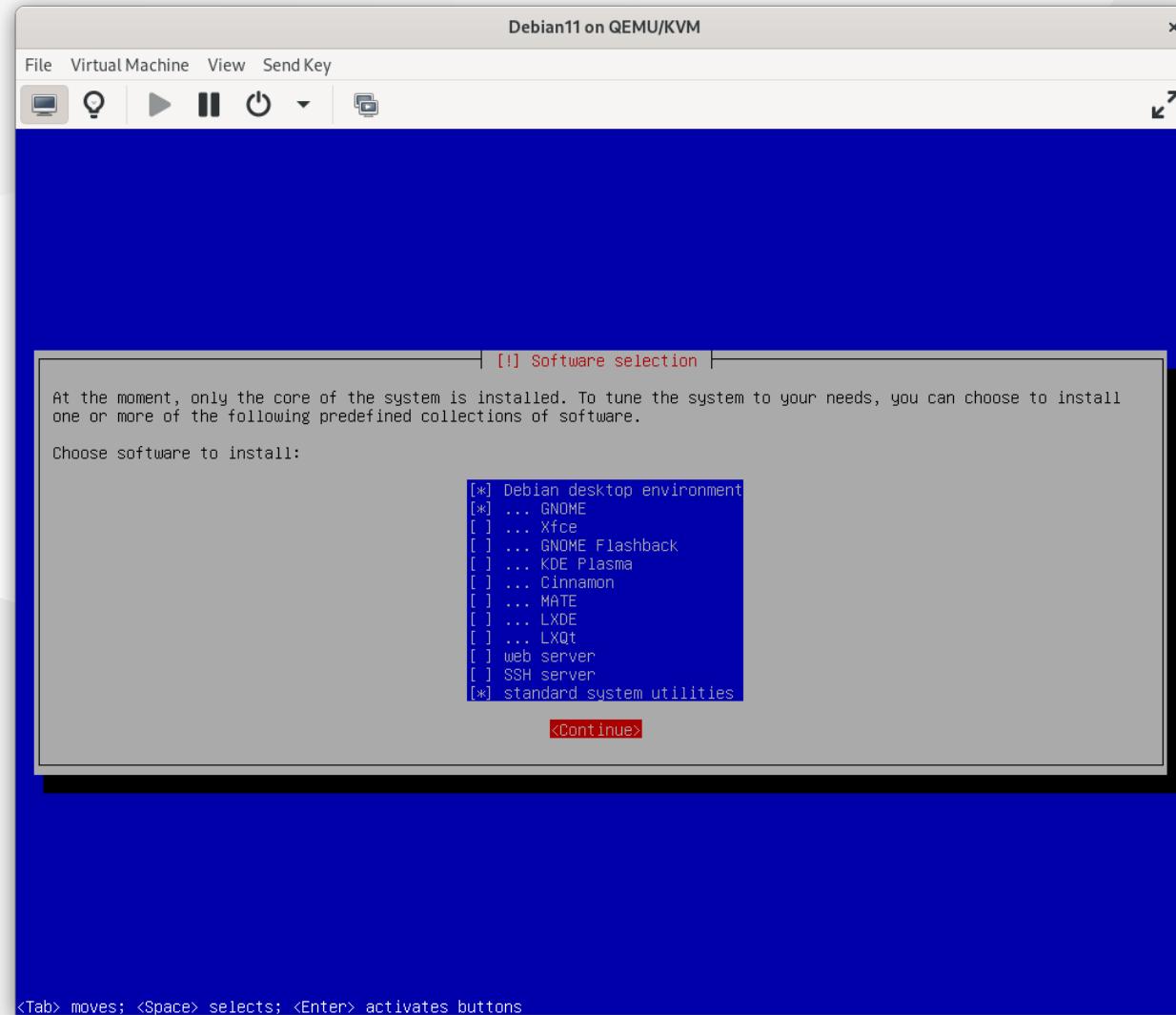


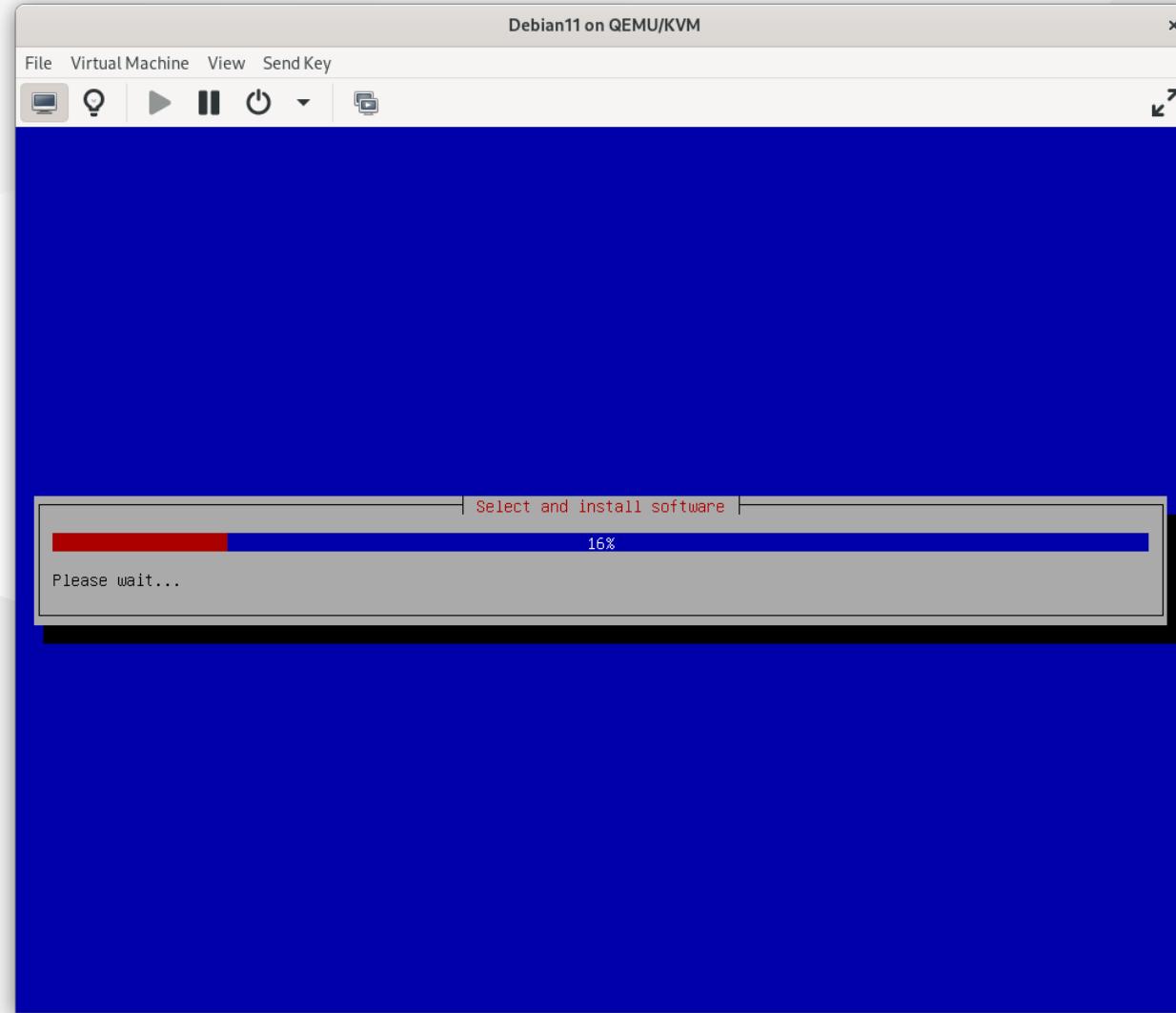


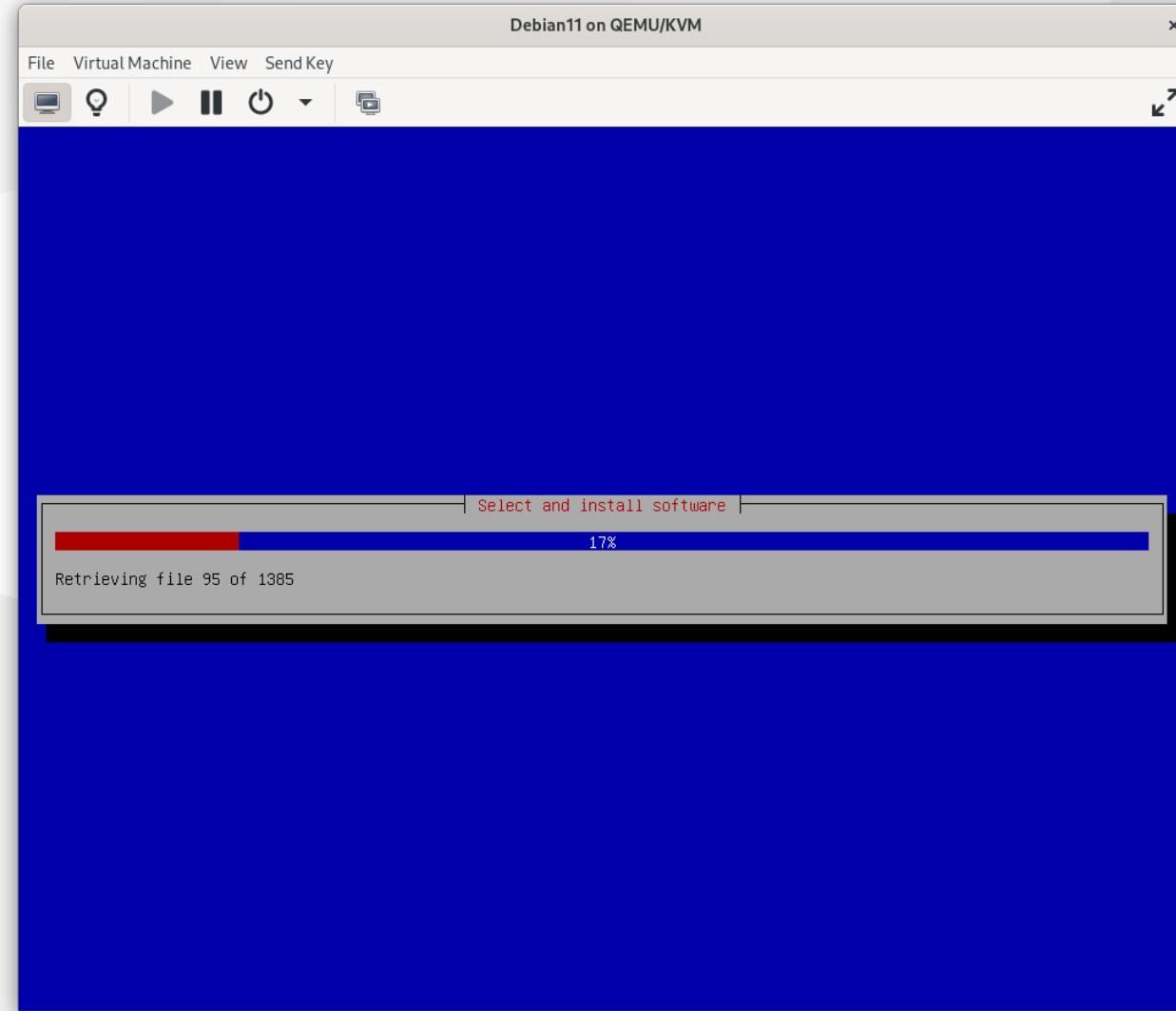


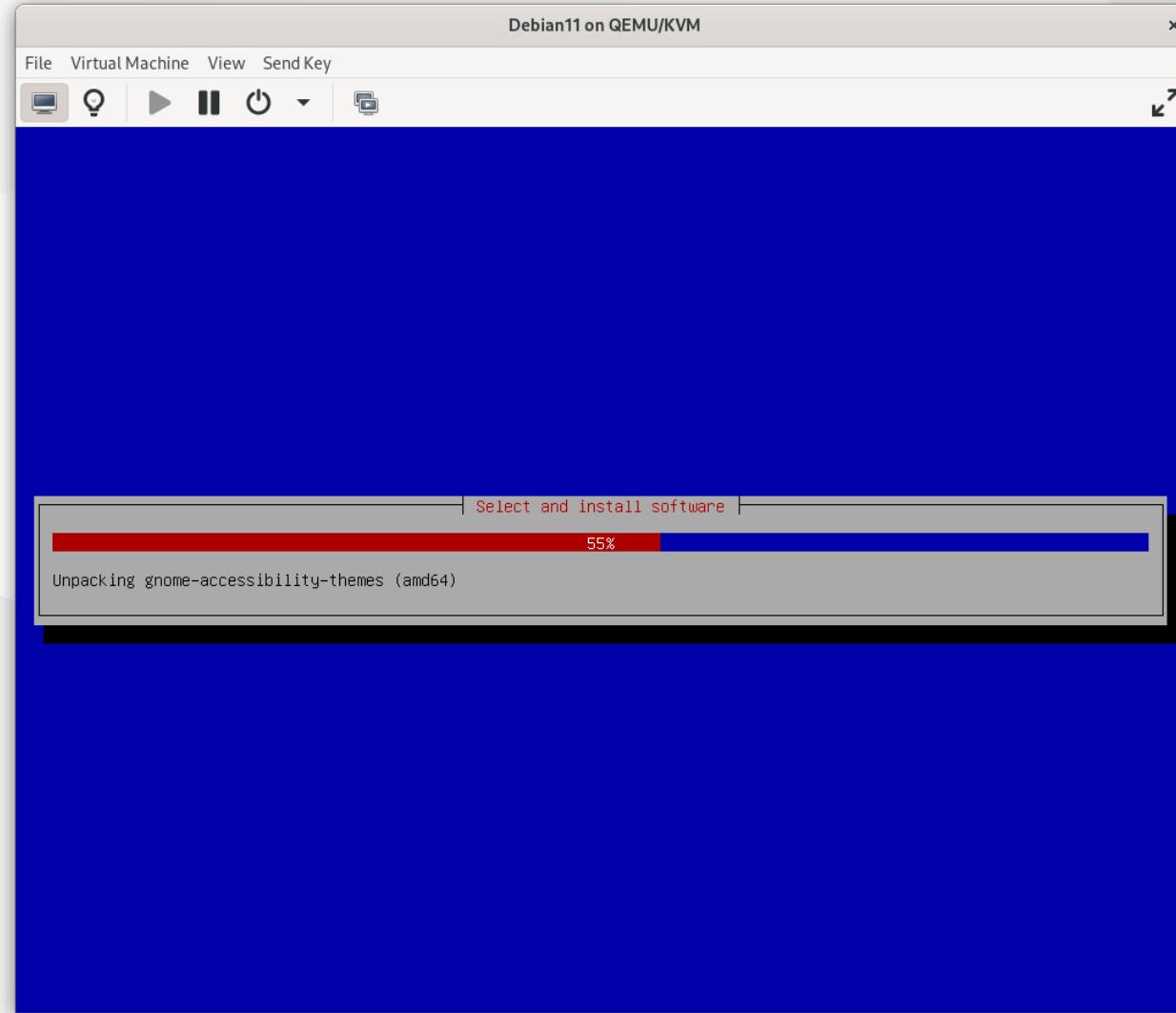


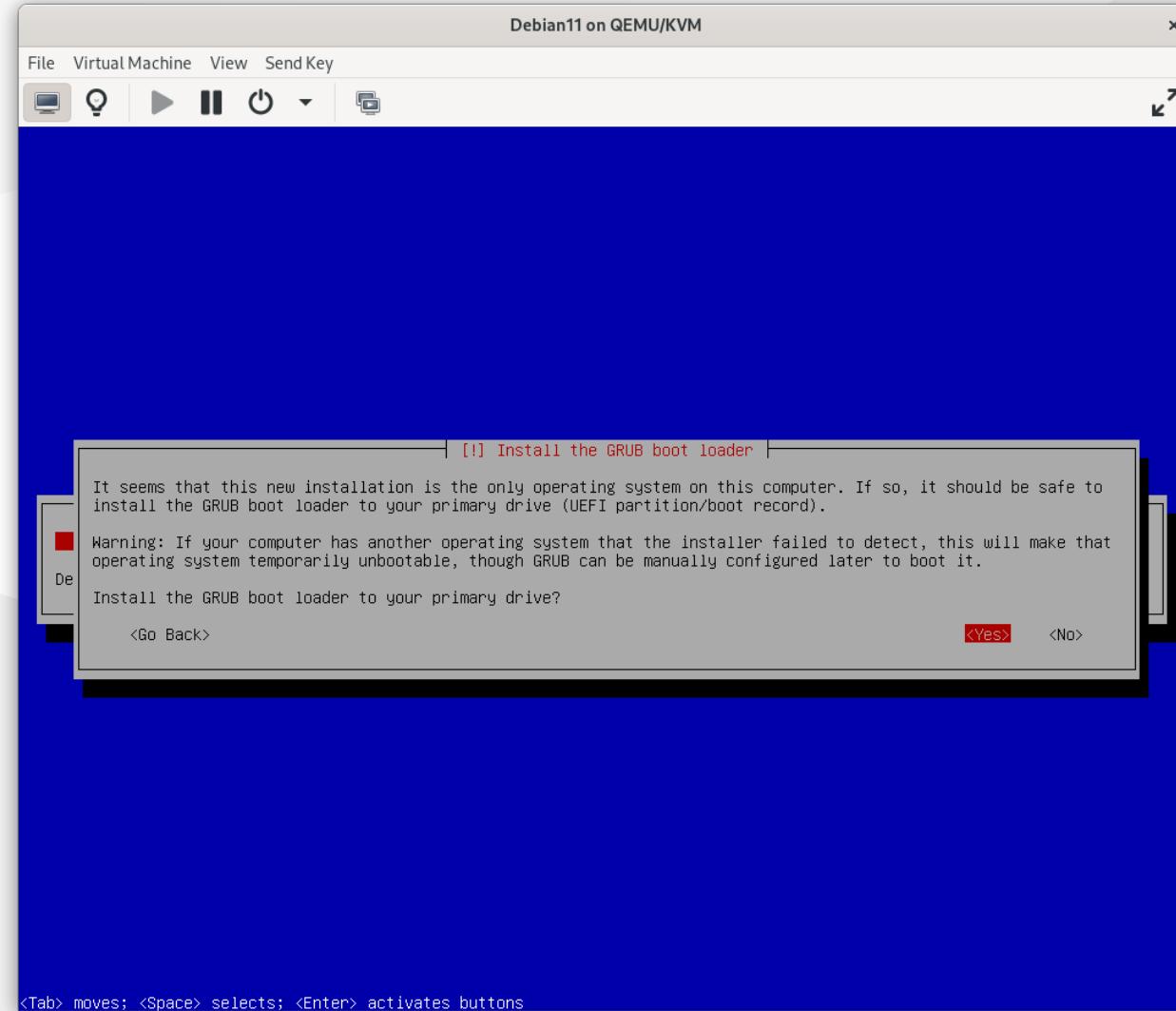


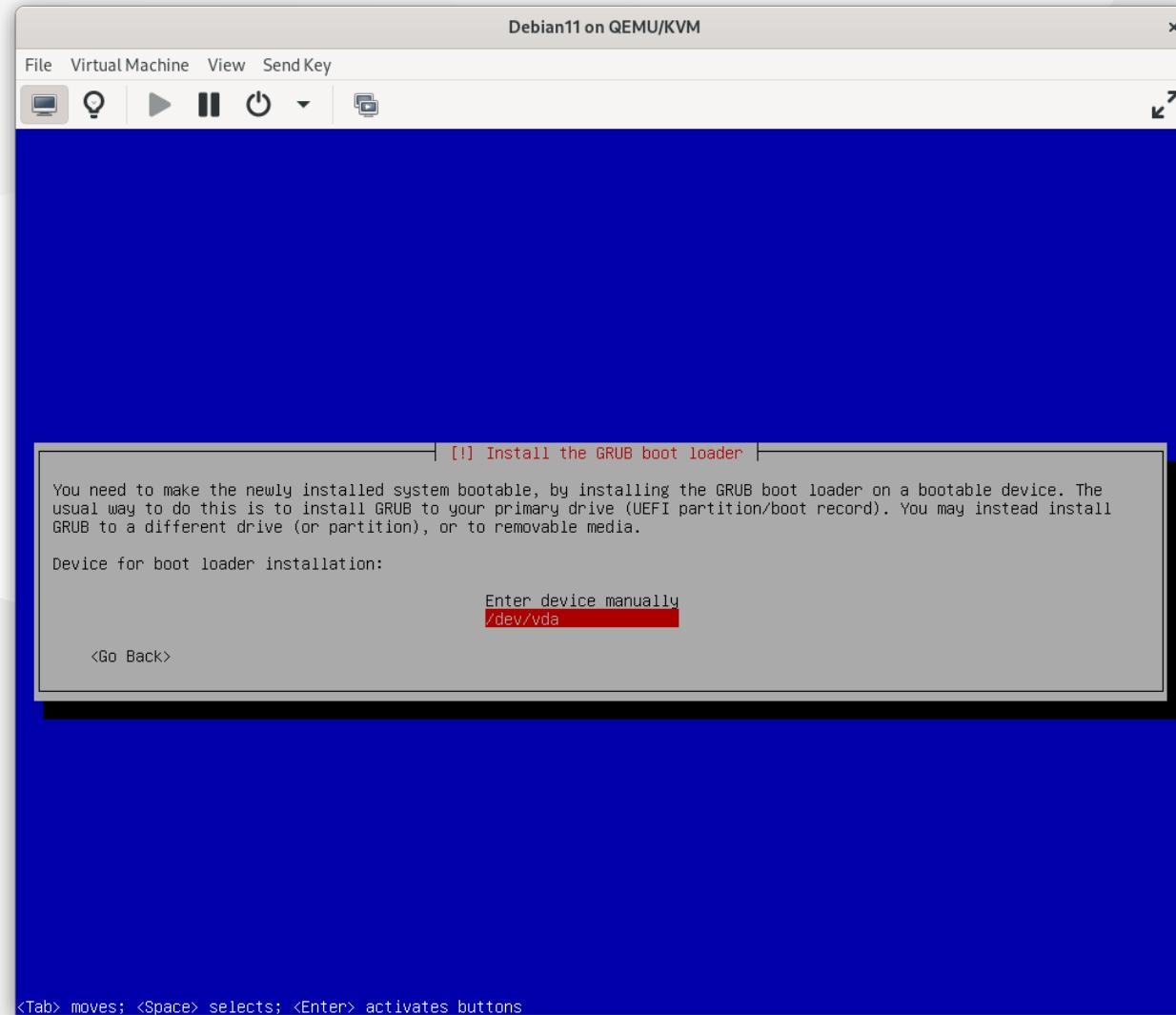


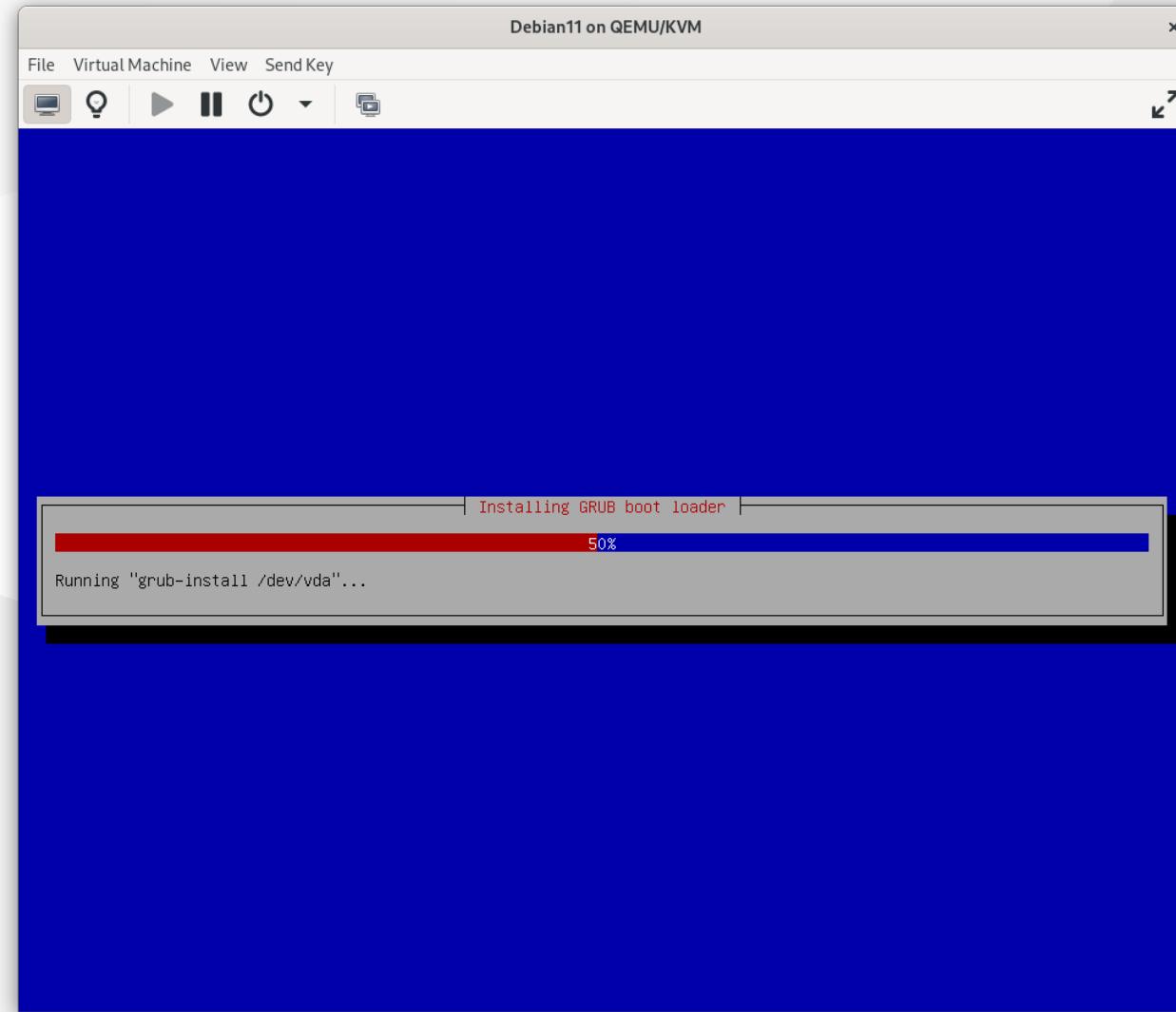


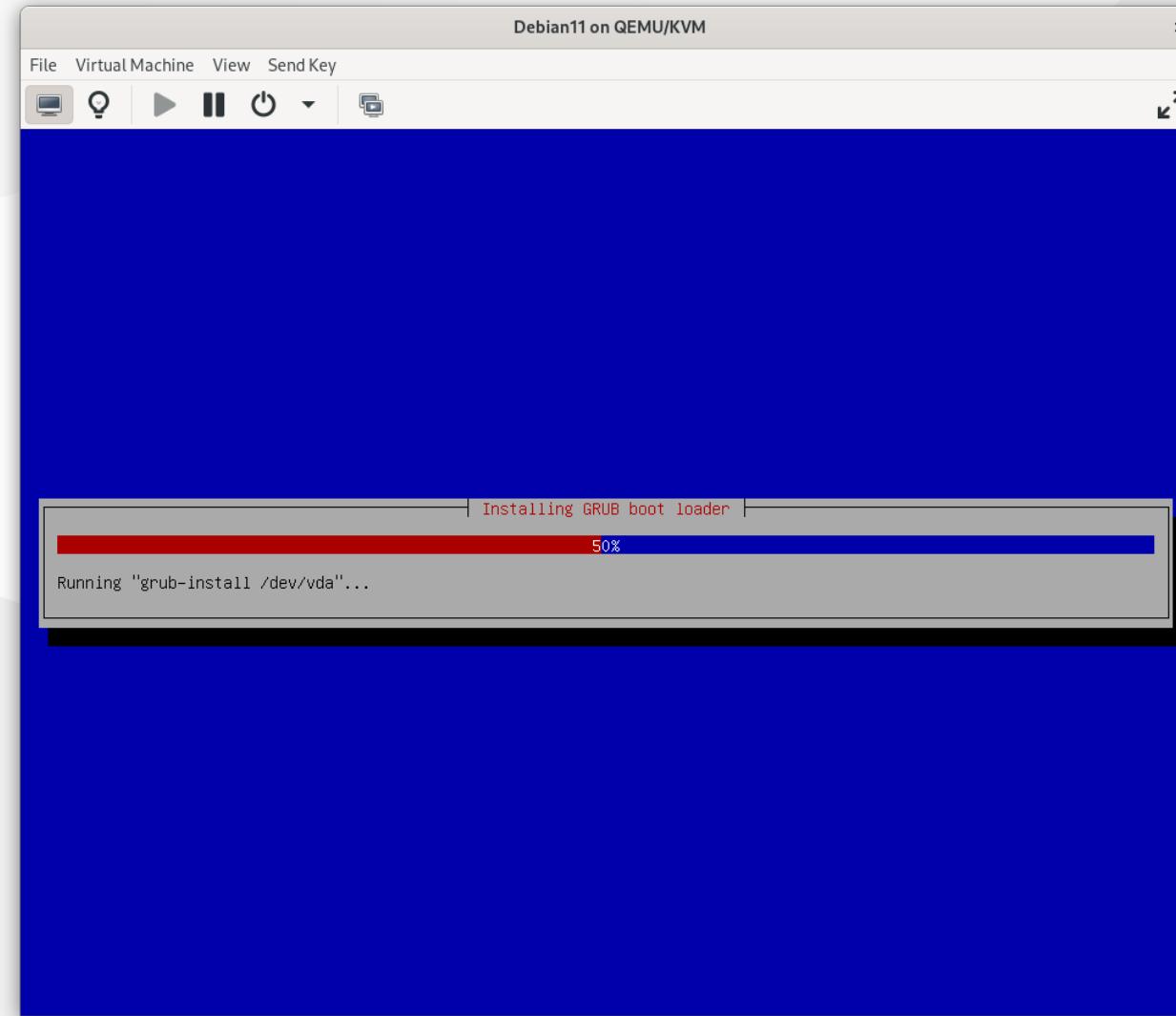


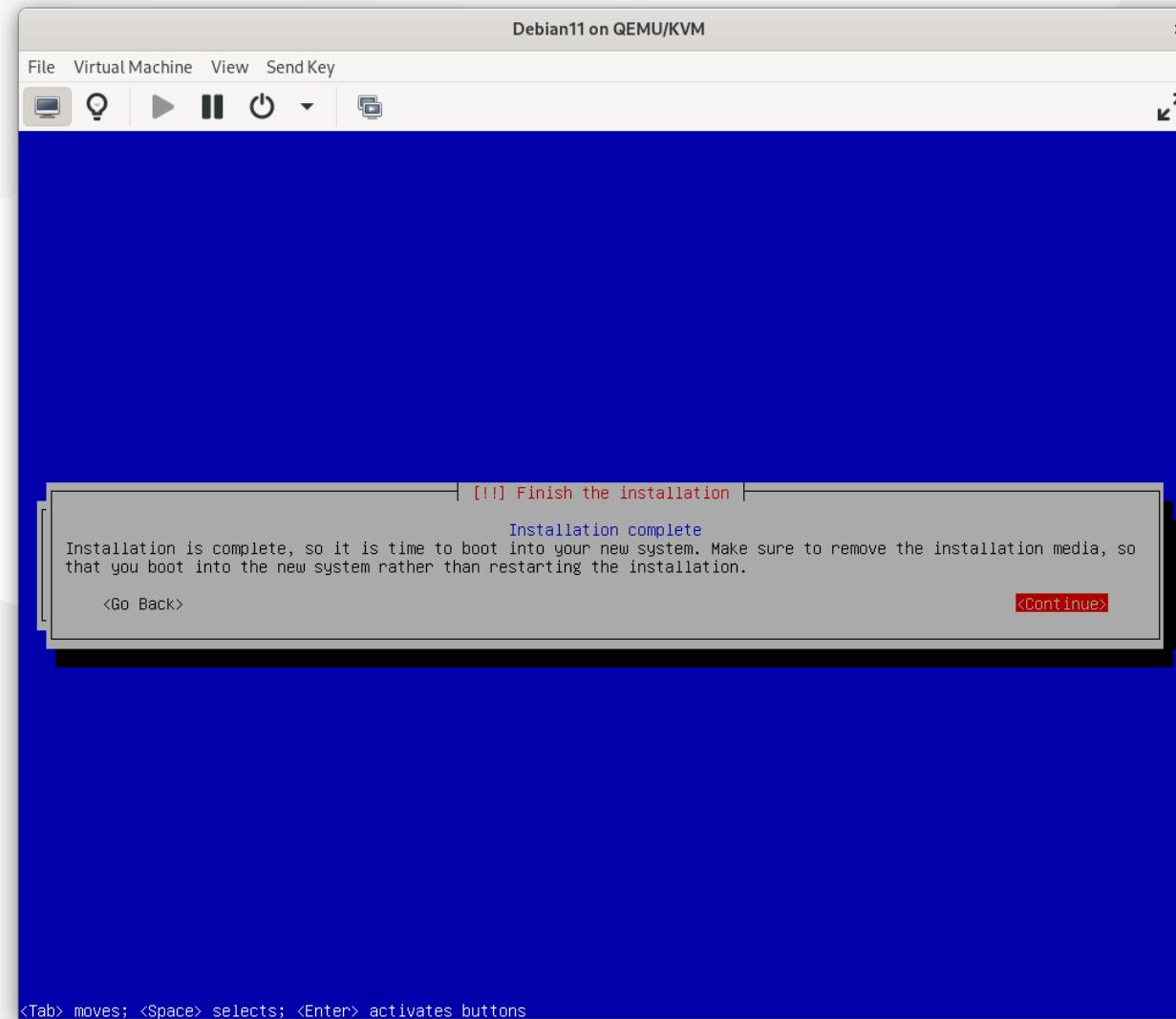


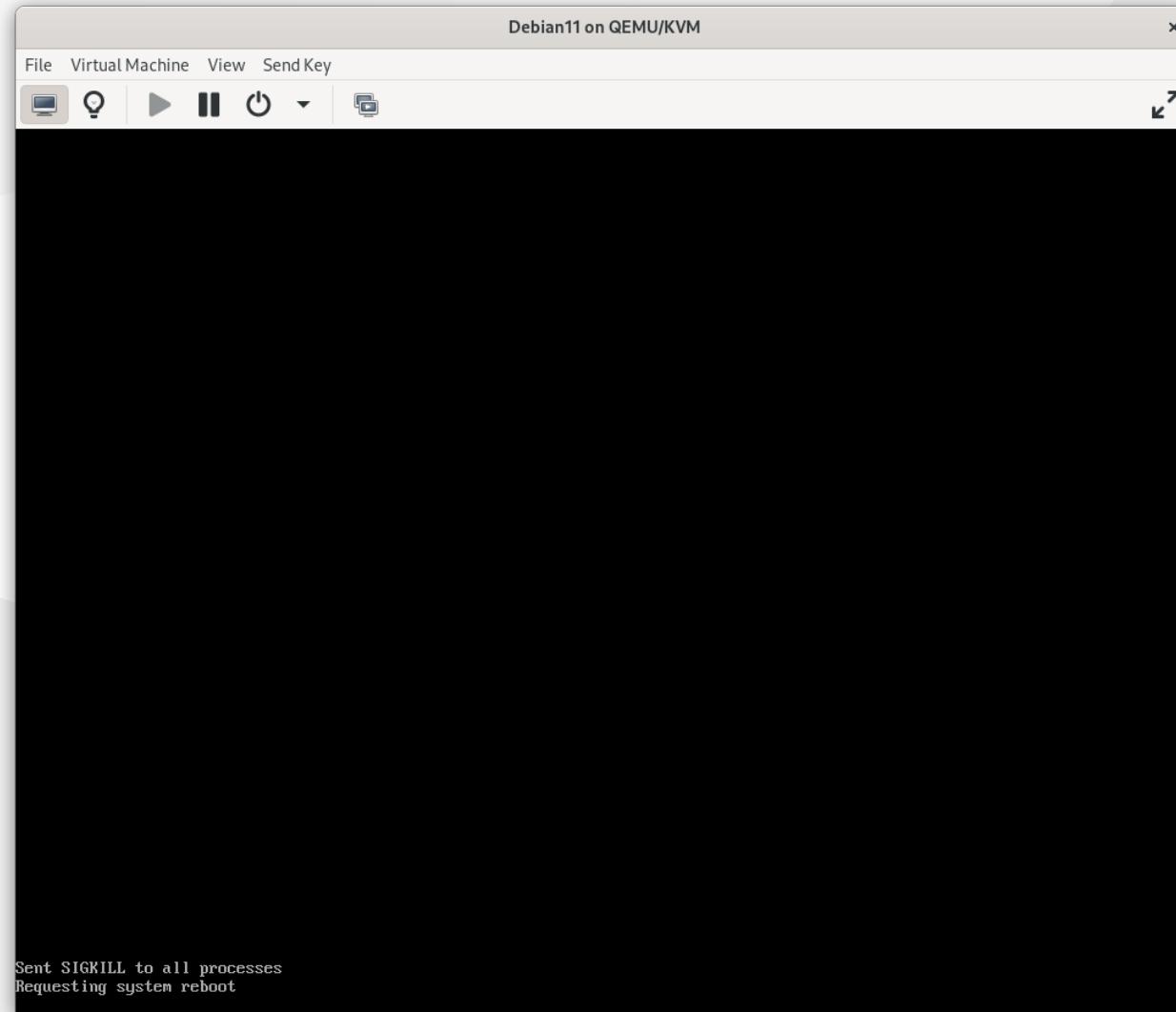


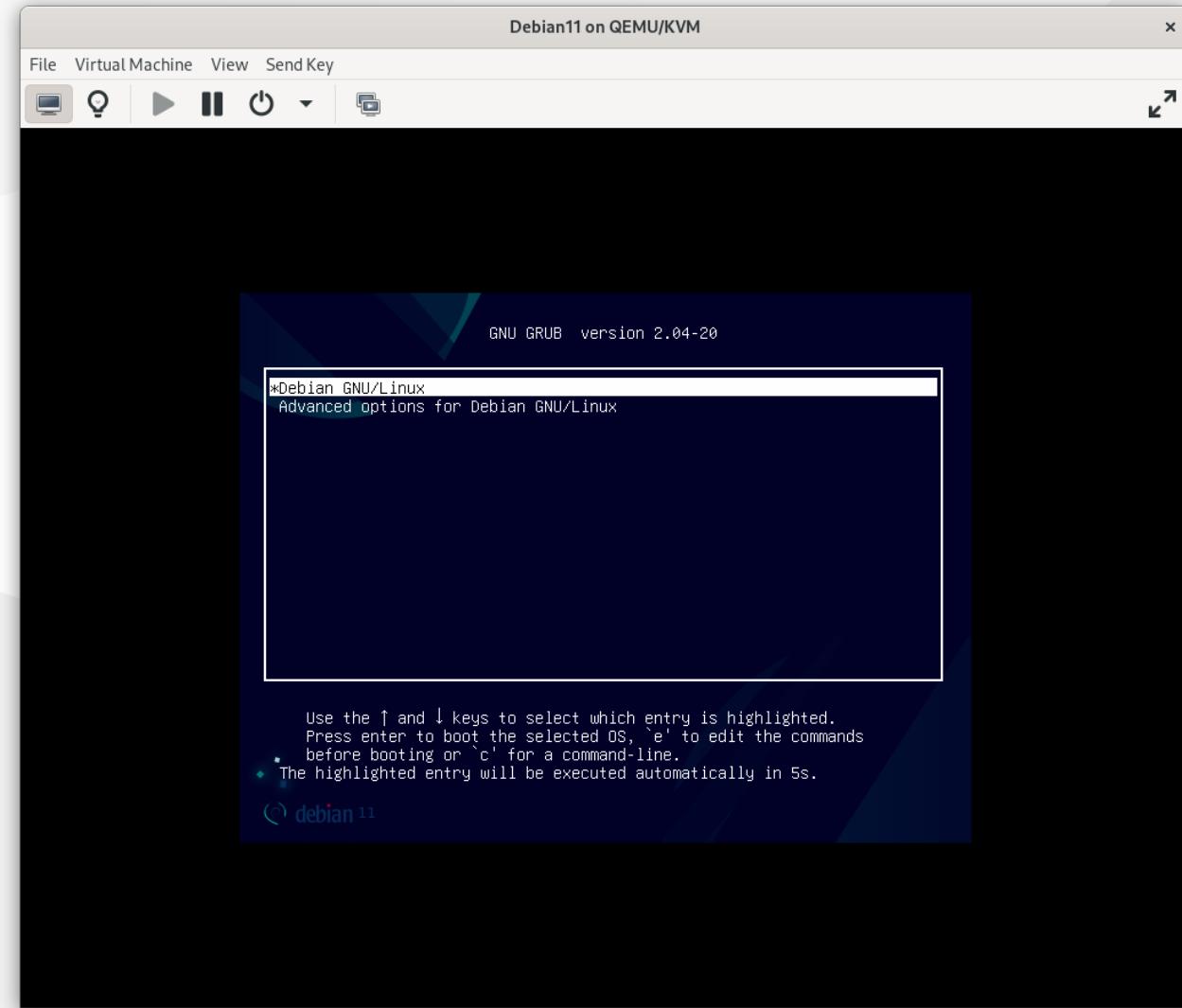


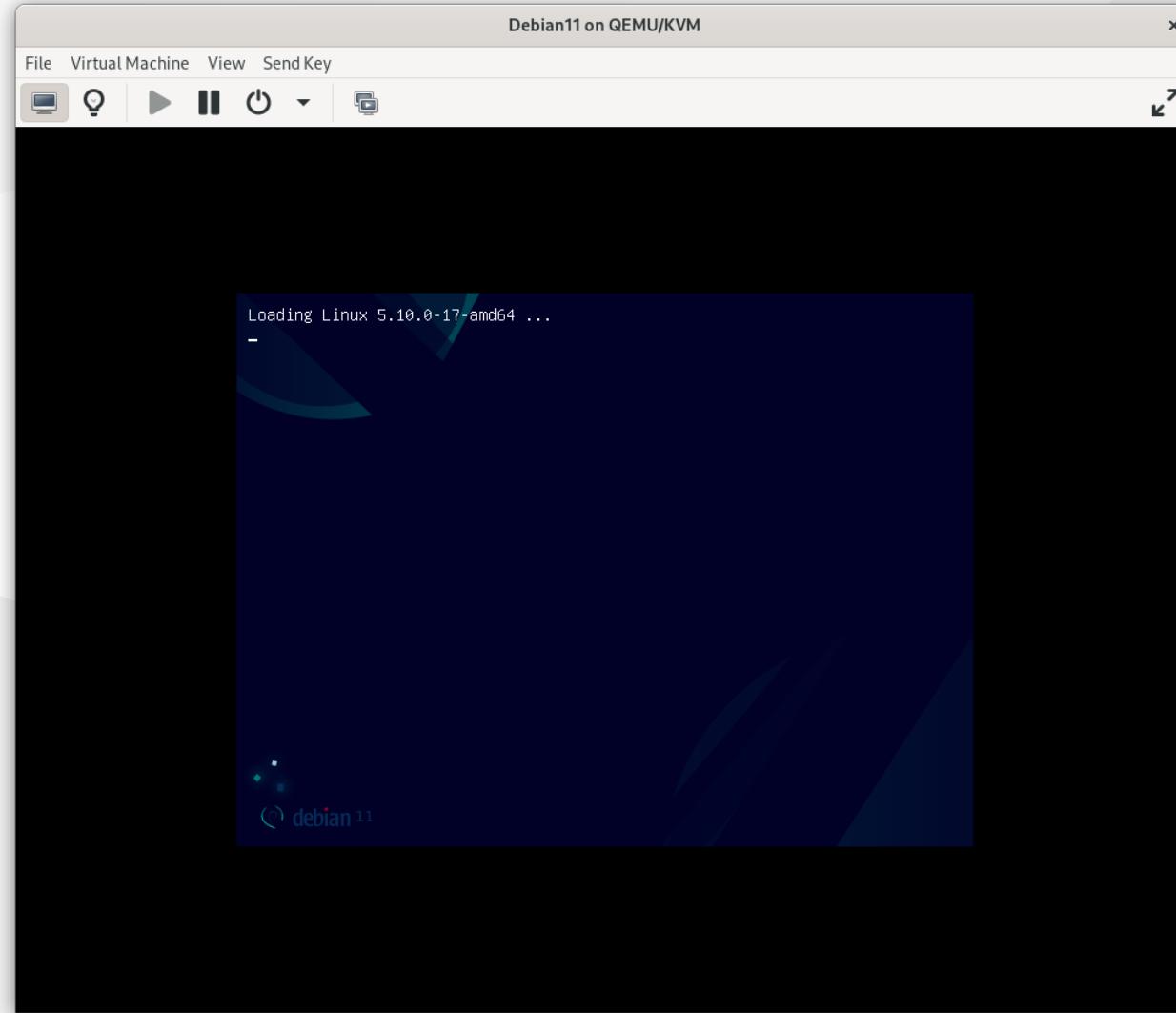


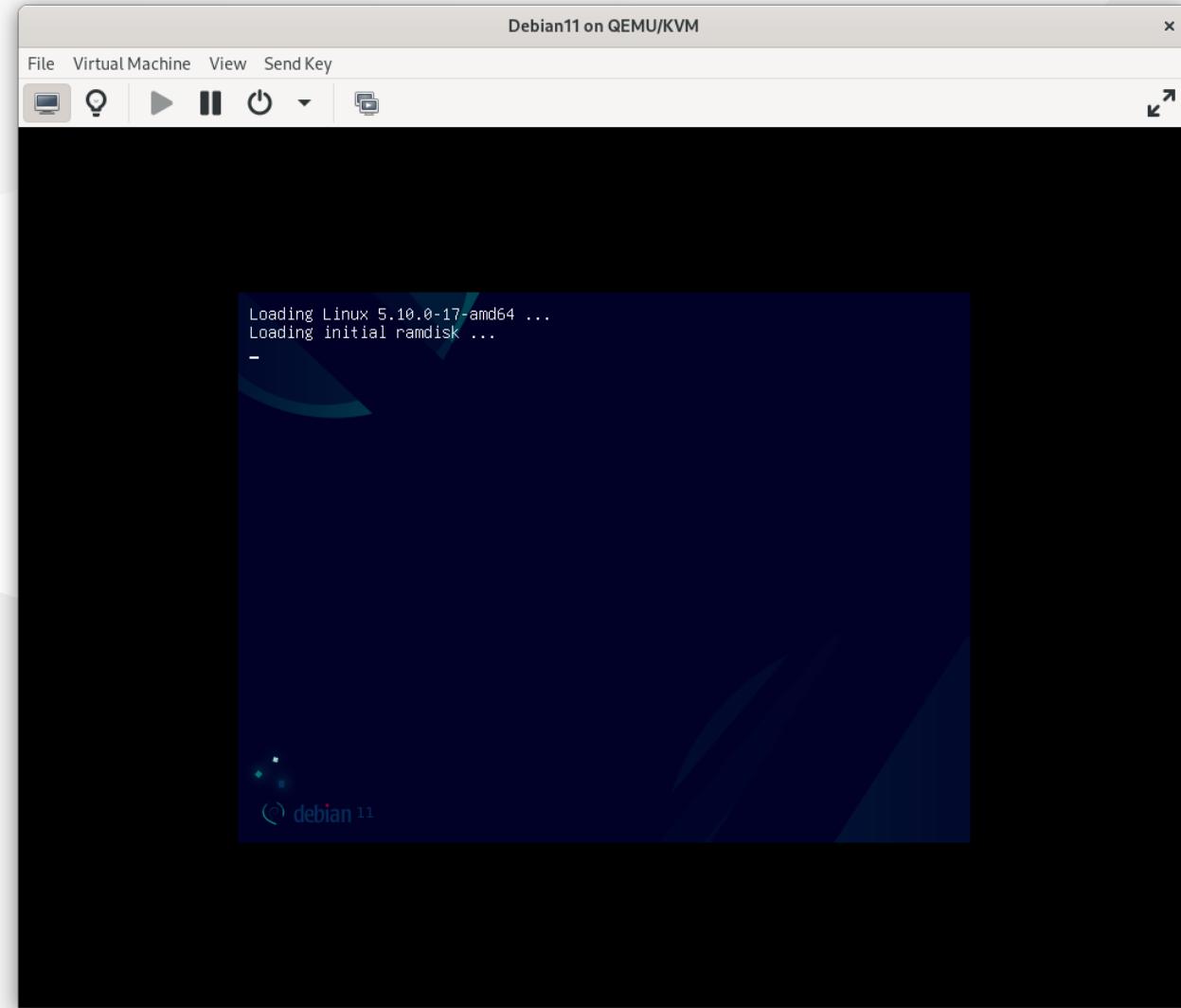


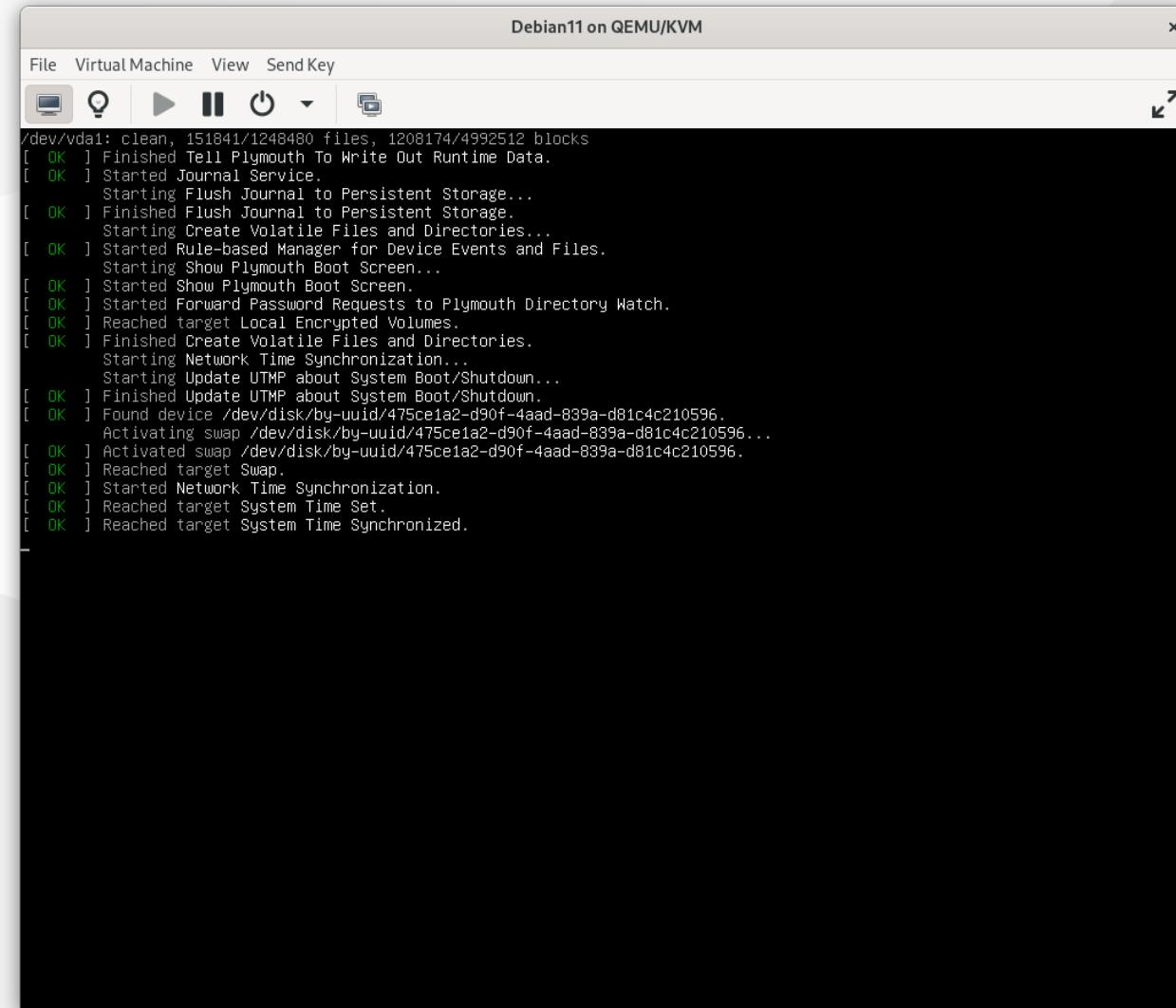


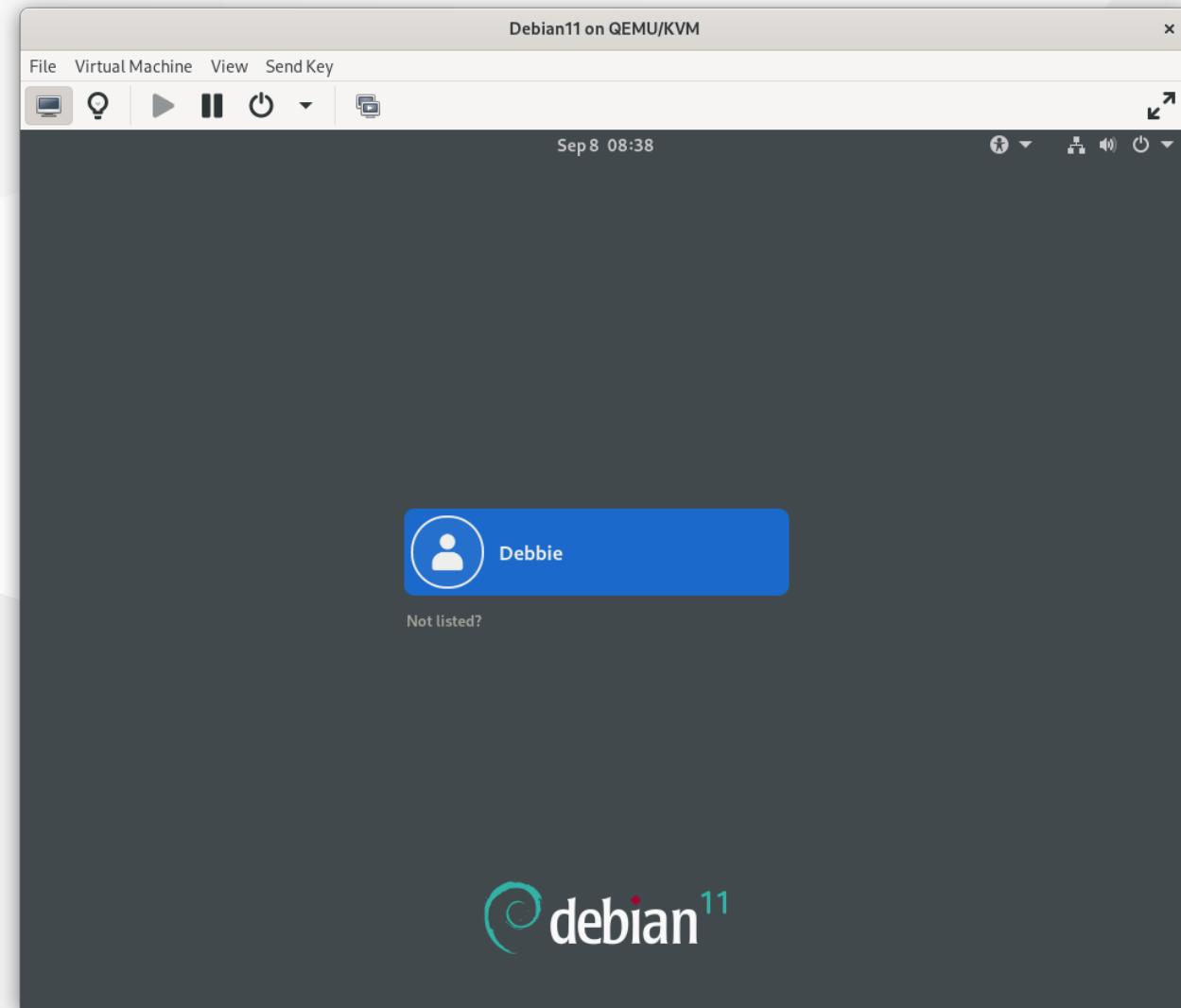


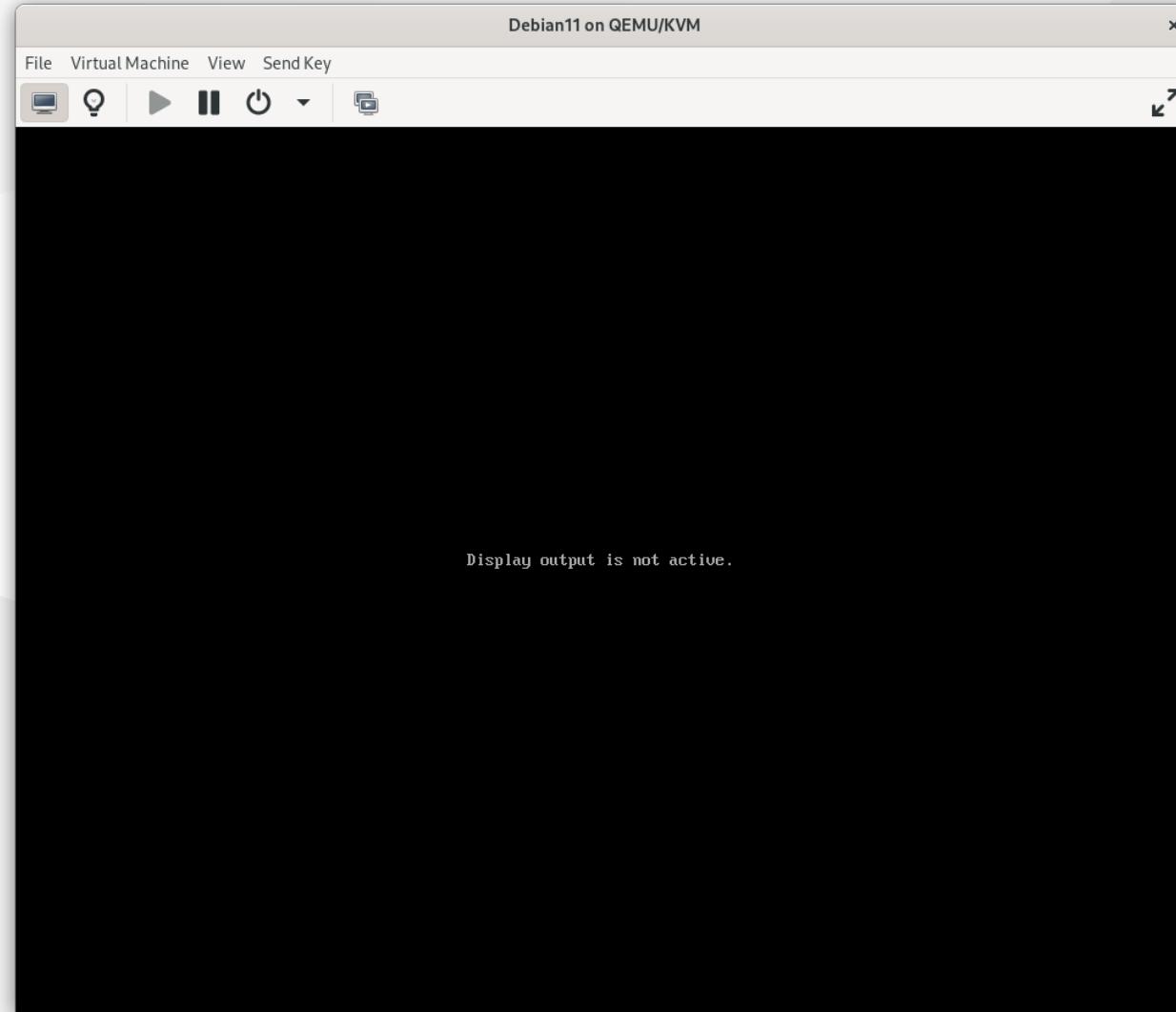


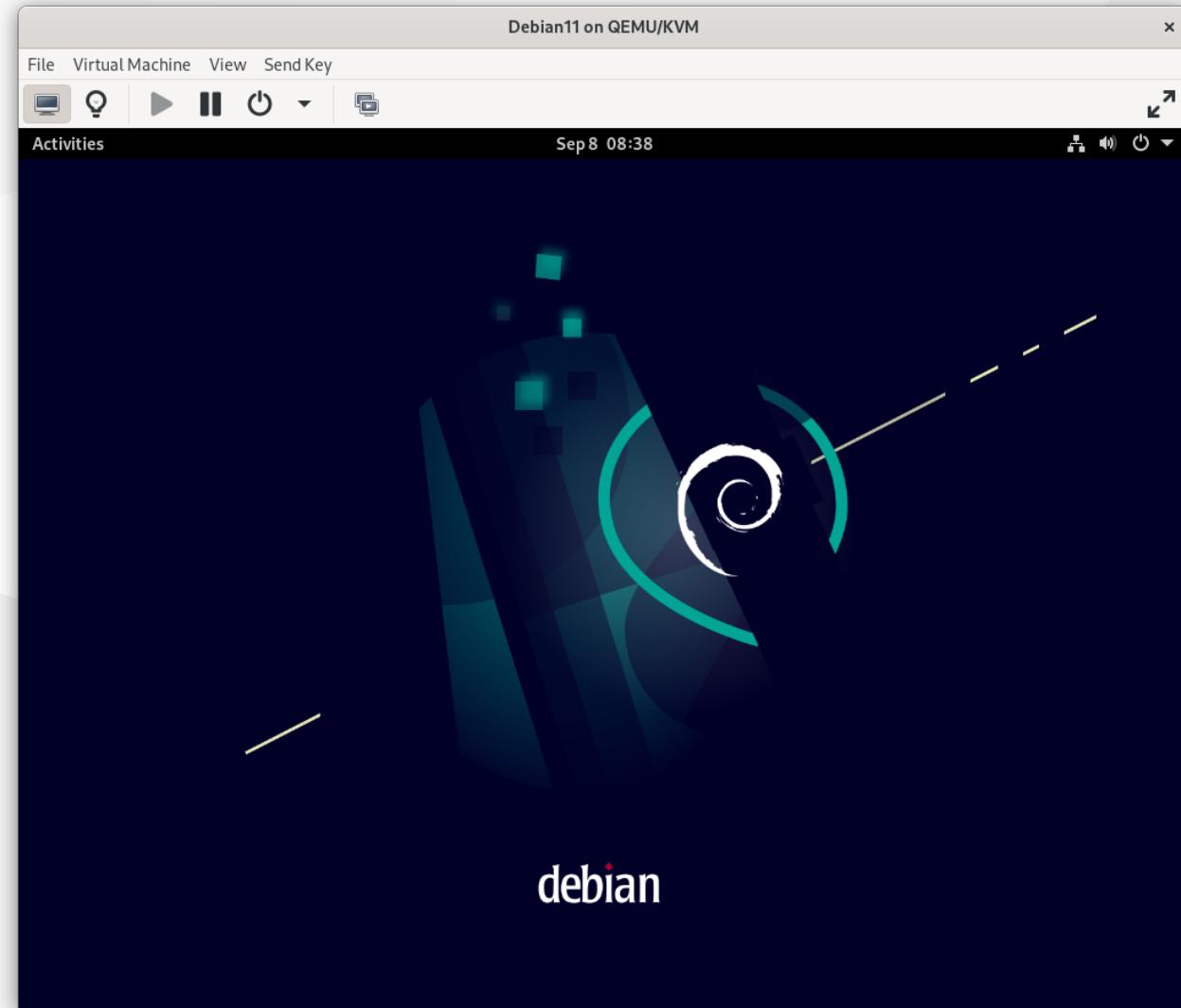


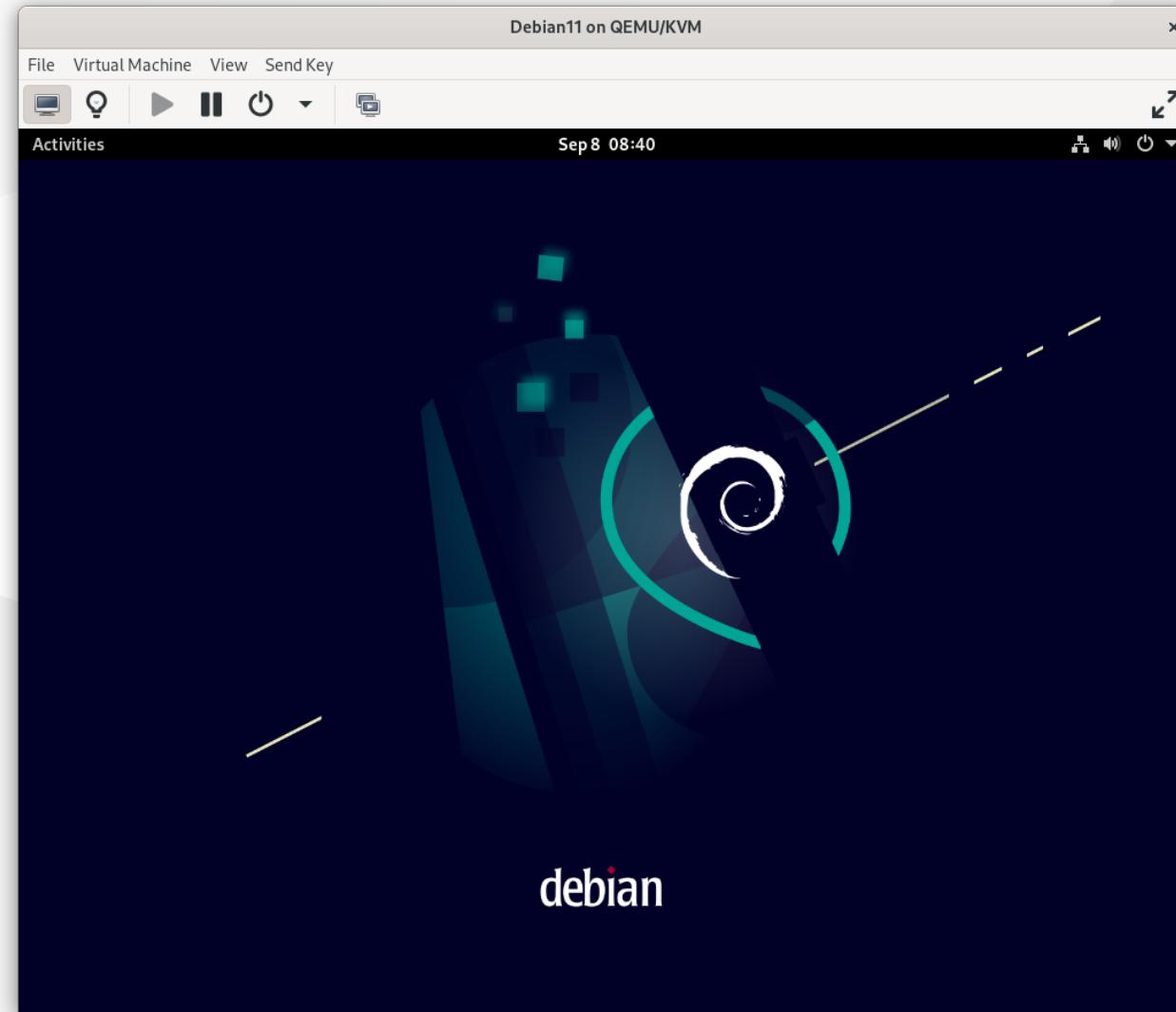


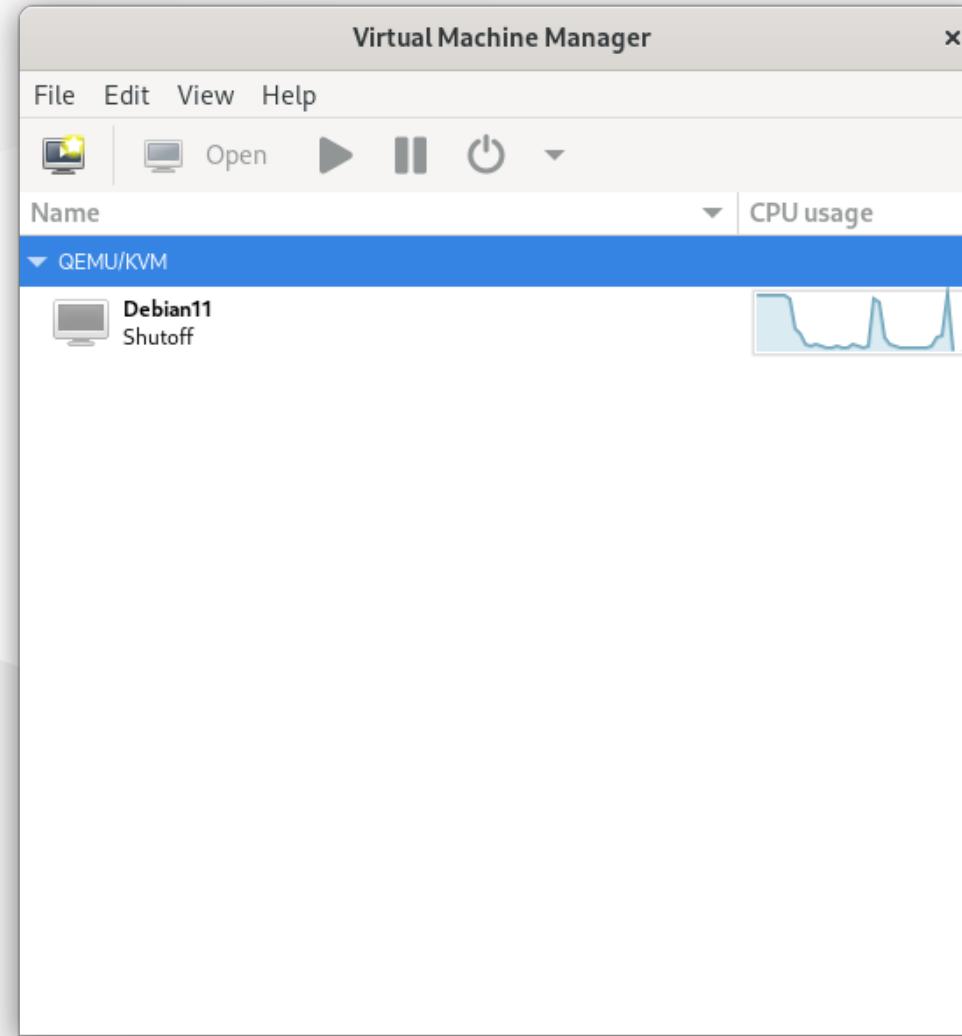


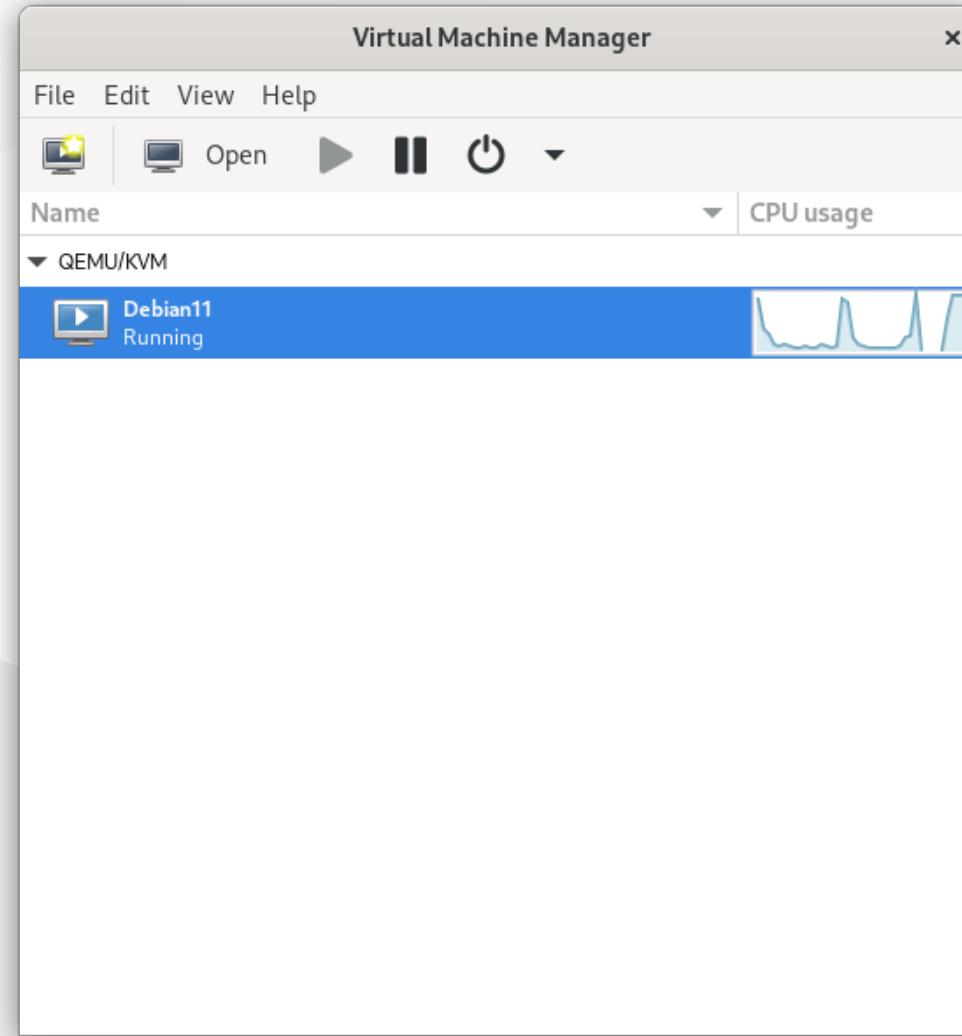












Bonus: Unattended Install

- Preseeding (Debian-based Linux Distributions) or Kickstart (Red Hat-based Linux Distributions) provides a way to set answers to questions asked during the installation process, without having to manually enter the answers while the installation is running.

Bonus: Assign Host USB Device

https://www.linux-kvm.org/page/USB_Host_Device_Assigned_to_Guest



End