UTM x86 Emulation Guide

Creating and Loading Virtual Machines on ARM-Based Macs (M1 Chipset and Later)

Author - Alex Lewtschuk

Introduction

As Apple has now switched to ARM-based chipsets in their new Mac releases the setup and running of virtual machines has become significantly more tedious. Although there is now a VirtualBox beta (at the time of writing) that will run on M1 chips and up, you are unable to boot any form of x86-based operating system using VirtualBox so far. This is likely because the ARM chipset and x86 chipset have incompatible instruction sets. In order to run x86-based operating systems you need to instead rely on emulation rather than virtualization. The purpose of this guide is to walk you through the process, from start to finish, of installing and configuring all necessary software to allow for the running of the provided VirtualBox .ova file natively on your ARM-based Mac. Due to the fact that this is a workaround, it is a multi-step process that will need multiple software installs as it is not possible to install and run .ova files out of the box on UTM.

NOTE: This guide assumes you are starting from scratch with none of the needed software currently installed on your system. If you have already installed one of the needed software packages skip that step and go on to the next step.

Step One - Installing Homebrew and Xcode Command Line Tools

Homebrew is a package manager that can be installed on Mac systems that allows you to install software through the command line, much like you would in a standard Linux system. To install Homebrew follow these steps:

Brew Install

Homebrew provides an installation script that you can run using a single command line command (you can check to make sure that it has not changed here). Install Homebrew using this command.

```
$ /bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

The install script will ask you to enter your Mac user password. Enter it to continue the installation process.

If you do not have Xcode Command Line Tools already installed on your system you will see a message stating, “The XCode Command Line Tools will be installed.” When prompted, press return to continue the installation.

You will see diagnostic and progress messages during this portion, and the process will take a few minutes to complete.

Update the $PATH

On M1 Macs, Homebrew installs itself into the `/opt/homebrew` folder. It’s important to note that this folder is not part of the default `$PATH` environment.

Add Homebrew to your PATH in ~/.zprofile:

```
echo 'eval "$(/opt/homebrew/bin/brew shellenv)"' >> ~/.zprofile
eval "$(/opt/homebrew/bin/brew shellenv)"
```

This can also be edited through a text editor, but since you are already in the command line why not just keep it simple.

Verify Homebrew Installation

After you have completed the above steps, or to check if you have previously installed Homebrew, ensure it is installed properly.

```
$ brew doctor
```

You should see:

```
Your system is ready to brew
```

NOTE: If you see `zsh: command not found: brew`, make sure you’ve created a `~/.zprofile` file as described above and restart your terminal. Congrats! You now have homebrew installed.

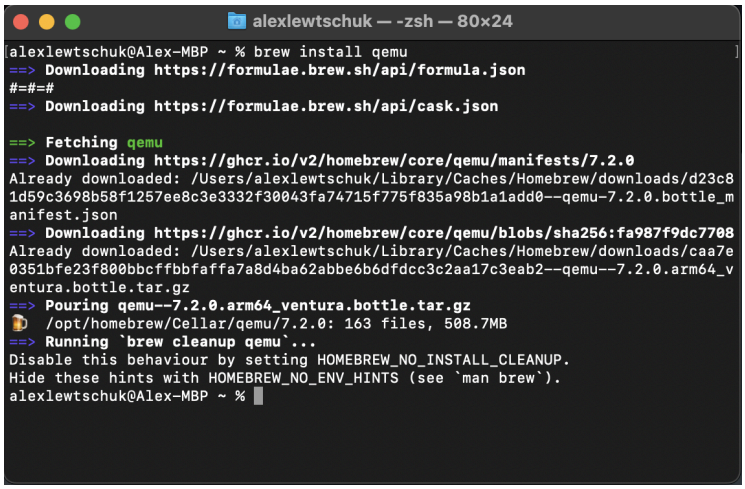
Step Two - Install QEMU

QEMU is a free and open-source machine emulator that can run operating systems and programs for one machine on another. This is what allows us to run the x86-based operating system you are attempting to boot as a virtual machine. UTM is actually a modern front end for QEMU and that is how it creates its virtual machines. This installation of QEMU is only needed for its utils package that you have to use in a later step. After that later step is complete you can feel free to remove the QEMU install as UTM includes it on its own.

QEMU Install

Now you are going to install QEMU using your newly completed Homebrew installation.

```
$ brew install qemu
```



You should see a verbose install message like this in your terminal. It's important to note that QEMU, and thus UTM, cannot open or run .ova files; however, now that QEMU and its util package are installed you can use these to change the .ova file format into a format that UTM can open and run.

Ensure You Have QEMU Installed

Check to make sure that QEMU is installed properly by using:

```
$ qemu-system-x86_64 --version
```

You should see something similar to

```
QEMU emulator version 7.2.0
```

Step Three - Install UTM

UTM is the main application that you will be setting up and running your virtual machine on. This is probably the easiest step. If you install it through the Mac App Store, the program will cost you $9.99; however, if you download it directly from their site, here, you can get it for free.

Step Four - Making the .ova File UTM Compatible

As stated before, UTM cannot handle .ova files by default so you need to convert the file into a type that UTM knows how to handle.

Converting OVA to QCOW2

In order to boot the virtual machine in UTM you must convert the .ova file to a .qcow2 file. Thankfully, this is a fairly simple process as an .ova file is essentially an archive format that you can change around pretty easily.

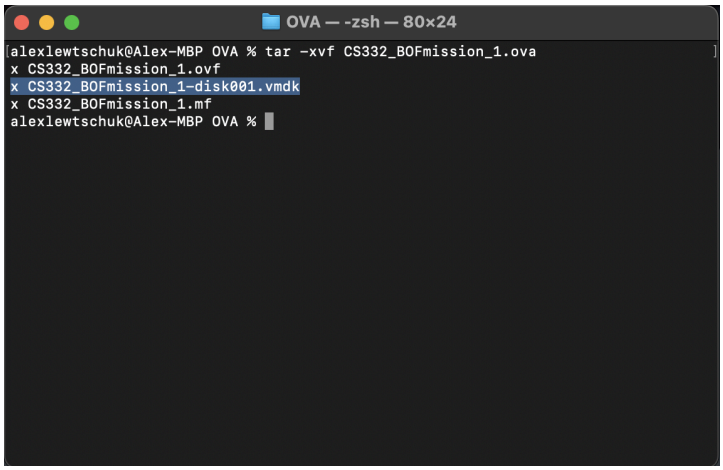
Download the .ova File and Create a Directory For It

Download the .ova file you need to convert. Make a new directory and move it from your `/Downloads` folder to the new directory, for example, `/Desktop/OVA`. This will make it easier when you unzip the file.

Extract Disk Image From the .ova File

Once you have navigated to the directory that contains the .ova file you need to extract the disk image. This is done with a simple `tar` command.

```
$ tar -xvf file.ova
```



The extracted .vdmk is the disk image file. This is the file you are going to be focusing on manipulating. (This can also produce a .vdi file, but in this case, you do not have to worry about that file type.) You don't need to be concerned about the other extra files in the directory for our purposes.

Convert the Disk Image to QCOW2

The extracted image now needs to be converted to the compatible QCOW2 file type. To do this you will be using the QEMU utils that you installed earlier.

To convert the VMDK image to QCOW2 format:

```
$ qemu-img convert -O qcow2 file.vmdk fileout.qcow2
```

This will take a moment so be patient. If you now look in the directory using either `ls` or your file browser you should now see the `fileout.qcow2` file.

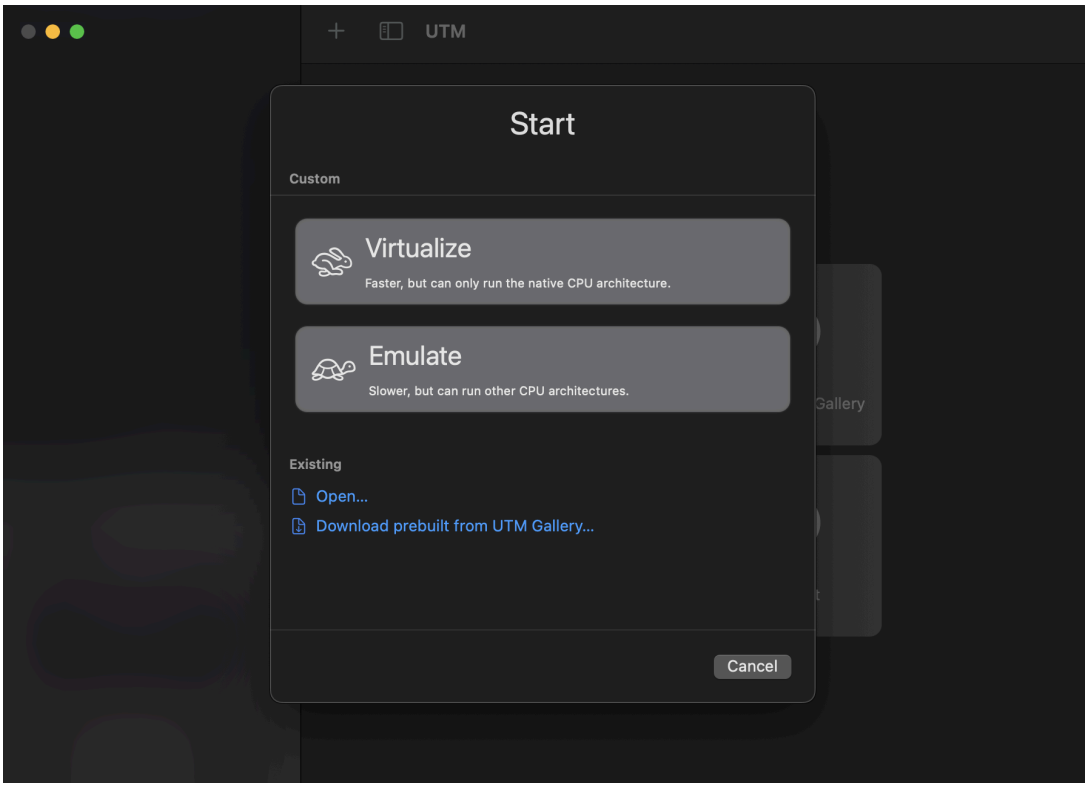
NOTE: See Final Note 6 at the end of this document to remove the QEMU install.

Step Five - Setting Up and Booting the Virtual Machine

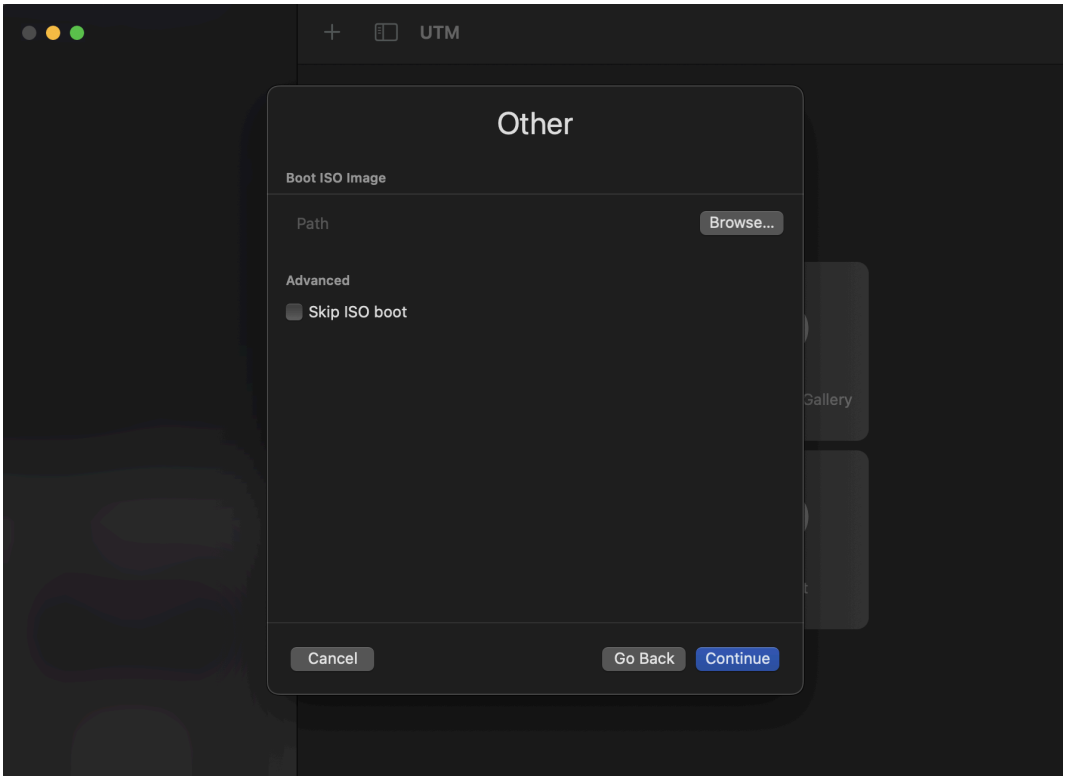
This is the last step in order to get the VM to boot. Usually you could create a VM in UTM similarly to the way you would in VirtualBox, but since this is using a premade image and not booting from an .iso you need to make a few special adjustments. There are going to be a lot of screenshots for this section, so brace yourself.

Create an Empty VM

Open up the UTM app and click “Create a New Virtual Machine.” You should see the following screen.



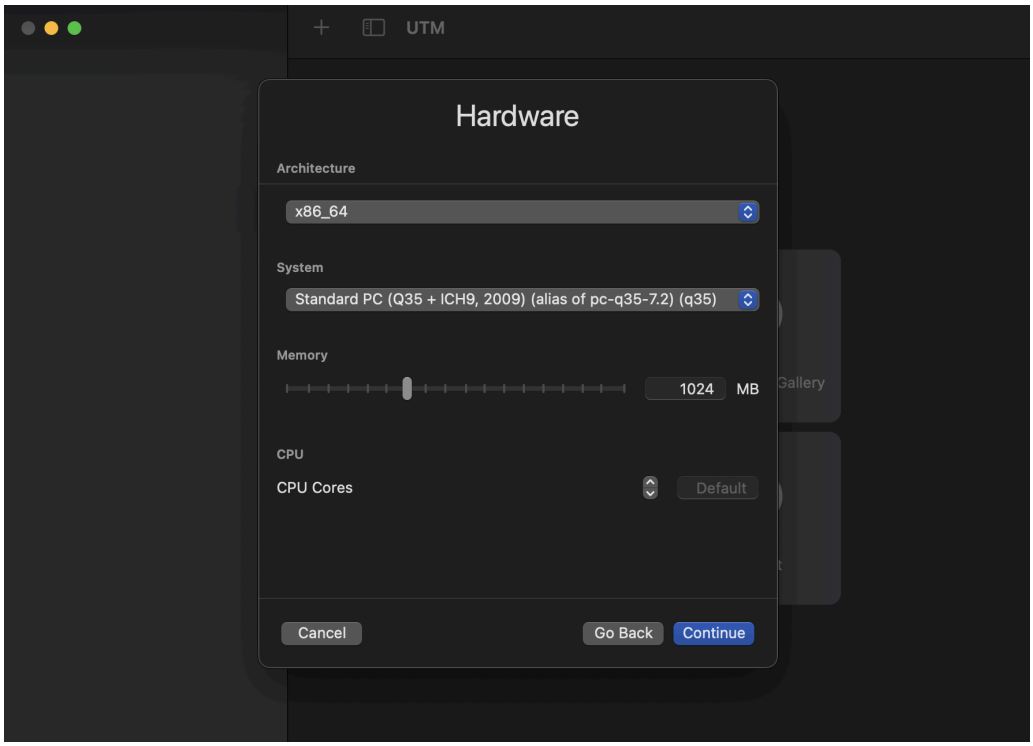
Click “Emulate” and then select “Other.” You should see the following screen.



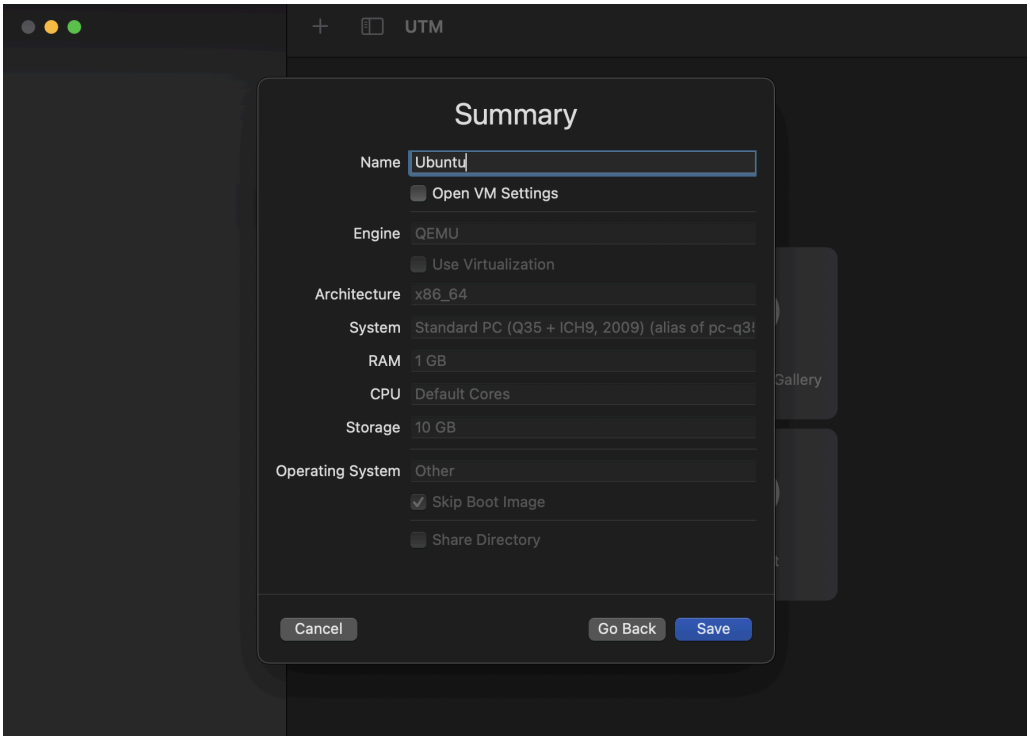
Check the box for “Skip ISO boot” and click the continue button.

On the hardware pane, leave the Architecture as “x86_64” and keep the System option as “Standard PC (Q35 + ICH9, 2009) (alias of pc-q35-7.2)(q35).” You can adjust the memory to whatever you want but 1024 MB works fine.

NOTE: See Final Note 1 at the end of this document.



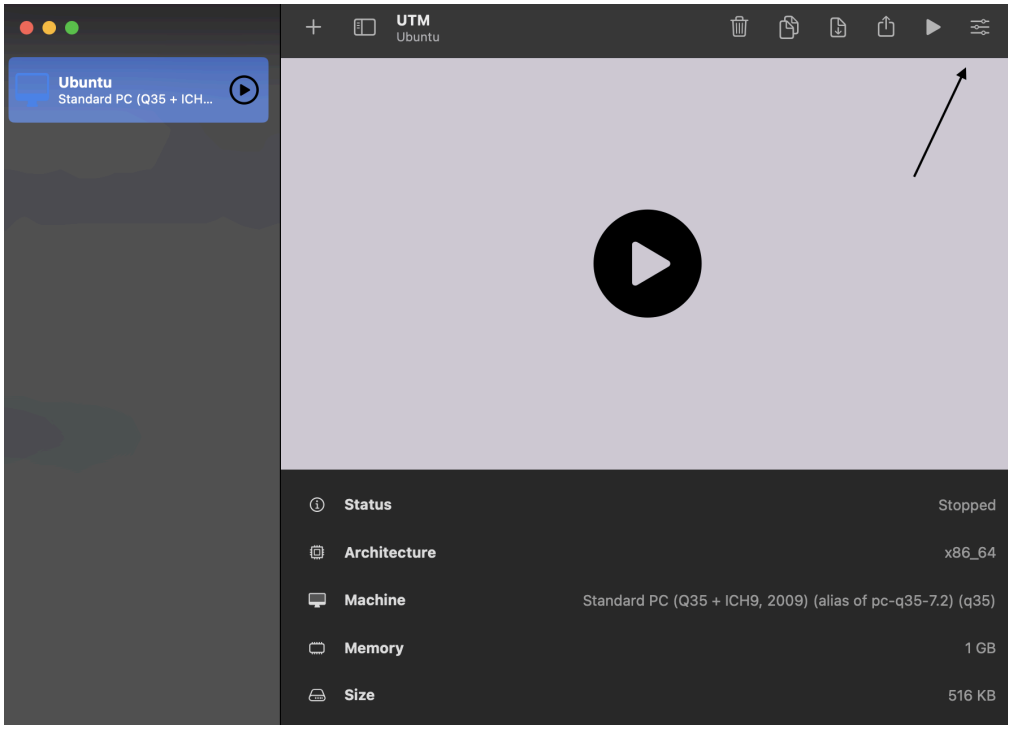
On the storage pane choose whatever value you want. I would use a minimum of 10 GB. Click continue and leave the shared directory pane as-is for now. Click continue again and you will reach the summary pane.



Name the VM whatever you want and click save.

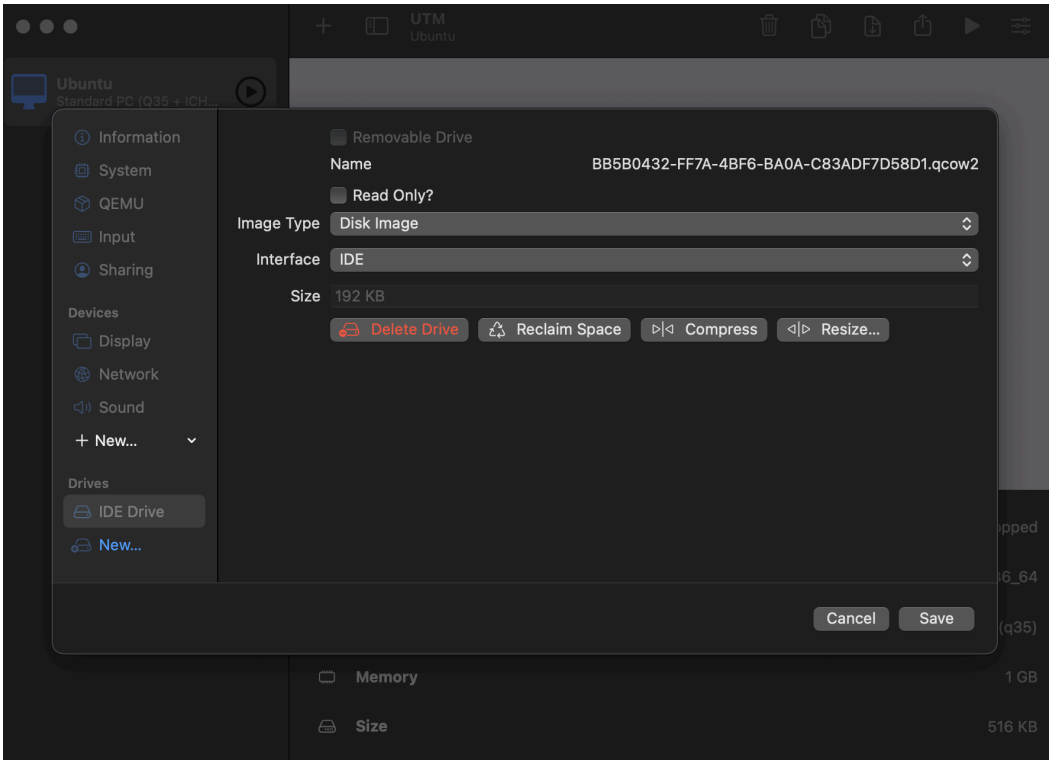
Import the .QCOW2 File

You should now see this screen.

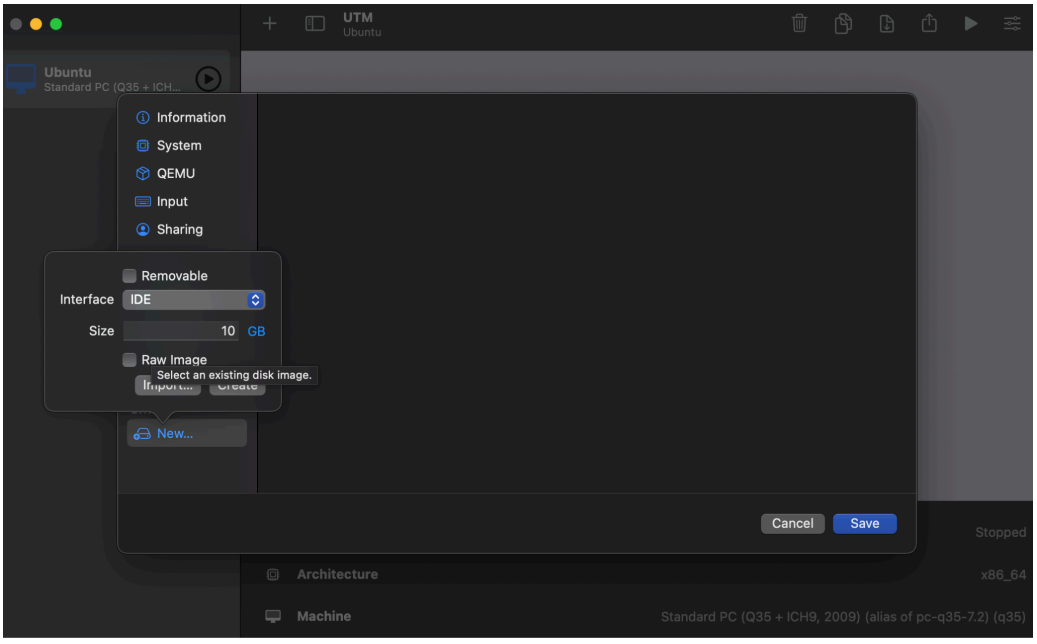


Click on the “Edit selected VM” button in the top right corner.

Navigate to the “IDE Drive” on the sidebar. Go ahead and delete this drive.

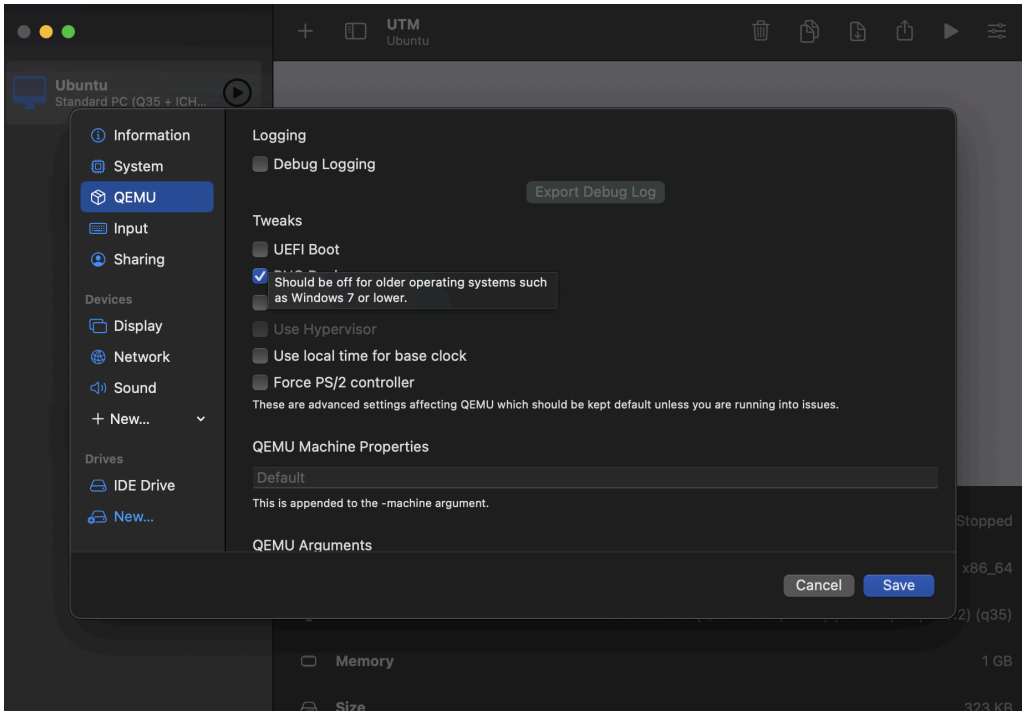


Then click “New Drive” and select import.



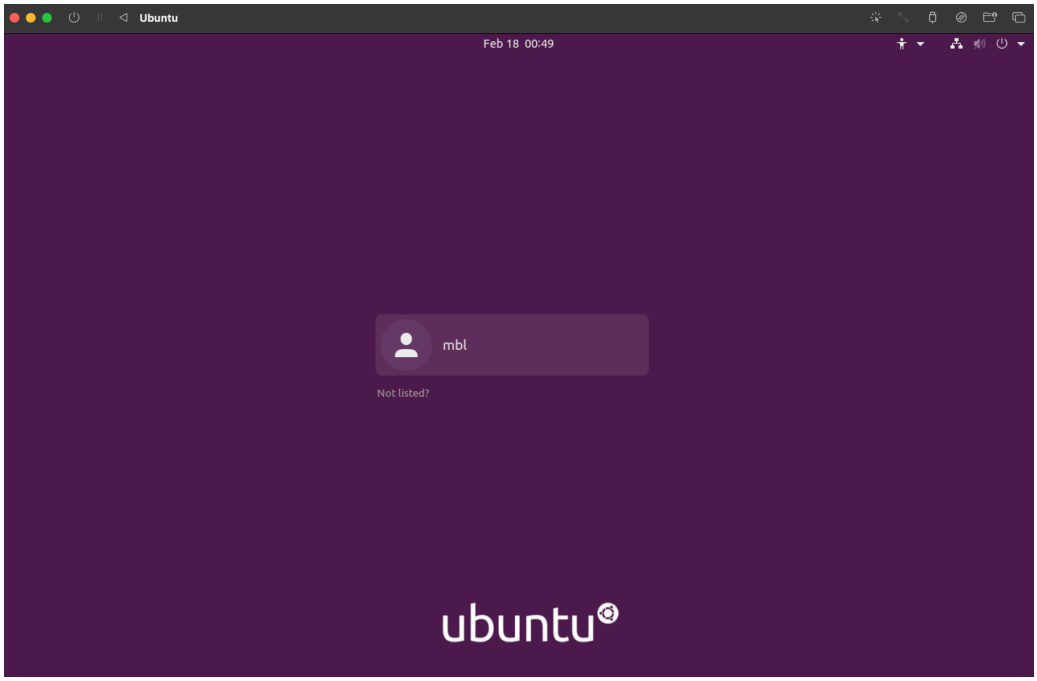
From there navigate to the directory that contains the .qcow2 file, select that file, and click open and save. This should return you back to the main settings menu, and the image file is now loaded.

Go to the “QEMU” settings in the side tab and disable “UEFI boot” as shown below and click save.



Boot Into the Guest OS

You should now be back on the main UTM screen. From here you can boot into the new VM. You should see the VM flash a quick screen that says it's booting from Hard Disk before the screen goes black. It may take a few moments to fully boot into the system. When you get in you should see the distro login screen.



Congrats, you have now set up your new VM!

Final Notes

1. Since this is an emulation rather than a virtualization the performance will be slower. This can be mitigated a bit by increasing the available memory the VM can use.
2. Information on converting file types was originally found in a blog post here.
3. Long-term stability after the file type change at the time of writing is unknown, but there have been no crashes or data loss in tests so far.
4. **Be sure to close the VM by first powering off within the guest operating system and then closing the UTM app to avoid corrupting the VM.**
5. YouTube tutorial for the import of the VM in UTM using the .qcow2 file can be found here.
6. To remove QEMU run the following command:

```
2. $ brew remove qemu
```