Custom Distribution Docs

How to create Ubuntu 22.04 custom with ALG.

What do we need?

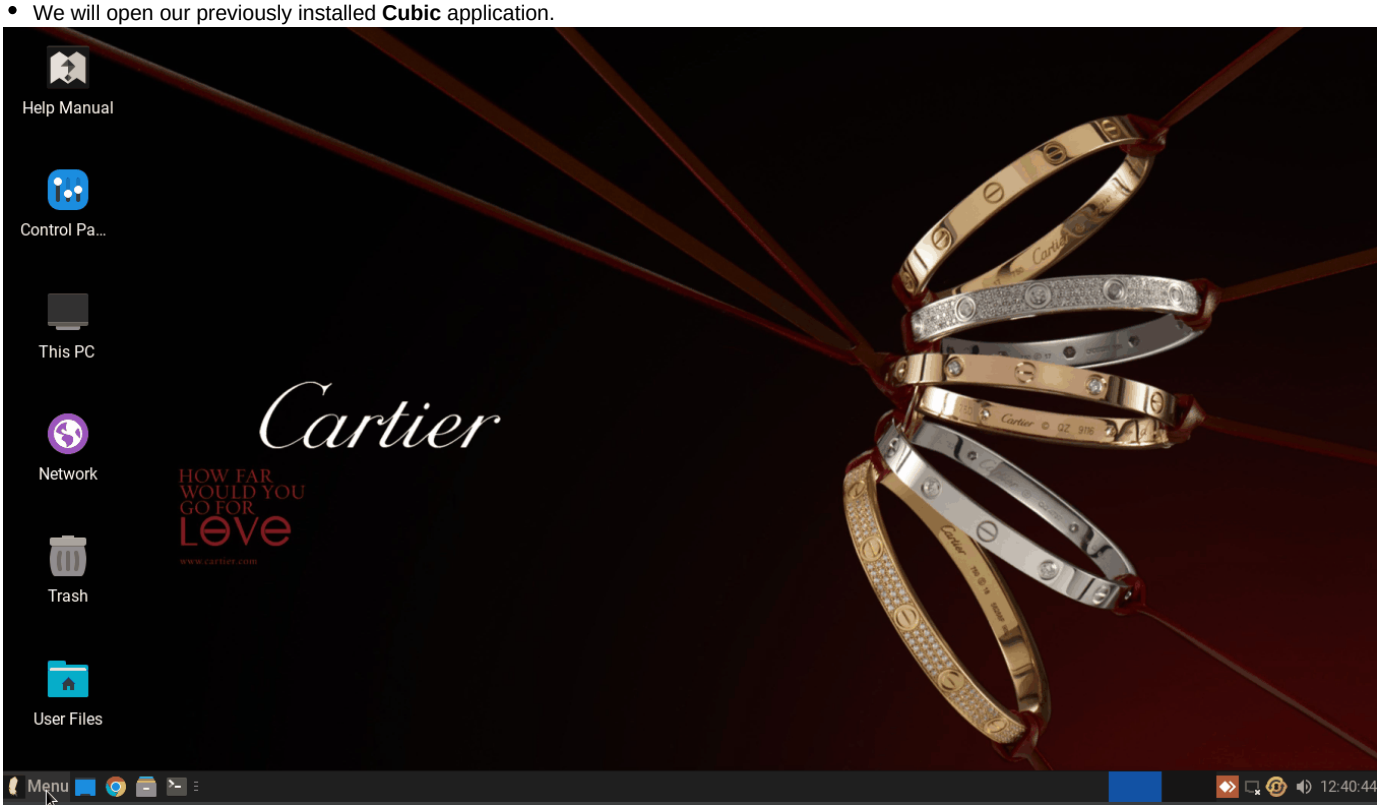
The first thing to know is that this tool has been designed to be used on an Ubuntu operating system. So we must have a version of Ubuntu installed on our system.

- **Cubic** (Github Official Repo)
  - Guide (https://ostechnix.com/how-to-create-a-custom-ubuntu-live-iso-image-with-cubic/)
- **ISO Ubuntu 22.04** (https://www.releases.ubuntu.com/22.04/ubuntu-22.04.2-desktop-amd64.iso)
- **Ubuntu 22.04 LTS (Jammy Jellyfish) complete sources.list (**https://gist.github.com/hakerdefo/9c99e140f543b5089e32176fe8721f5f **)**
- **Installation steps (**Vanilla Ubuntu setup **)**

Once we have all these elements at our disposal, we will proceed to create our distribution.

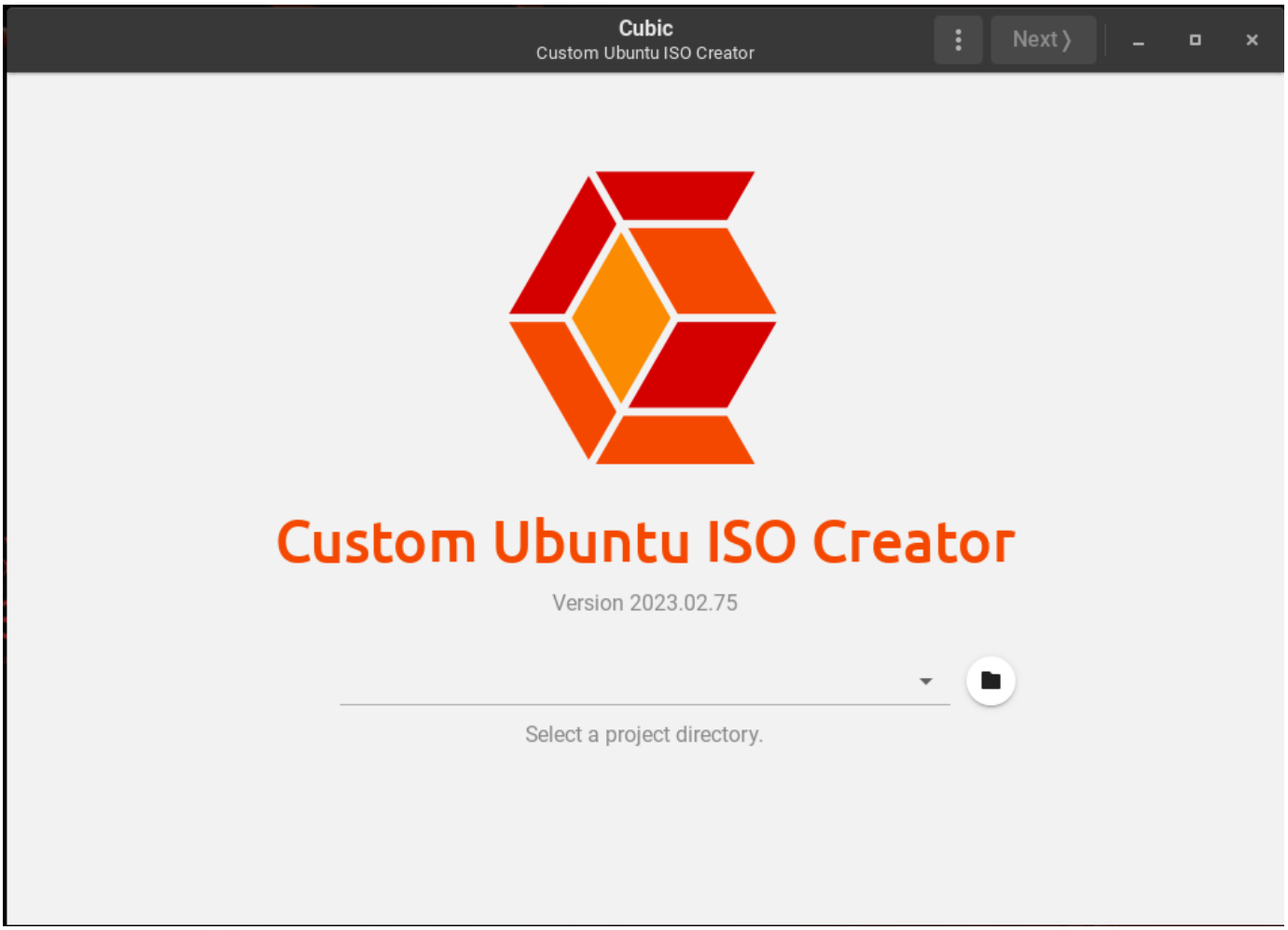
Step 0

- We will open our previously installed **Cubic** application.



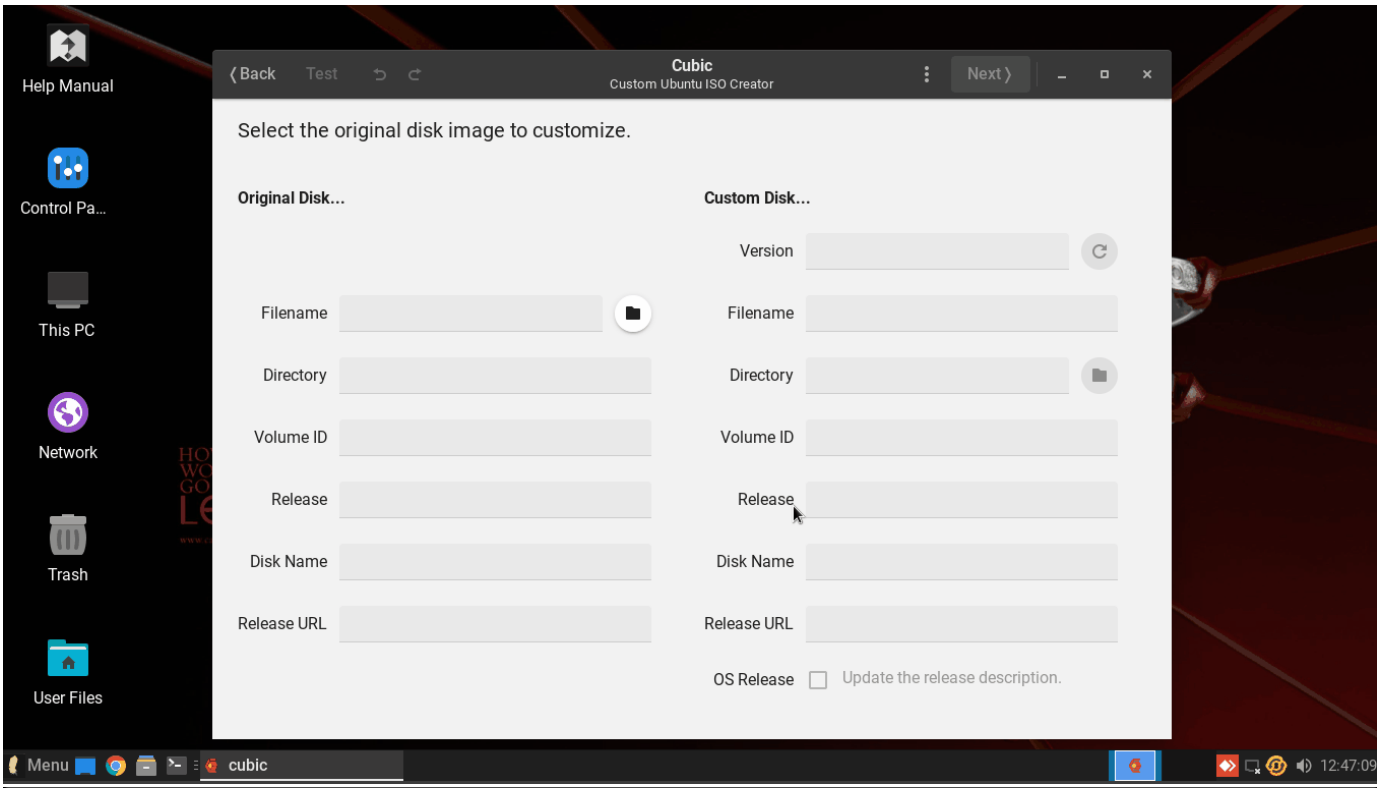
Step 1

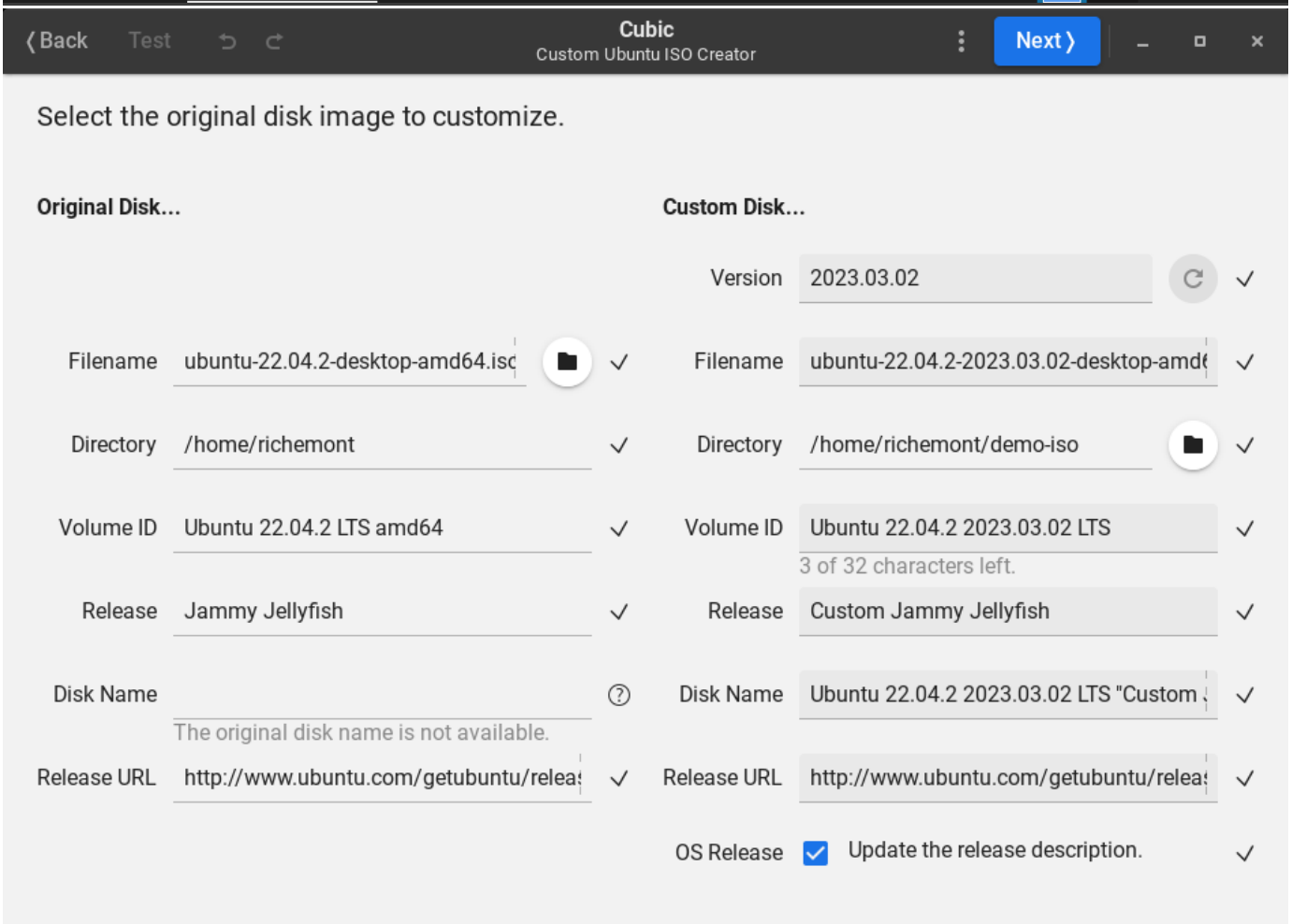
The first screen we are shown is the place where we are going to save our project and ISO image.



Once you have selected the folder where our information will be stored, click on the top button that says next.

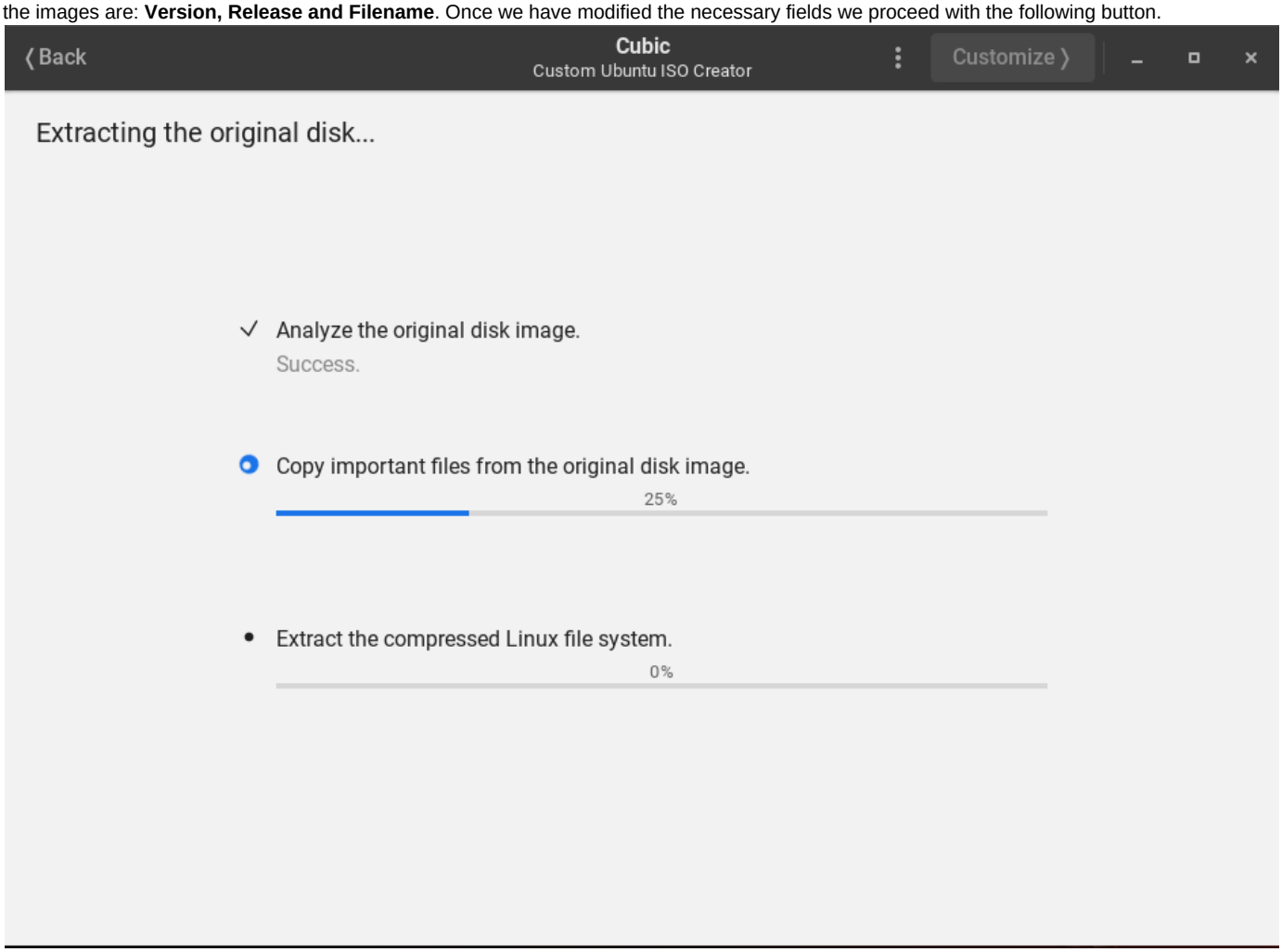
Step 2

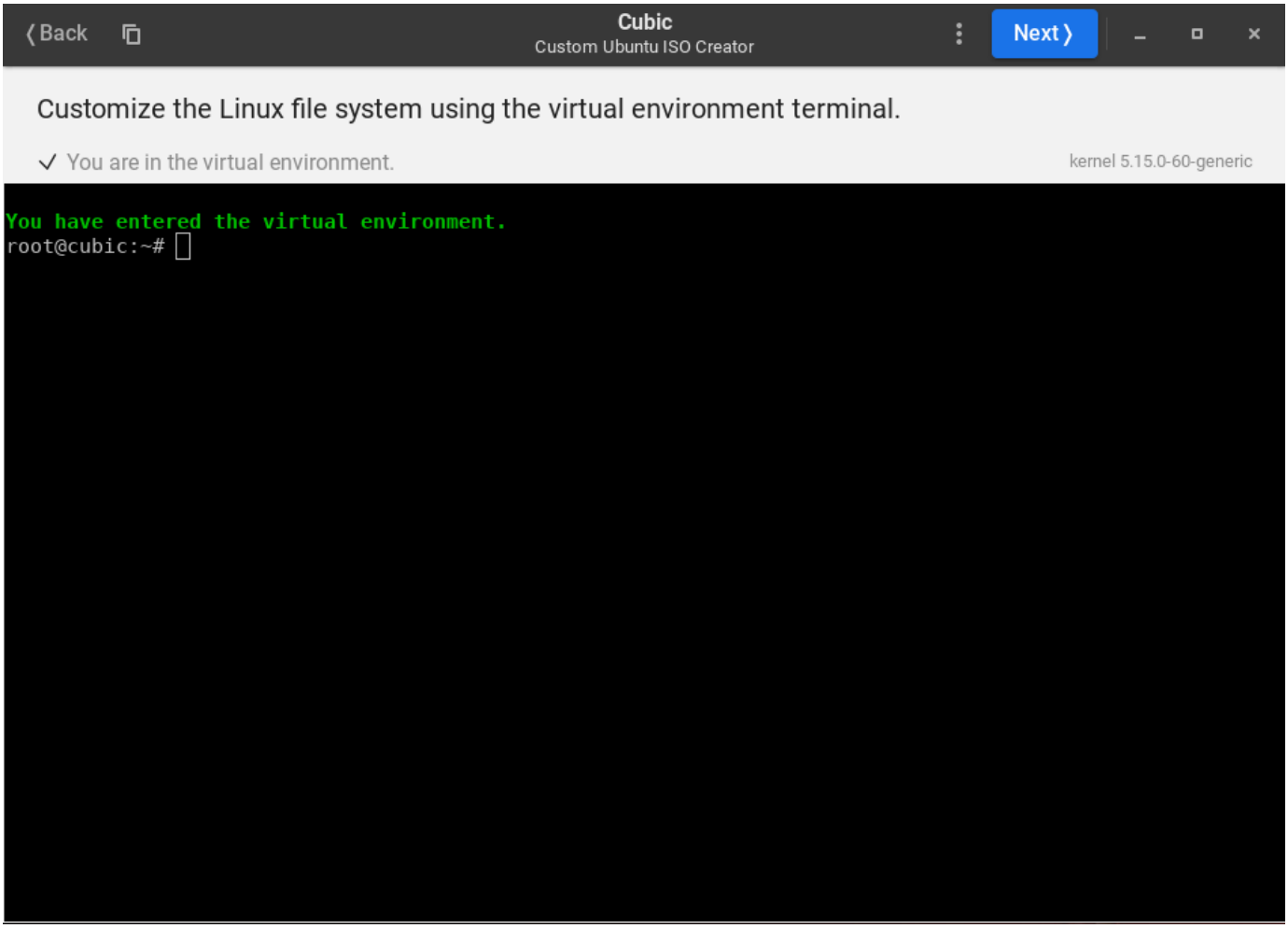




As we can see in this screen we have two types of information which we can modify. On the left side we have the information contained in the official Ubuntu image, and on the right side the information that will be saved in our custom image. It is advisable to change some of these

attributes to be able to differentiate the images in the future, as it is likely that different versions will be released. The fields that most differentiate the images are: **Version, Release and Filename**. Once we have modified the necessary fields we proceed with the following button.





Step 3:

We proceed to modify the source.list since the default one is limited. We will modify the content of the file here `/etc/apt/sources.list`

```
sudo nano /etc/apt/sources.list
```

Back

Cubic
Custom Ubuntu ISO Creator

Next

Customize the Linux file system using the virtual environment terminal.

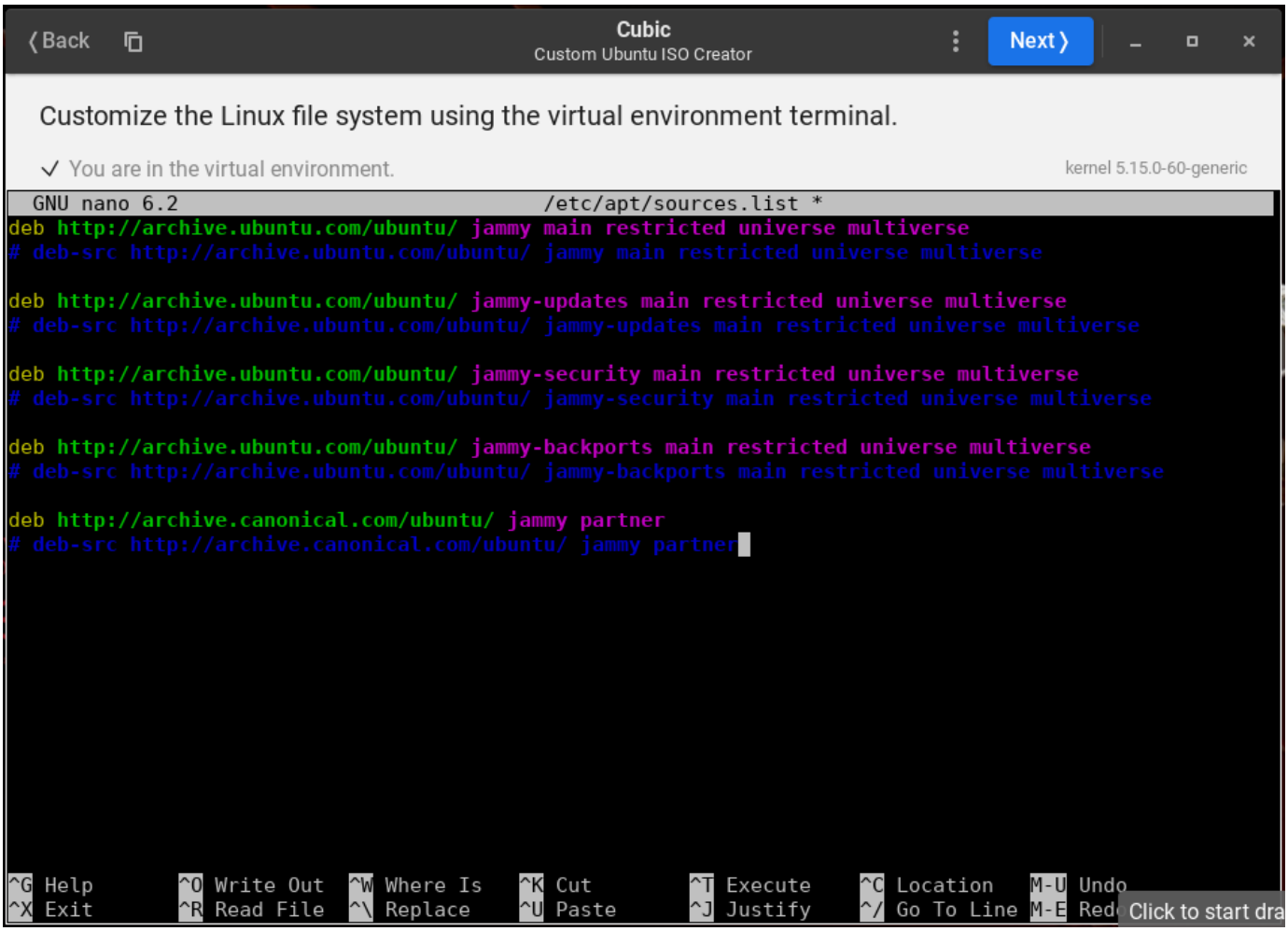
✓ You are in the virtual environment.

kernel 5.15.0-60-generic

```
  GNU nano 6.2                    /etc/apt/sources.list
deb http://archive.ubuntu.com/ubuntu/ jammy main restricted
deb http://security.ubuntu.com/ubuntu/ jammy-security main restricted
deb http://archive.ubuntu.com/ubuntu/ jammy-updates main restricted

                                [ Read 3 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify    ^/ Go To Line M-E Redo
```

Click to start d



With this step done, we will start the complete step-by-step installation of ALG until we reach point 4.

Step 4

Note: In the installation you will see this message each time you use `sudo` “sudo: unable to resolve host cubic: Temporary failure in name resolution“ No matter, everything is fine.

From step 4 onwards the installation varies a bit. I leave the modified steps below.

4 Install Vimba, Teamviewer, FFMPEG, Git, wmctrl and Ansible

```
# Download vimba software from https://www.alliedvision.com/en/products
/vimba-sdk/#c1497
cd /etc/skel
curl -L https://downloads.alliedvision.com/Vimba64_v6.0_Linux.tgz >
Vimba64_v6.0_Linux.tgz
tar -xzf Vimba64_v6.0_Linux.tgz -C /etc/skel

# TEAMVIEWER
cd /tmp
wget https://download.teamviewer.com/download/linux/signature
/TeamViewer2017.asc
sudo apt-key add TeamViewer2017.asc
sudo sh -c 'echo "deb http://linux.teamviewer.com/deb stable main" >>
/etc/apt/sources.list.d/teamviewer.list'
sudo apt update
sudo apt install -y teamviewer

# Install FFMPEG
sudo apt install -y ffmpeg
sudo apt install -y net-tools
sudo apt update && sudo apt upgrade -y

# Install Git
sudo apt install -y git

# Install wmctrl
sudo apt install -y wmctrl
```

5 Enable xhost +

```
# Execute only once
sudo echo "xhost +" >> /etc/profile
```

6 Repositories setup

```
# We need to add credentials in first instalation.
## Set vault
sudo git config --global credential.helper store
## Set vault in root folder
sudo git config --global credential.helper "store --file /root/.git-
credentials"
# Repositories setup
mkdir /etc/skel/alg-repositories
cd /etc/skel/alg-repositories

# input username and password for gitlab
# Complete command with credentials in https://brooklynlab.atlassian.net
/wiki/spaces/A/pages/16973825/Mega+computer+credentials#BTS-repository-
user
sudo git clone https://username:password@git.jolibrain.com/cartier
/cartier_lg_web.git

# input username and password for gitlab when asked
# Complete command with credentials in https://brooklynlab.atlassian.net
/wiki/spaces/A/pages/16973825/Mega+computer+credentials#BTS-repository-
user
sudo git clone https://username:password@gitlab.com/cartier-lab
/megacomputer-scripts.git

# input username and password for gitlab when asked
# Complete command with credentials in https://brooklynlab.atlassian.net
/wiki/spaces/A/pages/16973825/Mega+computer+credentials#BTS-repository-
user
sudo git clone https://username:password@gitlab.com/cartier-lab/mc-
playbooks.git
```

7 Disable Automatic Updates via Command Line

Update preferences are stored in the `/etc/apt/apt.conf.d/20auto-upgrades` file. Open it with nano or your favorite text editor to make some changes to it.

```
sudo nano /etc/apt/apt.conf.d/20auto-upgrades
```

To disable automatic updates completely, make sure all these directives are set to “0”. When done, save your changes and exit the file.

```
APT::Periodic::Update-Package-Lists "0";
APT::Periodic::Download-Upgradeable-Packages "0";
APT::Periodic::AutocleanInterval "0";
APT::Periodic::Unattended-Upgrade "0";
```

8 Create script “First Start“

```
cd /etc/skel
touch first_start.sh
sudo chmod +x first_start.sh
nano first_start.sh
# Add content that appear below.
```

```
#!/bin/bash

# Check if the current user is root or has sudo privileges
if [[ $EUID -ne 0 ]]; then
    echo "This script must be run as root or with sudo privileges"
    exit 1
fi
set -e
# Sequence of commands you want to run
# Install Vimba
vimba=$(find /home -type d -name VimbaUSBTL)
cd $vimba
sudo ./Uninstall.sh
sudo ./Install.sh
# Login in registries
sudo docker login registry.gitlab.com
sudo docker login https://cartierdocker.deepdetect.com
# Give permission to Docker
sudo usermod -aG docker $SUDO_USER
sudo chown -R $SUDO_USER /var/run/docker.sock
sudo chown -R $SUDO_USER /usr/local/bin/docker-compose
# Disable ipv6
sudo sysctl net.ipv6.conf.all.disable_ipv6=1
#find repos
#alg_backend=$(find /home -type d -name alg-backend)
megacomputer_scripts=$(find /home -type d -name megacomputer-scripts)
cartier_lg_web=$(find /home -type d -name cartier_lg_web)
standalone=$(find /home -type d -name standalone_setup_realtime)
alg=$(find /home -type d -name alg-repositories)
# Add safe directory
```

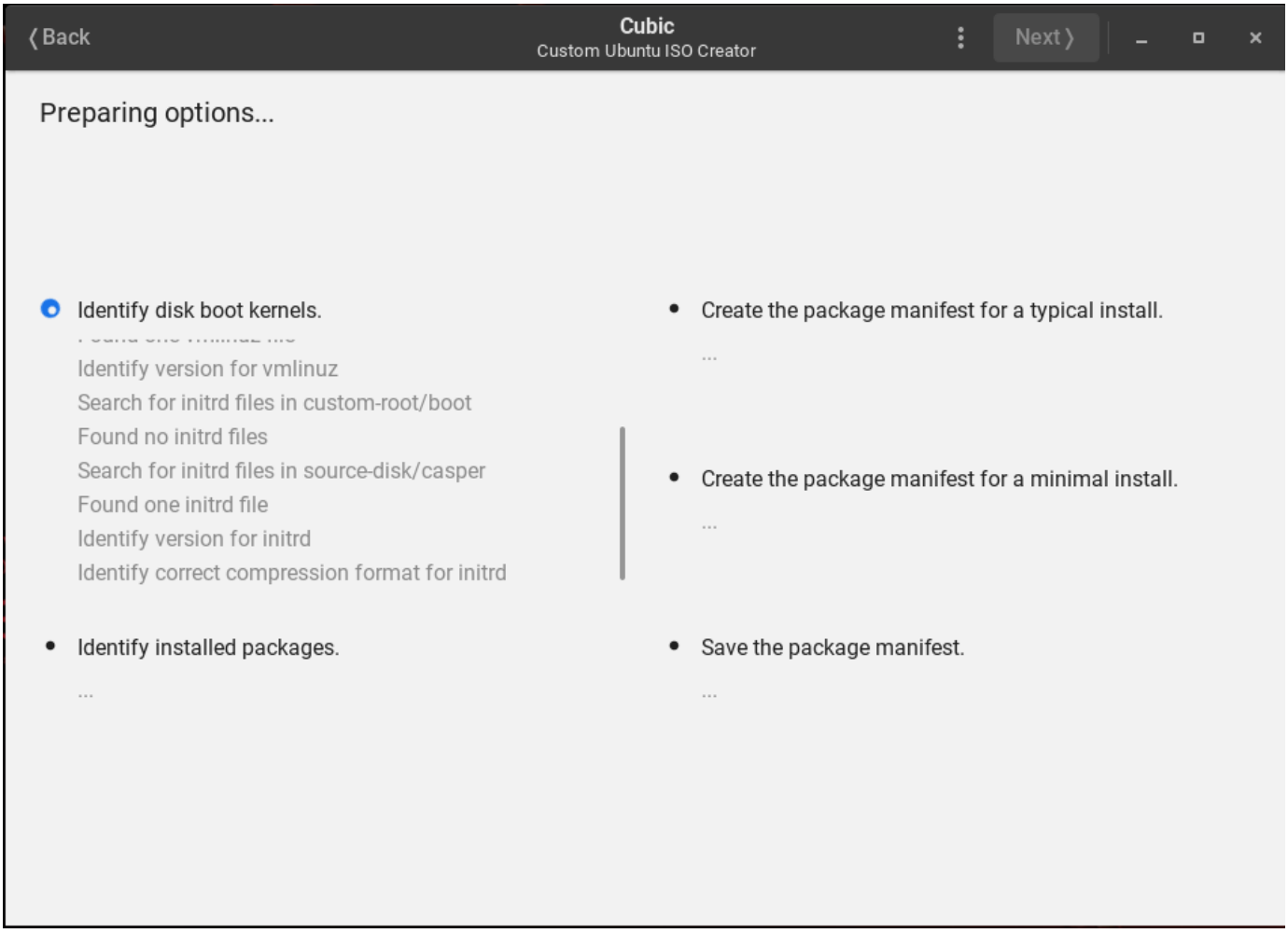
```
sudo git config --global --add safe.directory $megacomputer_scripts
#sudo git config --global --add safe.directory $alg_backend
sudo git config --global --add safe.directory $cartier_lg_web

## Setup default version alg
# Permission to alg directory
sudo chown -R $USER $alg
# Define VARS
export DATA_ROOT_PATH=/data/cartier/cartier_realtime_data
export COMPOSE_UID=$(id -u)
export COMPOSE_GUID=$(id -g)
cd $standalone
sudo docker-compose up -d
# Copy alg files into jolibrain
cp $megacomputer_scripts/config/nginx.conf $cartier_lg_web/docker
/standalone_setup_realtime/config/nginx
cp $megacomputer_scripts/config/docker-compose.yml $cartier_lg_web
/docker/standalone_setup_realtime
cp $megacomputer_scripts/config/.env_api_backend $cartier_lg_web/docker
/standalone_setup_realtime
cp $megacomputer_scripts/config/.env_postgres $cartier_lg_web/docker
/standalone_setup_realtime
cp $megacomputer_scripts/config/.env_elastic $cartier_lg_web/docker
/standalone_setup_realtime
cp $megacomputer_scripts/config/filebeat.yml $cartier_lg_web/docker
/standalone_setup_realtime/config
cp $megacomputer_scripts/config/logstash.conf $cartier_lg_web/docker
/standalone_setup_realtime/config
# Next start to apply alg changes
sudo docker-compose up -d
# Install
cd $megacomputer_scripts
sudo ./install.sh MEGA_COMPUTER 192.168.121.101 root root /data/cartier
/cartier_realtime_data
sudo apt autoremove -y
# Programed restart
echo "Press any key to cancel restart"
sleep 30

if read -t 0; then
  echo "Restart canceled"
else
  echo "Restarting..."
  shutdown -r now
fi
```

When we have completed the full installation and all our files are inside our custom distribution, click next, don't worry if you forget any step you can always go back to modify the project.

Step 5


Back
Cubic
Custom Ubuntu ISO Creator
Next
Preparing options...
Identify disk boot kernels.
Identify version for vmlinuz
Search for initrd files in custom-root/boot
Found no initrd files
Search for initrd files in source-disk/casper
Found one initrd file
Identify version for initrd
Identify correct compression format for initrd
Identify installed packages.
...
Create the package manifest for a typical install.
...
Create the package manifest for a minimal install.
...
Save the package manifest.
...

Back

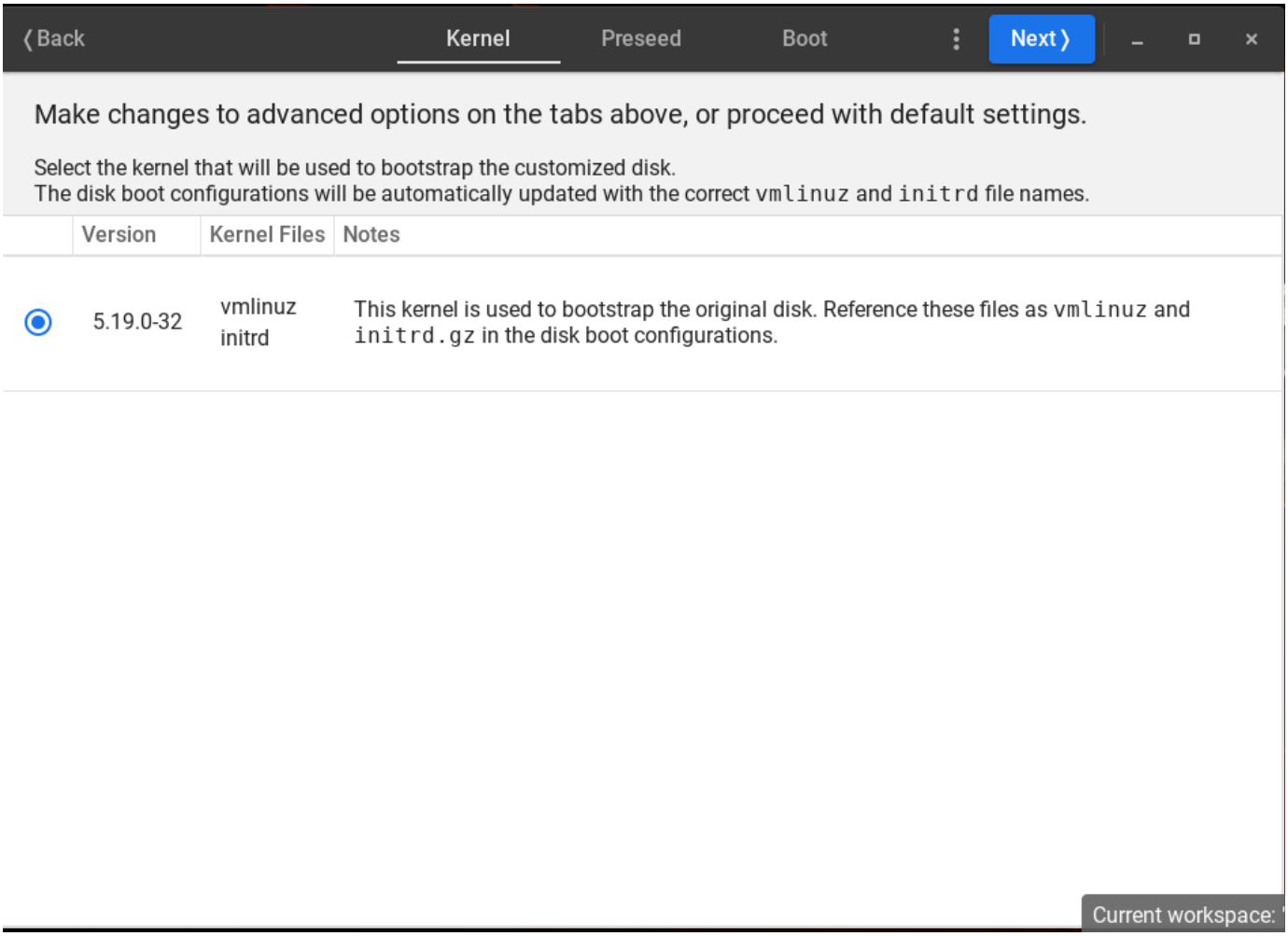
Cubic
Custom Ubuntu ISO Creator

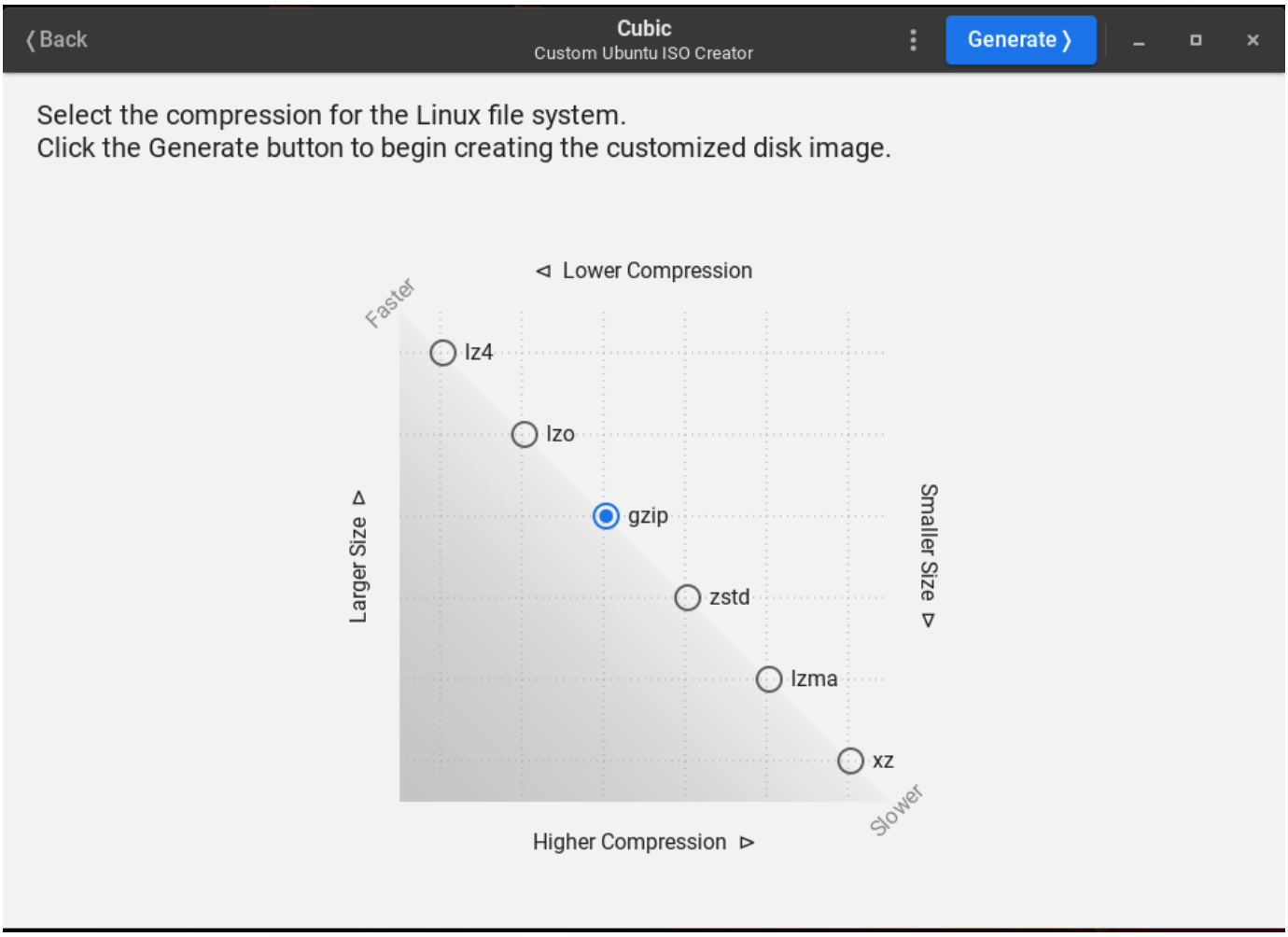
Next

Select packages that will be automatically removed for a typical or minimal install.

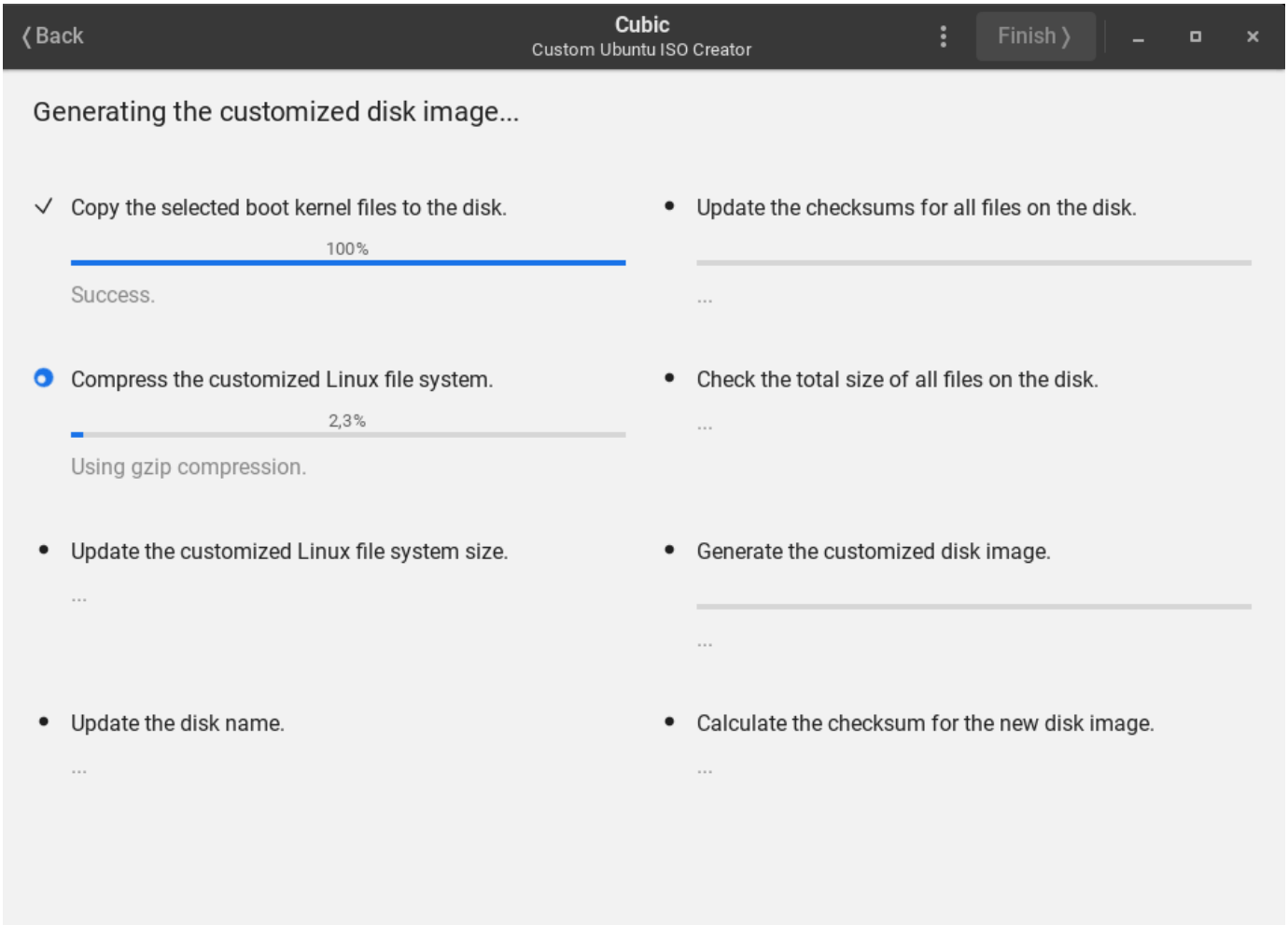
All packages listed are available in the live environment. Check marked packages will be removed during installation.

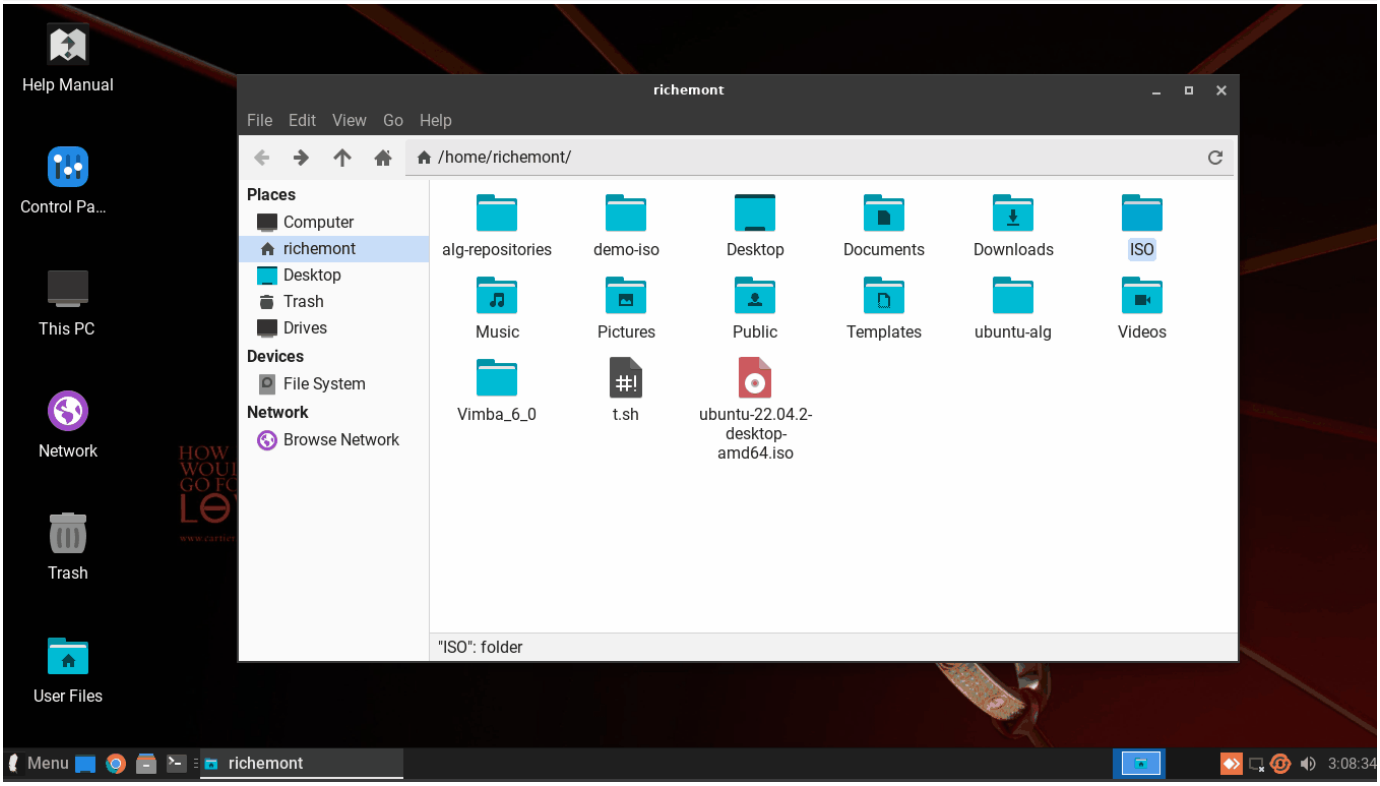
Typical	Minimal	Package	Version
☐	☐	accountsservice	22.07.5-2ubuntu1.3
☐	☐	acl	2.3.1-1
☐	☐	acpi-support	0.144
☐	☐	acpid	1:2.0.33-1ubuntu1
☑	☑	adcli	0.9.1-1ubuntu2
☐	☐	adduser	3.118ubuntu5
☐	☐	adwaita-icon-theme	41.0-1ubuntu1
☐	☑	aisleriot	1:3.22.22-1
☐	☐	alsa-base	1.0.25+dfsg-0ubuntu7
☐	☐	alsa-topology-conf	1.2.5.1-2
☐	☐	alsa-ucm-conf	1.2.6.3-1ubuntu1.4
☐	☐	alsa-utils	1.2.6-1ubuntu1
☐	☐	amd64-microcode	3.20191218.1ubuntu2
☐	☐	anacron	2.3-31ubuntu2
☐	☐	apg	2.2.3.dfsg.1-5build2
☐	☐	apparmor	3.0.4-2ubuntu2.1
☐	☐	apport	2.20.11-0ubuntu82.3


Back
Kernel
Preseed
Boot
Next
Make changes to advanced options on the tabs above, or proceed with default settings.
Select the kernel that will be used to bootstrap the customized disk.
The disk boot configurations will be automatically updated with the correct vmlinuz and initrd file names.
Version
Kernel Files
Notes
5.19.0-32
vmlinuz
initrd
This kernel is used to bootstrap the original disk. Reference these files as vmlinuz and initrd.gz in the disk boot configurations.
Current workspace:



Finally, click the **generate** button and the build process of your distribution will start.





To create an USB Live with generated ISO follow the steps on this tutorial:https://www.makeuseof.com/create-bootable-usb-drive-with-etcher-linux/