

Minecraft Forge Server auf Debian 12 GNU/Linux

Nikola Mihaylov

August 31, 2024

1 Overview

This document details the installation process of a Minecraft Forge server on a Debian 12 base which operates under Hyper-V in RemoteLabs.

Please note that this is for private use only; this document is not officially associated with GFN GmbH and the instructions (specifically the Java segment) should be approached carefully in a production environment.

2 Prerequisites

The prerequisites for this project are as follows:

- An instance of Hyper-V
- Debian 12 base image (netinst)
- JDK package (preferably OpenJDK, but this tutorial will use the Oracle JDK 22 .deb package)
- Forge .jar file

3 Creating the VM

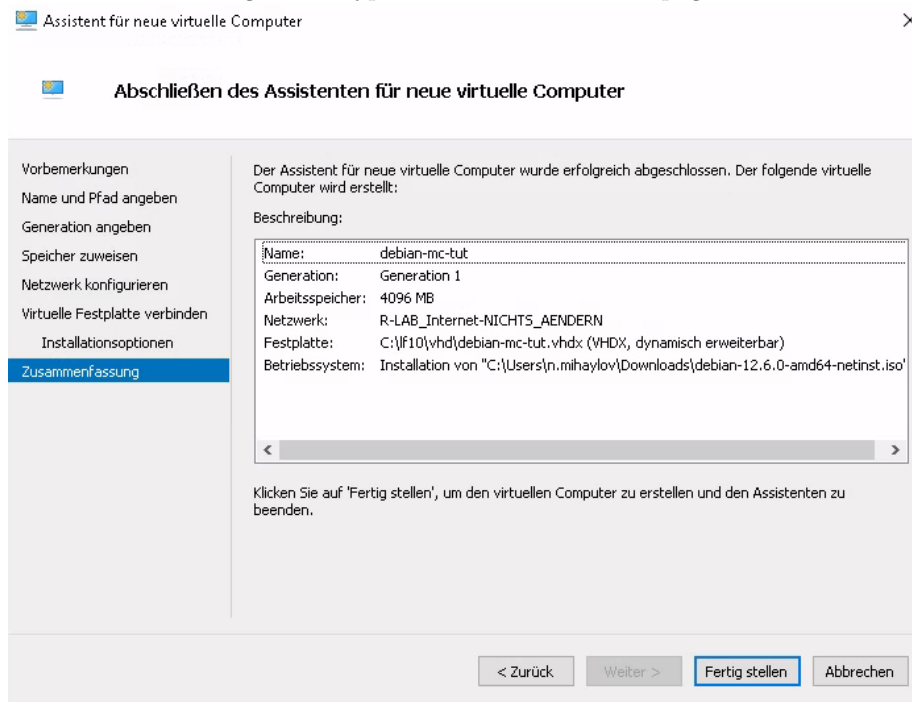
The VM is created with the following parameters:

- Generation 1. (Gen. 2 will work just as well but we're saving ourselves the trouble of dealing with Secure Boot under GNU/Linux)
- 4096MB of RAM
- 2 virtual processors
- 64GB dynamically-sized virtual hard disk drive in the VHDX format

- Network switch "R-LAB-Internet" for internet connection (required for the installation process)

At the end, you should be left with the following:

Figure 1: Hyper-V assistant overview page



4 Acquiring the newest Debian 12 image

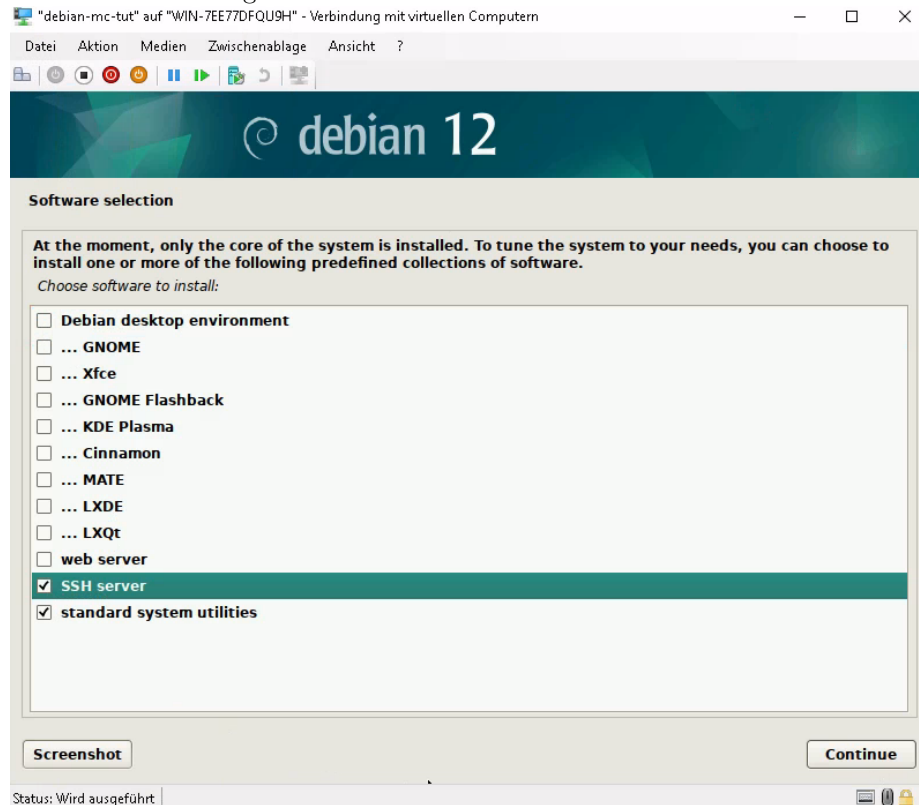
The newest version of the Debian 12 image can be acquired from the official website (click here for a link to the download page). You should note that we're using the netinst for this tutorial which requires an internet connection to download any additional packages, but you can also use a regular complete image for offline installation.

5 Installation process

The GUI-based installation should be relatively straight-forward - you can use the defaults and modify the hostname to your liking (here we're just going to use "debian-mc"). Debian's defaults will take of disk partitioning on our new VHDX. Make sure to select your appropriate time zone, locale and mirror region!

Some things to note for the installation process is to **not** install a desktop environment; we are only going to require an SSH connection to interface with our server as we want to avoid the overhead of any graphical environment, therefore you should only select "SSH server" and "standard system utilities" (refer to Figure 2. for a visual example).

Figure 2: Our Debian 12 software selection

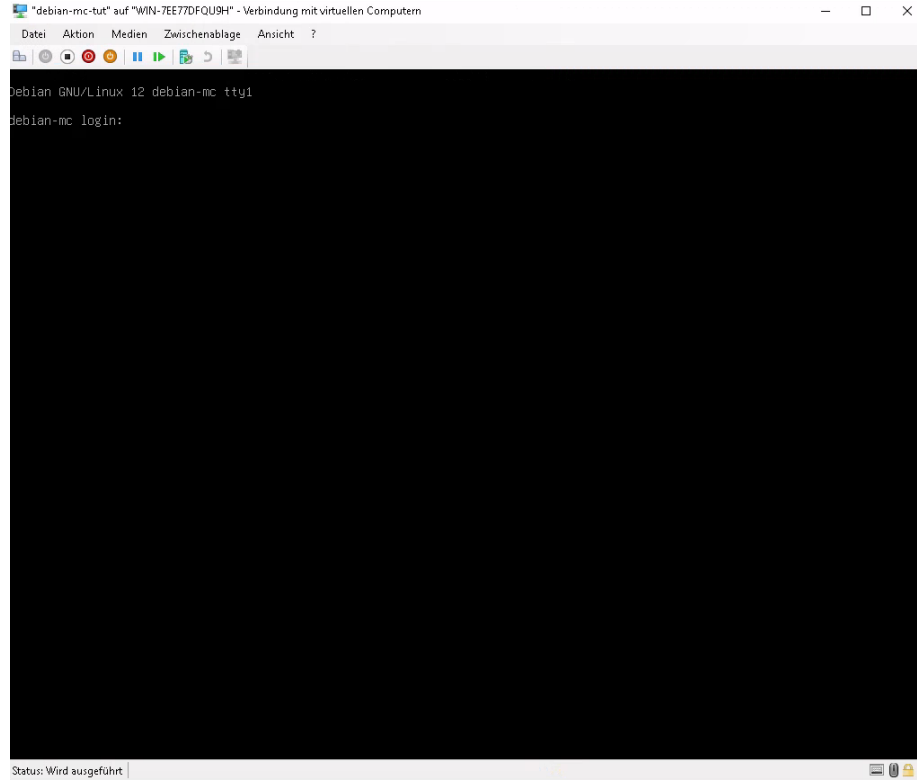


The reason why we're looking into having SSH out of the box is because it will make copying and pasting text much easier, and you will most likely need to do that later during the tutorial. Having a user with sudo privileges will be a prerequisite for SSH access, and now that you've made a user account during the installation all that's left in this regard is to add them to the sudo group post-install.

Once that is done, make sure to install the GRUB boot loader to your primary drive (/dev/sda), otherwise you won't be able to boot into your installation afterwards.

Now you should have a functioning Debian 12 installation with a TTY login prompt.

Figure 3: The Debian 12 TTY prompt



6 Initial post-installation configuration

Before we begin with anything Minecraft-related, there are a couple of post-installation things to configure, with our first and most important one being giving our newly-created user sudo permissions. To do this, we log in as root:

```
debian-mc login: root
Password: {your-root-password-here}
```

Now you should be logged in as root and facing the prompt.

First order of business is to install the sudo package if it is not already installed. To do this, we are going to type:

```
apt install sudo
```

The apt command is used to manage packages on the Debian system. With the "install" flag we can specify a package name to install, in this case the sudo package. If sudo wasn't available on your system already, your package manager will ask you if you want to install the package. Confirm it with enter. You can

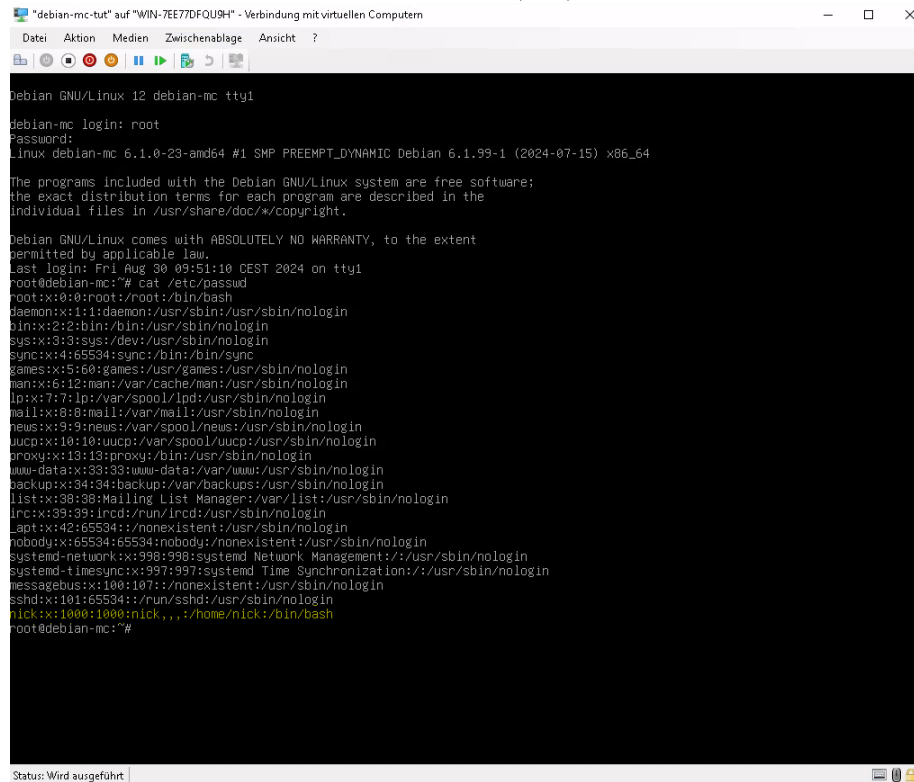
type `sudo` afterwards to confirm that the package has been installed with the output being the help for the command.

Now that that's done, we can continue with our preliminary configuration. We can list all users on the system with the following command:

```
cat /etc/passwd
```

Your newly-created user from the graphical installer will most likely be at the very bottom. Here you can see the local user entries per-line and their associated User ID (UID), Group ID (GID), home directory, default shell and more.

Figure 4: Output of the `/etc/passwd` file



```
Debian GNU/Linux 12 debian-mc tty1
debian-mc login: root
Password:
Linux debian-mc 6.1.0-23-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.99-1 (2024-07-15) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Aug 30 09:51:10 CEST 2024 on tty1
root@debian-mc:~# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mail List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
lapt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
systemd-timesync:x:997:997:systemd Time Synchronization:/:/usr/sbin/nologin
messagebus:x:100:101::/nonexistent:/usr/sbin/nologin
sshd:x:101:65534:/:run/sshd:/usr/sbin/nologin
nick:x:1000:1000:nick,,:/home/nick:/bin/bash
root@debian-mc:~#
```

Now we are going to add our user to the `sudo` group. The `sudo` group is a special group which has administrative privileges using the `sudo` command. To do this, we are going to type the following:

```
usermod -aG sudo {your-username-here}
```

The `usermod` command allows us to modify the user entries we saw earlier in the `/etc/passwd` file. The `-a` option is for "append", i.e. add a user to a group.

The `-G` option is for specifying a group by its name (in this case `sudo`). In a lot of GNU/Linux tools you are able to combine options together like `"-aG"` instead of `"-a -G"`. If you typed the command and received no output, that means it should be all good.

You can now switch back your user with the `su` (switch user) command to test your newly-given `sudo` privileges:

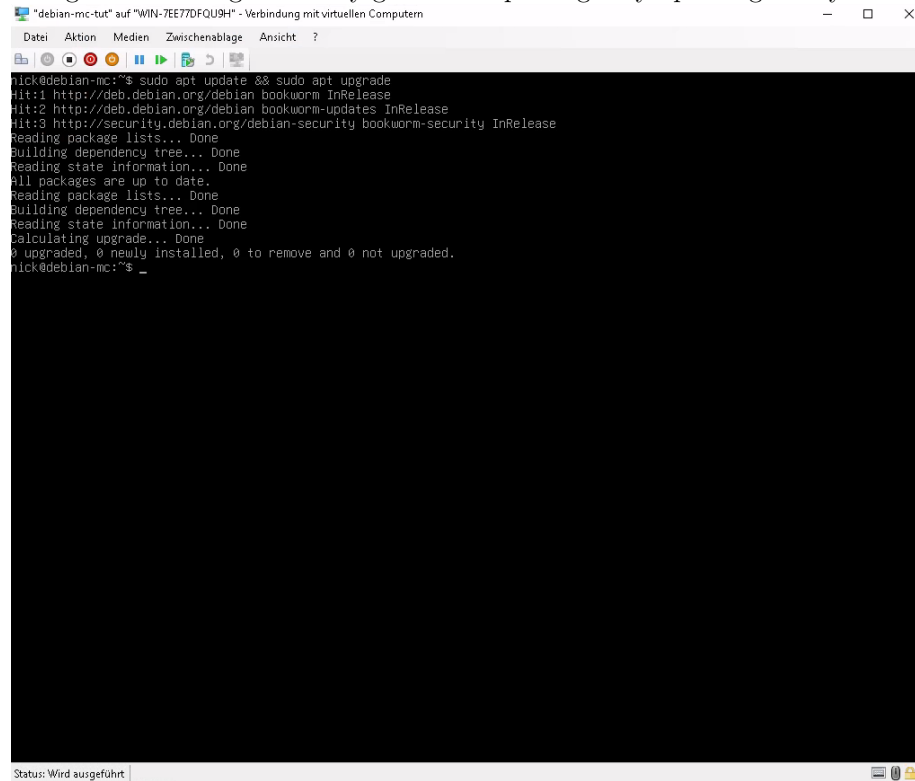
```
su {your-username-here}
```

Now try updating your system using `sudo`:

```
sudo apt update && sudo apt upgrade
```

If everything went well, you should be able to synchronize your system with the apt repository and install any available updates on your computer!

Figure 5: Testing the newly-given `sudo` privileges by updating our system



```
nick@debian-mc:~$ sudo apt update && sudo apt upgrade
Hit:1 http://deb.debian.org/debian bookworm InRelease
Hit:2 http://deb.debian.org/debian bookworm-updates InRelease
Hit:3 http://security.debian.org/debian-security bookworm-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
nick@debian-mc:~$
```

7 Establishing an SSH connection

If you followed the installation section of this tutorial, you most likely already have OpenSSH installed which is required for this section. To check it, you can

issue the following command:

```
ssh
```

You will now get the command help output if the OpenSSH package is installed on your system. If not, install it with the following command:

```
sudo apt install openssh-server
```

Once that is done, we can hop back to our host system and open a command prompt (for example PowerShell). To establish a connection we must follow a specific syntax:

```
ssh {username}@{ipv4-address}
```

We connect as a user on our SSH server at the given IPv4 address of the server. If you don't know your IPv4, you can issue the following command to get the IPv4 of your ethernet adapter/NIC:

```
ip addr | grep "eth0"
```

The "ip addr" command outputs information on the network interfaces of the system. Because we're working with a VM, it is highly likely we're using a virtual ethernet adapter, hence the default "eth0" we're trying to get the information from. If you are to normally run "ip addr", it will output all interfaces which shouldn't be a problem on our fresh install, but when dealing with commands that output lots of information of which we only require a small amount of, we can simultaneously utilize the "grep" command. This utility scans the output of your previous command (in our case ip addr) and only outputs the line which is given as an argument to grep (this being "eth0").

The pipe between these two commands is the most important part; this should be your introduction to the concept of input/output redirection in Linux, with the pipe taking the output of our command to the left of it and providing it as an input to the command on the right. Vaguely said, the flow of this entire command is now "ip addr" -> | -> "grep 'eth0'".

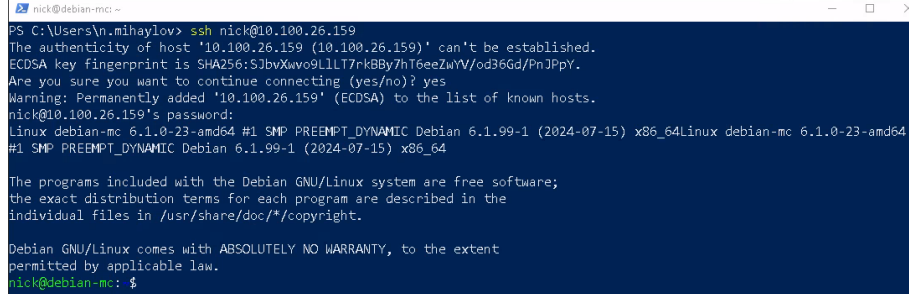
Your resulting output should be similar to this:

Figure 6: The output of the ip addr | grep "eth0" command

```
nick@debian-mc:~$ ip addr | grep "eth0"
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    inet 10.100.26.159/24 brd 10.100.26.255 scope global dynamic eth0
nick@debian-mc:~$
```

With our IP address in hand, we can now use our terminal emulator of choice and use the syntax from earlier to connect to our machine from our host:

Figure 7: Establishing an SSH connection to the VM



```
PS C:\Users\mihaylov> ssh nick@10.100.26.159
The authenticity of host '10.100.26.159 (10.100.26.159)' can't be established.
ECDSA key fingerprint is SHA256:5JbvXwvo9LlLT7rk8By7hT6eeZwVv/od36Gd/PnJPy.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.100.26.159' (ECDSA) to the list of known hosts.
nick@10.100.26.159's password:
Linux debian-mc 6.1.0-23-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.99-1 (2024-07-15) x86_64linux debian-mc 6.1.0-23-amd64
#1 SMP PREEMPT_DYNAMIC Debian 6.1.99-1 (2024-07-15) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
nick@debian-mc: $
```

Success! You should now have an SSH connection to your Debian 12 virtual machine. Of course, it is important to note that the machines should be on the same network, but you can take this as far as using a terminal emulator on your GFN-issued laptop instead of RemoteLabs to connect to the VM via SSH. With this in hand you are now able to copy and paste lengthy commands easily into your VM without requiring to type them out manually.

8 Preparing the Minecraft Forge server installation

In order to host our instance of Forge for the Minecraft server we require an installation of the Java Development Kit on our system.

Here a problem arises: the most recent version of Forge requires version 21 of the JDK but this version of OpenJDK is still in the testing repository as of the time of writing this document, and the Oracle JDK is licensed under their own license and isn't available in the repositories either. This is where I must make it clear that this is not to be used in a production environment just because of how messy the current Java situation on Debian is.

With all that being said, this document will continue with the Oracle JDK 21 .deb file which can be found (here). Copy the link to the .deb file from this page and keep it in your clipboard.

Back to the established SSH connection to our Debian 12, we need to create a folder to store our downloaded file. For this we're going to first navigate to our user's home folder by just typing "cd" (you can verify which directory you're currently in with the "pwd" command) and then use the "mkdir" command (make directory) within our home directory to create a new one like so:

```
mkdir Downloads
```

Now if we do an "ls" (list) we should see that the new directory with the name "Downloads" has been created.

Figure 8: Creating a directory in the user's home

```
nick@debian-mc:~$ cd
nick@debian-mc:~$ pwd
/home/nick
nick@debian-mc:~$ mkdir Downloads
nick@debian-mc:~$ cd Downloads/
nick@debian-mc:~/Downloads$ pwd
/home/nick/Downloads
nick@debian-mc:~/Downloads$ ^
```

At this point we can download the JDK .deb file we currently have in our clipboard using the "wget" tool. The command should be pretty simple now that we have a functional clipboard at hand thanks to SSH:

```
wget https://download.oracle.com/java/22/latest/jdk-22_linux-x64_bin.deb
```

The link above is for the Oracle JDK 22. Depending on when you read this document, the OpenJDK 21 package might already be in the Debian stable repository. As for the rest of the tutorial, we will proceed with the downloaded .deb file from Oracle.

If you perform an "ls" in your Downloads directory you should see the file you just downloaded with "wget":

Figure 9: The downloaded JDK .deb file

```
nick@debian-mc:~/Downloads$ wget https://download.oracle.com/java/22/latest/jdk-22_linux-x64_bin.deb
--2024-08-30 14:10:26-- https://download.oracle.com/java/22/latest/jdk-22_linux-x64_bin.deb
Resolving download.oracle.com (download.oracle.com)... 23.32.100.101
Connecting to download.oracle.com (download.oracle.com)|23.32.100.101|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 167396788 (160M) [text/plain]
Saving to: 'jdk-22_linux-x64_bin.deb'

jdk-22_linux-x64_bin.deb 100%[====>] 159.64M 2.30MB/s in 69s
jdk-22_linux-x64_bin.deb 4%[==>] 6.96M 2.50MB/s
jdk-22_linux-x64_bin.deb 4%[==>] 7.02M 2.50MB/s
jdk-22_linux-x64_bin.deb 4%[==>] 7.09M 2.49MB/s

2024-08-30 14:11:36 (2.30 MB/s) - 'jdk-22_linux-x64_bin.deb' saved [167396788/167396788]

nick@debian-mc:~/Downloads$ ls
jdk-22_linux-x64_bin.deb
nick@debian-mc:~/Downloads$
```

Files with the .deb extension are not unlike a Windows .exe installer file - they contain the required binaries and data for an application as well as all necessary meta information for the system. As a matter of fact, you can open

this file with an archive viewer and see that it consists of two tarball archives, each for the application itself and its metadata.

To install a .deb file as a system package, we will need the "dpkg" command. "apt" actually uses "dpkg" under the hood to perform the installation of packages, but right now we're skipping "apt" and going directly to "dpkg" to install a file. We're going to use the following command:

```
sudo dpkg -i jdk-22_linux-x64_bin.deb
```

Remember that you can use tab-complete! Start with the first few letters of the file name and use the Tab key to automatically fill in the rest. You will be prompted for your sudo password and installation should begin.

When the installation completes there will be all sorts of output on your terminal. If it looks similar to the following figure, you should be good to go.

Figure 10: Output of the JDK .deb installation

```
nick@debian-mc:~/Downloads$ sudo dpkg -i jdk-22_linux-x64_bin.deb
[sudo] password for nick:
Selecting previously unselected package jdk-22.
(Reading database ... 33102 files and directories currently installed.)
Preparing to unpack jdk-22_linux-x64_bin.deb ...
Unpacking jdk-22 (22.0.2-ga) ...
Setting up jdk-22 (22.0.2-ga) ...
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jar to provide /usr/bin/jar (jar) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jarsigner to provide /usr/bin/jarsigner (jarsigner) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/java to provide /usr/bin/java (java) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/javac to provide /usr/bin/javac (javac) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/javadoc to provide /usr/bin/javadoc (javadoc) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/javap to provide /usr/bin/javap (javap) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jcmd to provide /usr/bin/jcmd (jcmd) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jconsole to provide /usr/bin/jconsole (jconsole) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jdb to provide /usr/bin/jdb (jdb) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jdeprscan to provide /usr/bin/jdeprscan (jdeprscan) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jdeps to provide /usr/bin/jdeps (jdeps) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jfr to provide /usr/bin/jfr (jfr) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jhsdb to provide /usr/bin/jhsdb (jhsdb) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jimage to provide /usr/bin/jimage (jimage) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jinfo to provide /usr/bin/jinfo (jinfo) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jlink to provide /usr/bin/jlink (jlink) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jmap to provide /usr/bin/jmap (jmap) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jmod to provide /usr/bin/jmod (jmod) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jpackage to provide /usr/bin/jpackage (jpackage) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jps to provide /usr/bin/jps (jps) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jrunscript to provide /usr/bin/jrunscript (jrunscript) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jshell to provide /usr/bin/jshell (jshell) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jstack to provide /usr/bin/jstack (jstack) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jstat to provide /usr/bin/jstat (jstat) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jstatd to provide /usr/bin/jstatd (jstatd) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jwebserver to provide /usr/bin/jwebserver (jwebserver) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/keytool to provide /usr/bin/keytool (keytool) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/rmiregistry to provide /usr/bin/rmiregistry (rmiregistry) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/serialver to provide /usr/bin/serialver (serialver) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/java to provide /usr/bin/java (java) in auto mode
update-alternatives: using /usr/lib/jvm/jdk-22.0.2-oracle-x64/bin/jexec to provide /usr/bin/jexec (jexec) in auto mode
nick@debian-mc:~/Downloads$
```

You can verify the installation by running `java --version` which will hopefully display the version of the package you just installed (in this case 22.0.2). You have now successfully installed the required Java JDK for running Forge!

9 Installing Minecraft Forge

We should first create a dedicated directory for our installation of Minecraft and cd into it:

```
sudo mkdir /opt/minecraft
cd /opt/minecraft
```

Now using "wget" we can download the newest recommended version of Forge as of the time of writing this document which you can get from (here). Running "wget" with this link as the argument will download the file in our /opt/minecraft directory:

```
sudo wget https://maven.minecraftforge.net/net/minecraftforge/forge/1.20.6-50.1.0/forge-1.20.6-50.1.0-installer.jar
```

Verify that the file has been downloaded in the current directory with an "ls". We can now proceed with installing the server version of Minecraft Forge with the .jar file using Java:

```
sudo java -jar forge-1.20.6-50.1.0-installer.jar --installServer
```

This process might take a little while. You will know it completed successfully by the "The server installed successfully" message at the end.

The next step is to make an "eula.txt" file which is required by the Minecraft server. We can either "touch" the file to create it with no contents or create it using a text editor like "nano" as we will need to write contents into the file anyway.

"Touching" a file creates an empty file of the same name you provided to the touch command like so:

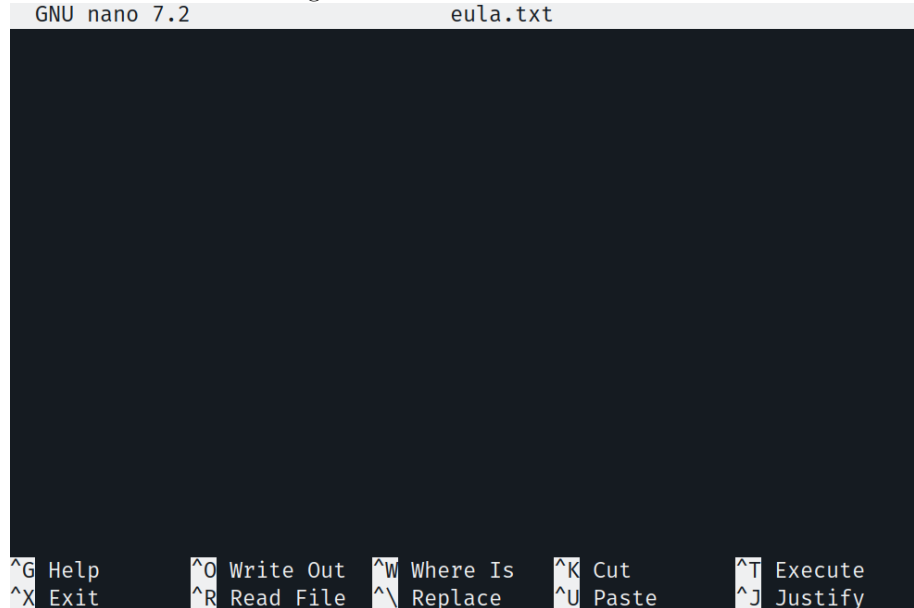
```
sudo touch eula.txt
```

Now when you "ls" the directory you should see the file. If you "cat" it to output its contents, you will see nothing as it is an empty file. To edit it we will use the "nano" text editor:

```
sudo nano eula.txt
```

You will now find yourself in the interface of the nano text editor:

Figure 11: The nano text editor



nano is a relatively straight-forward text editor - you can start typing away to write text in the editor buffer and save it by pressing Ctrl+O to "write it out" and confirm with Enter. Here we only have to add the following line:

```
eula=true
```

Write the buffer out to the file by pressing Ctrl+O and hitting Enter to save under the default file name. If you face an error while saving you have most likely forgotten to start nano as a superuser with the sudo command.

If you're still in the mood for output redirection, we can skip using "nano" and make our Bash shell write this line to our file by using this command:

```
echo "eula=true" | sudo tee -a eula.txt
```

The "echo" command prints whatever comes after it to the shell, but the pipe takes this output and passes it to "tee" with sudo privileges to append it with the -a flag to the eula.txt file. "tee" command is a utility that reads an output of a command, display it on the screen and also write it into an output file (in this case the "eula.txt"). While it is more convenient to add this line to our file using a text editor like nano, the concept of redirection is important when it comes to creating scripts which may be useful for you eventually on your Linux journey.

We can verify the contents of the file with "cat" which should output what we just wrote in the file.

Now let's "touch" an empty "server.properties" file into existence:

```
touch server.properties
```

Let's also modify an existing file in this directory with the name `user_jvm_args.txt`. This should contain the instructions for how the Java Virtual Machine is to be started.

```
sudo nano user_jvm_args.txt
```

All lines starting with a `#` are going to be ignored as they're interpreted as comments; whatever follows afterwards that doesn't have a `#` will be read as a configuration file entry. All we need to add is the following:

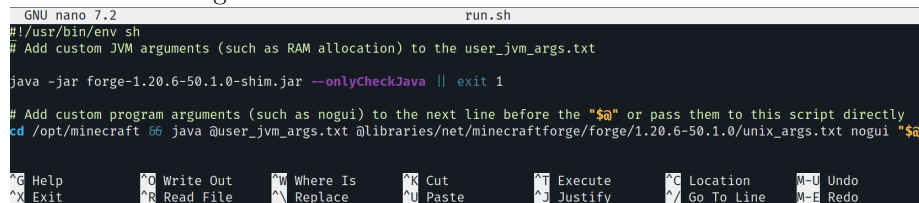
```
-Xms4G
-Xmx4G
```

This allocates a minimum and maximum of 4GB of RAM to the JVM. Note that the first X's are capitalized.

All that's left is to edit the "run.sh" shell script file to modify the very last line by prepending `cd /opt/minecraft &&` at the beginning of the line and adding a `nogui`.

Your file should now look like this:

Figure 12: The contents of the edited run.sh file



```
GNU nano 7.2 run.sh
#!/usr/bin/env sh
# Add custom JVM arguments (such as RAM allocation) to the user_jvm_args.txt
java -jar forge-1.20.6-50.1.0-shim.jar --onlyCheckJava || exit 1
# Add custom program arguments (such as nogui) to the next line before the "$@" or pass them to this script directly
cd /opt/minecraft && java @user_jvm_args.txt @libraries/net/minecraftforge/forge/1.20.6-50.1.0/unix_args.txt nogui "$@"
```

With everything hopefully being finalized, we can at long last run the shell script to start our Minecraft server:

```
sudo sh run.sh
```

This process will take a while especially during the world generation stage. You will know it's done by "Done" message and the `;` of the Minecraft console prompt once everything is set up. You can now try connecting to your newly-created Minecraft server while keeping in mind you might need a specific version of the client (in this case I used the client version 1.26.0 which is managed by Prism Launcher).

The screenshot displays a Raspberry Pi desktop environment. The top terminal window shows the installation of the 'mc' command and the server's startup logs, including 'Preparing spawn area: 2%' and 'Successfully initialized permission handler'. The bottom window shows the 'Direct Connection' screen with the server address '10.100.26.159' entered and the 'Join Server' button highlighted.

14