AI large model environment deployment

The SSD mounting and Ollam model download path changes are optional for users.

> **Demo Environment**

Development Board: RDK S100(P) Motherboard

Ollama is an open-source tool designed to simplify the deployment and operation of large language models, allowing users to use high-quality language models in their local environment.

1. Large Language Models (LLM)

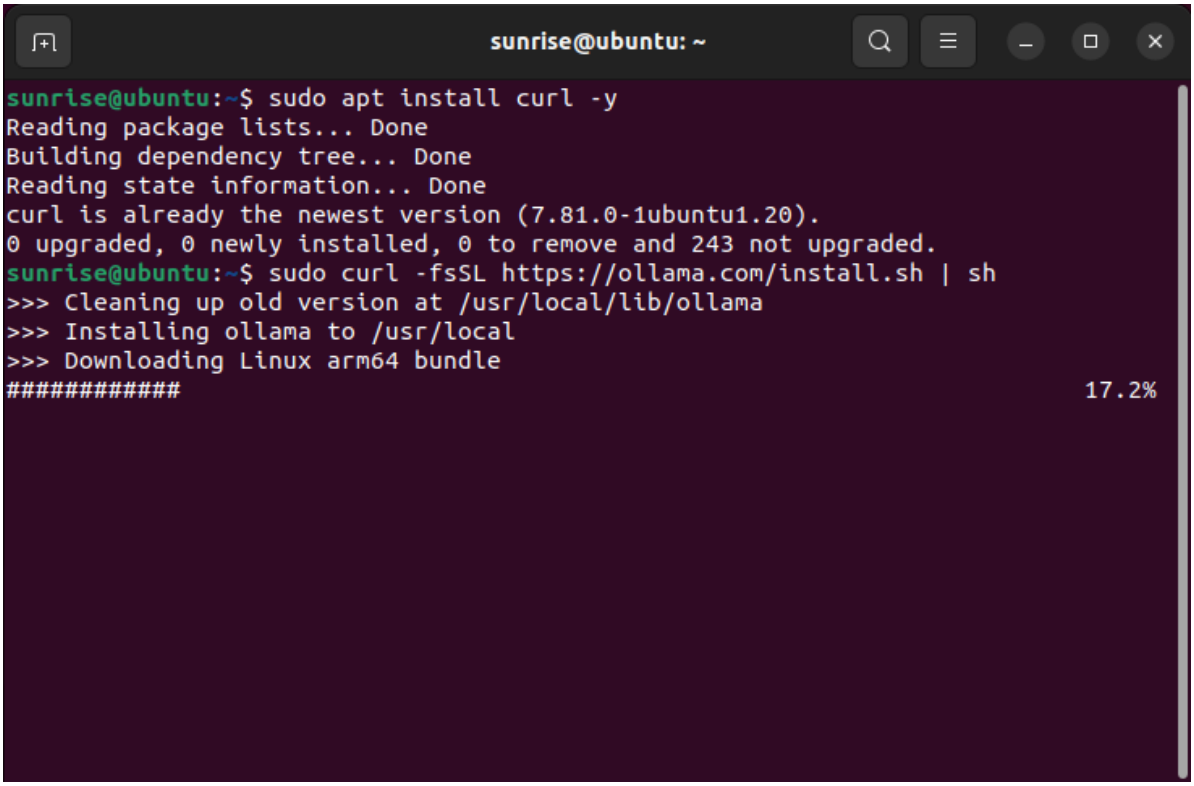
Large Language Models (LLMs) are a class of advanced text generation systems based on artificial intelligence. Their main characteristic is their ability to learn and understand human language through large-scale training data and generate natural and fluent text.

2. Ollam Installation

Note: Ollam pull failures are normal. You can try multiple times, or try pulling using a proxy. You'll need to search online for instructions on using a proxy.

Script Installation

```
sudo apt install curl -y
sudo curl -fsSL https://ollama.com/install.sh | sh
```



The entire installation process takes a relatively long time. Please be patient!

Once the installation is complete, a warning message will appear; ignore it:

```
sunrise@ubuntu:~$ sudo curl -fsSL https://ollama.com/install.sh | sh
>>> Cleaning up old version at /usr/local/lib/ollama
>>> Installing ollama to /usr/local
>>> Downloading Linux arm64 bundle
######################################################################## 100.0%
>>> Creating ollama user...
>>> Adding ollama user to render group...
>>> Adding ollama user to video group...
>>> Adding current user to ollama group...
>>> Creating ollama systemd service...
>>> Enabling and starting ollama service...
Created symlink /etc/systemd/system/default.target.wants/ollama.service → /etc/s
ystemd/system/ollama.service.
>>> The Ollama API is now available at 127.0.0.1:11434.
>>> Install complete. Run "ollama" from the command line.
WARNING: No NVIDIA/AMD GPU detected. Ollama will run in CPU-only mode.
sunrise@ubuntu:~$
```

Use `ollama -v` to check the Ollam version number:

```
sunrise@ubuntu:~$ ollama -v
ollama version is 0.9.3
```

3. Using Ollam

Type `ollama` in the terminal to see usage prompts:

```
sunrise@ubuntu:~$ ollama
Usage:
  ollama [flags]
  ollama [command]

Available Commands:
  serve       Start ollama
  create      Create a model from a Modelfile
  show        Show information for a model
  run         Run a model
  stop        Stop a running model
  pull        Pull a model from a registry
  push        Push a model to a registry
  list        List models
  ps          List running models
  cp          Copy a model
  rm          Remove a model
  help        Help about any command

Flags:
  -h, --help      help for ollama
  -v, --version   Show version information

Use "ollama [command] --help" for more information about a command.
```

Command	Function
ollama serve	Start Ollam
ollama create	Create a model from a model file
ollama show	Display model information
ollama run	Run a model
ollama stop	Stop a running model
ollama pull	Pull a model from the registry
ollama push	Push a model to the registry
ollama list	List models
ollama ps	List running models
ollama cp	Copy a model
ollama rm	Delete a model
ollama help	Get help information for any command

4. Uninstalling Ollama

- Delete service

```
sudo systemctl stop ollama
sudo systemctl disable ollama
sudo rm /etc/systemd/system/ollama.service
```

- Delete file

```
sudo rm $(which ollama)
```

- Delete model and service user and group

```
sudo rm -r /usr/share/ollama
sudo userdel ollama
sudo groupdel ollama
```

5. Memory Allocation Adjustment

1. Insufficient Available Memory at Factory Start

Newly manufactured RDK S100(P) motherboards may have very little available memory. You can run the following two commands to update and increase the available memory:

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

After updating, restart. You can then use the `free -h` command to see that your memory has increased.

2. Customers who need to adjust memory can continue reading

RDK provides a flexible memory allocation adjustment scheme.

This scheme supports multiple memory strategies and can be quickly switched according to the application scenario (such as BPU priority, CPU priority, or balanced mode).

Since our factory-installed available memory is relatively small, some large models with large amounts of data cannot run, so we need to increase the available memory.

1. Open a terminal and enter the following commands:

```
touch update_ion_dtb.sh` `gedit update_ion_dtb.sh`
```

2. Then paste the following content into a text file, save and exit:

```
#!/bin/bash

help_msg()
{
    echo "Usage: $0 <bpu_first | cpu_first | balanced | default>"
    echo "Example: $0 bpu_first"
    exit 1
}

if [ $# -ne 1 ]; then
    help_msg
fi
```

```
target=$1

# Check if dtc command exists
if ! command -v dtc &> /dev/null; then
    apt update
    apt install device-tree-compiler -y
fi

function update_tmp_dts()
{
    COMPATIBLE="$1"
    NEW_REG="$2"
    tmp_dts_f=$3
    echo "Setting $COMPATIBLE's reg to <$NEW_REG>"
    # TODO: The following implementation assumes all ion nodes is properly
    #       layed out as reg directly following compatible format, use awk
    #       in the future to properly handle dts node.
    sed -i "/compatible = \"$COMPATIBLE\";/{n;d}" "${tmp_dts_f}" || {
        echo "Delete Original reg failed!"
        exit 1
    }

    sed -i "/compatible = \"$COMPATIBLE\";/a reg = <$NEW_REG>;" "${tmp_dts_f}" ||
{
        echo "Add"
        exit 1
    }
}

function get_dtb()
{
    boardid_sys_path="/sys/class/boardinfo/adc_boardid"
    if [ -f "$boardid_sys_path" ]; then
        boardid="$(cat $boardid_sys_path)"

        # get chip
        case $boardid in
            *"64"*) ;&
            *"65"*)
                chip_str="s100p"
            ;;
            *"6A"*) ;&
            *"6B"*)
                chip_str="s100"
            ;;
        esac
        case $boardid in
            *"60")
                hw_str="v1-21"
            ;;
            *"70")
                hw_str="v1-20"
            ;;
            *"84")
                hw_str="v0p5"
            ;;
            *"85")
                hw_str="v0p6"
```

```
            ;;
            *"86")
                hw_str="v1p0"
            ;;
        esac
        echo "rdk-${chip_str}-${hw_str}.dtb"
    else
        echo "Invalid"
    fi
}

# Create tmp dir
TMP_DIR=$(mktemp -d)
trap "rm -rf $TMP_DIR" EXIT

dtb_name=$(get_dtb)

if [ "$dtb_name" = "Invalid" ];then
    echo "ERROR: Cannot find valid dtb for current board!"
    exit 1
fi
INPUT_DTB="/boot/hobot/${dtb_name}"
ORIG_INPUT_DTB="${INPUT_DTB}.bak"
OUTPUT_DTB="${INPUT_DTB}"

case $target in
    "balanced")
        if [[ "${INPUT_DTB}" = *"-s100p-"* ]];then
            # ion-pool 768OMiB
            ion_pool_reg_val="0x4 0x00000000 0x1 0xe0000000"
            # ion-carveout 128MiB
            ion_carveout_reg_val="0x8 0x00000000 0x0 0x08000000"
            # ion-cam 128MiB
            ion_cma_reg_val="0xc 0x80000000 0x0 0x08000000"
        else
            # ion-pool 384OMiB
            ion_pool_reg_val="0x4 0x00000000 0x0 0xF0000000"
            # ion-carveout 128OMiB
            ion_carveout_reg_val="0x8 0x00000000 0x0 0x50000000"
            # ion-cam 1GiB
            ion_cma_reg_val="0xc 0x80000000 0x0 0x40000000"
        fi
    ;;
    "bpu_first")
        if [[ "${INPUT_DTB}" = *"-s100p-"* ]];then
            # ion-pool 768OMiB
            ion_pool_reg_val="0x4 0x00000000 0x1 0xe0000000"
            # ion-carveout 768OMiB
            ion_carveout_reg_val="0x8 0x00000000 0x1 0xe0000000"
            # ion-cma 512OMiB
            ion_cma_reg_val="0xc 0x80000000 0x1 0x40000000"
        else
            # ion-pool 384OMiB
            ion_pool_reg_val="0x4 0x00000000 0x0 0xF0000000"
            # ion-carveout 230MiB
            ion_carveout_reg_val="0x8 0x00000000 0x0 0x90000000"
            # ion-cam 2GiB
            ion_cma_reg_val="0xc 0x80000000 0x0 0x80000000"
```

```
        fi
    ;;
    "cpu_first")
            # ion-pool 1GiB
            ion_pool_reg_val="0x4 0x00000000 0x0 0x40000000"
            # ion-carveout 512MiB
            ion_carveout_reg_val="0x8 0x00000000 0x0 0x20000000"
            # ion-cma 512MiB
            ion_cma_reg_val="0xc 0x80000000 0x0 0x20000000"
    ;;
    "default")
        echo "INFO: Restoring to default..."
        if [ -f ${ORIG_INPUT_DTB} ];then
            cp ${ORIG_INPUT_DTB} ${OUTPUT_DTB}
            rm ${ORIG_INPUT_DTB}
        fi
        exit 0
    ;;
    *)
        echo "ERROR: Invalid option $target"
        help_msg
    ;;
esac


# Set ion region sizes according to input

# DTB->DTS
if [ ! -f ${ORIG_INPUT_DTB} ];then
    cp ${INPUT_DTB} ${ORIG_INPUT_DTB}
    echo "INFO: Backup created:${ORIG_INPUT_DTB}"
else
    echo "INFO: Backup(${ORIG_INPUT_DTB}) already exists, skip backup..."
fi
tmp_dts_f="$TMP_DIR/temp.dts"
dtc -I dtb -O dts -o "${tmp_dts_f}" "$INPUT_DTB" > /dev/null 2>&1 || { echo
"ERROR: Decompile $INPUT_DTB failed!"; exit 1; }

update_tmp_dts "ion-pool" "${ion_pool_reg_val}" "$tmp_dts_f"
update_tmp_dts "ion-carveout" "${ion_carveout_reg_val}" "$tmp_dts_f"
update_tmp_dts "ion-cma" "${ion_cma_reg_val}" "$tmp_dts_f"

# DTS->DTB
dtc -I dts -O dtb -o "$OUTPUT_DTB" "$tmp_dts_f" > /dev/null 2>&1 || { echo
"ERROR: Compile $INPUT_DTB from $TMP_DIR/modified.dts failed!"; exit 1; }


echo "INFO: Update ${OUTPUT_DTB} for $target Done!"
```

3. Next, run the following in the terminal:

```
sudo chmod 777 update_ion_dtb.sh
sudo ./update_ion_dtb.sh cpu_first
```

```
sunrise@ubuntu:~$ sudo ./update_ion_dtb.sh cpu_first
INFO: Backup(/boot/hobot/rdk-s100-v1p0.dtb.bak) already exists, skip backup...
Setting ion-pool's reg to <0x4 0x00000000 0x0 0x40000000>
Setting ion-carveout's reg to <0x8 0x00000000 0x0 0x20000000>
Setting ion-cma's reg to <0xc 0x80000000 0x0 0x20000000>
INFO: Update /boot/hobot/rdk-s100-v1p0.dtb for cpu_first Done!
sunrise@ubuntu:~$
```

Finally, restart the motherboard, and then use the `free -h` command to see the changes in your memory allocation.

This script supports modification of the following four modes:

Mode Name	Description
bpu_first	Prioritizes memory allocation for BPUs, suitable for AI algorithm inference scenarios
cpu_first	Prioritizes memory allocation for CPUs, suitable for traditional computing load scenarios
balanced	Balances memory allocation between BPUs and CPUs, suitable for general scenarios
default	Restores factory default memory allocation (restores to .bak backup)

Our Ollama system relies entirely on CPU computation for running large models, so we use the cpu_first option.

6. SSD Mounting (Optional)

Models pulled by Ollama are stored on eMMC by default, occupying system space. You can use an SSD to expand storage and store large models on the SSD.

If you purchased the SSD as part of the Yabo RDK S100(P) package, it comes pre-installed with some large Ollama models and other models required by FastSD CPU. Therefore, after connecting and booting, it will automatically mount these models, and you won't need to download the existing model files manually.

Simply follow the tutorial to change the path for the Ollama or FastSD CPU models to the SSD, and you can directly use the models already on the SSD.

For users who purchased the packaged SSD, you can skip to the following section to specify the model path:

Ollama Model Specification: Skip to point 4 of `7. Changing the Ollama Model Download Path (Optional)` in this tutorial and modify ollama.service to begin.

FastSD CPU Model Specification: Before starting the FastSD CPU web UI, enter `export HF_HOME=/media/ssd/HF_Model` and then start it. If you find it troublesome to type this every time you start the computer, you can directly add it to your .bashrc file.

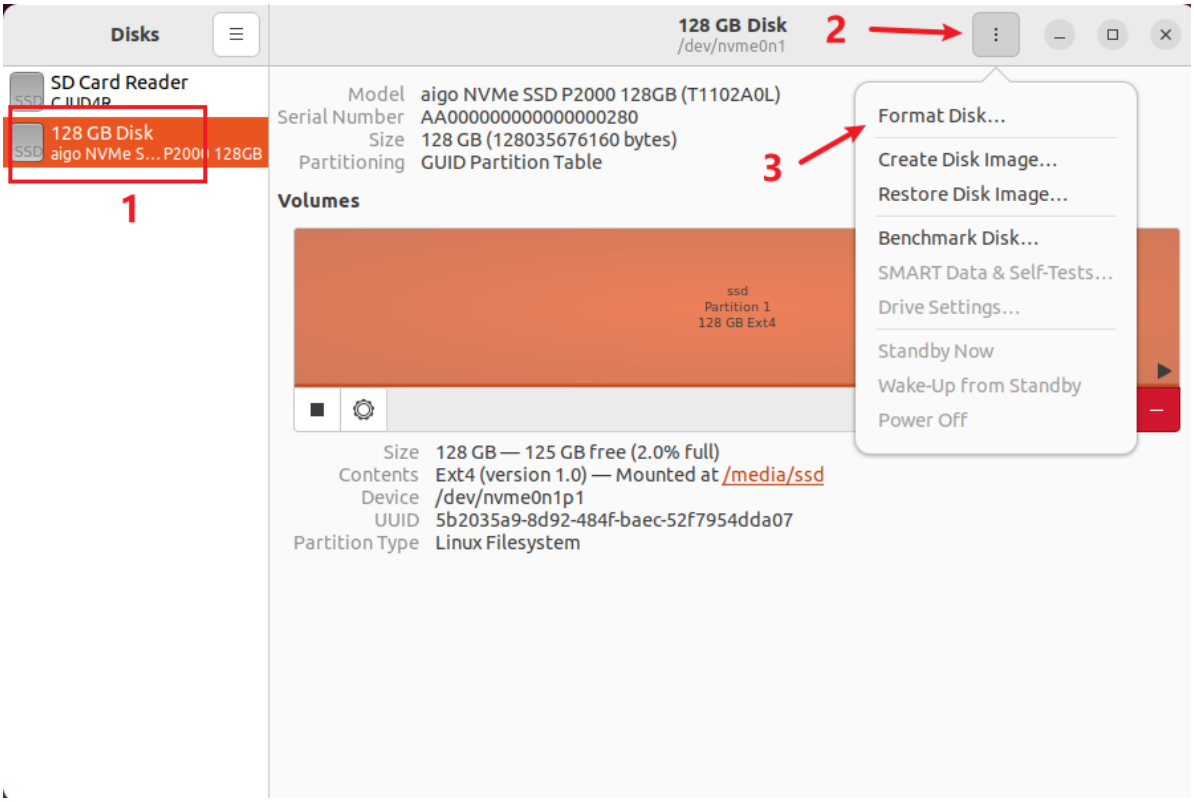
The default mount path for the SSD included in the package is: /media/ssd

The following are the mounting instructions for SSDs not included in the package:

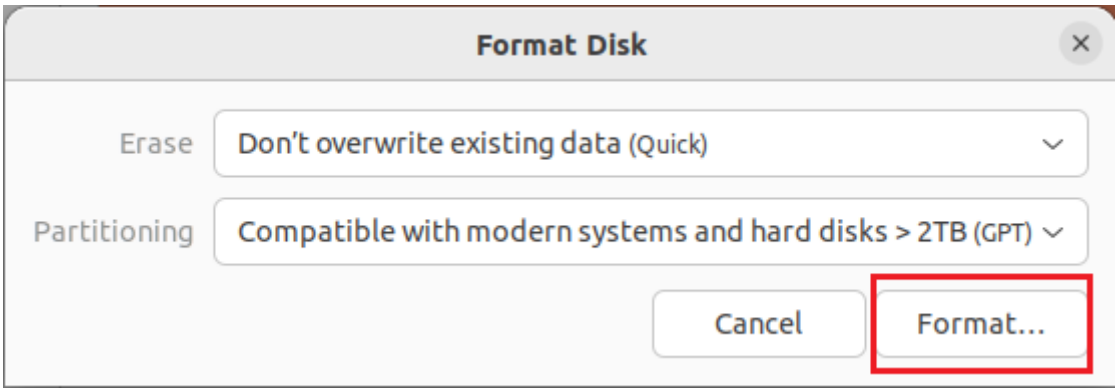
1. Open a terminal and enter the command to open Disks:

```
sudo gnome-disks
```

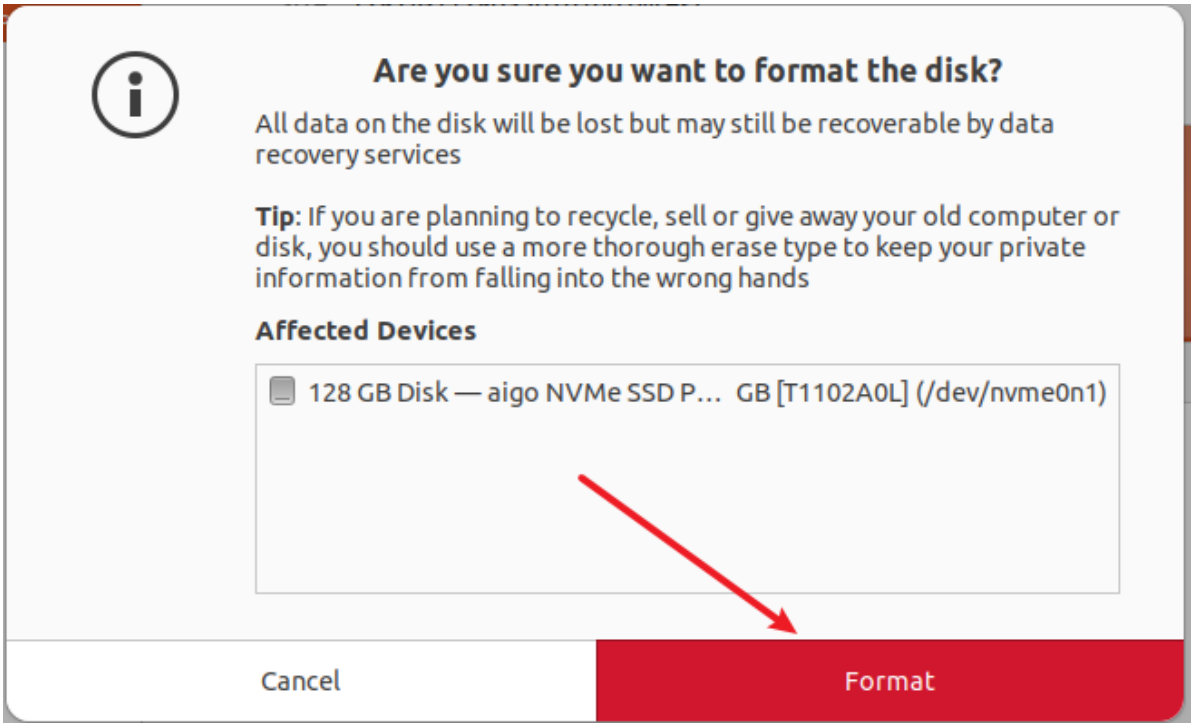
2. In the left sidebar of Disks, select the SSD you connected, and then click the three dots in the upper right corner of the interface. Then click Format DIsk



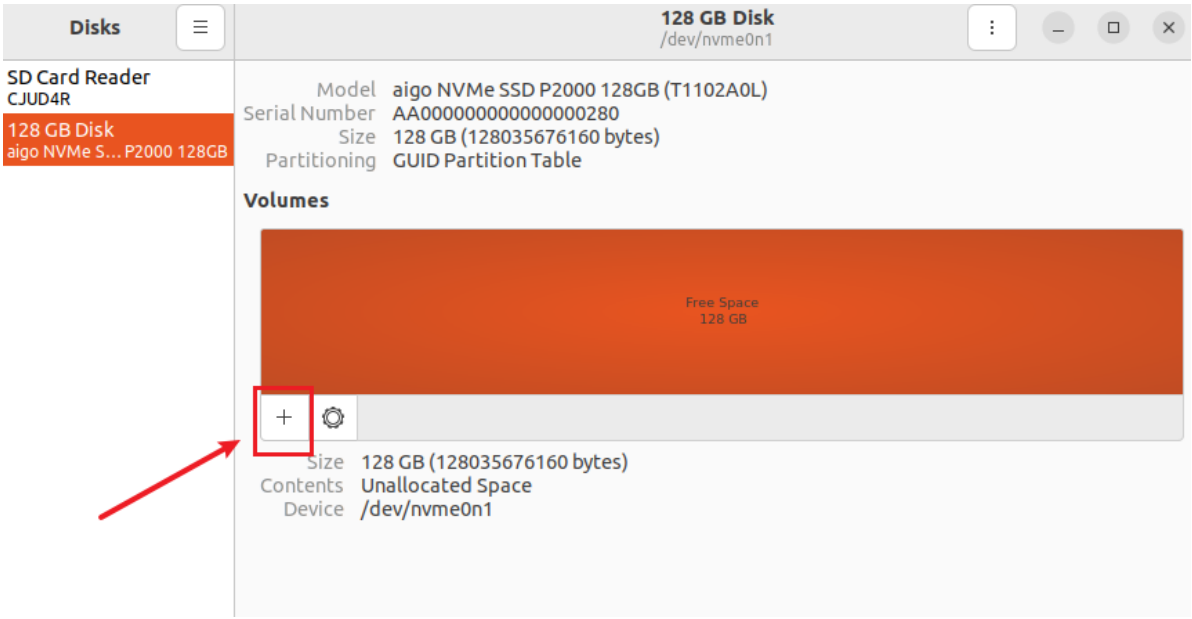
3. The default selection is fine; no changes are needed. Just click Format.



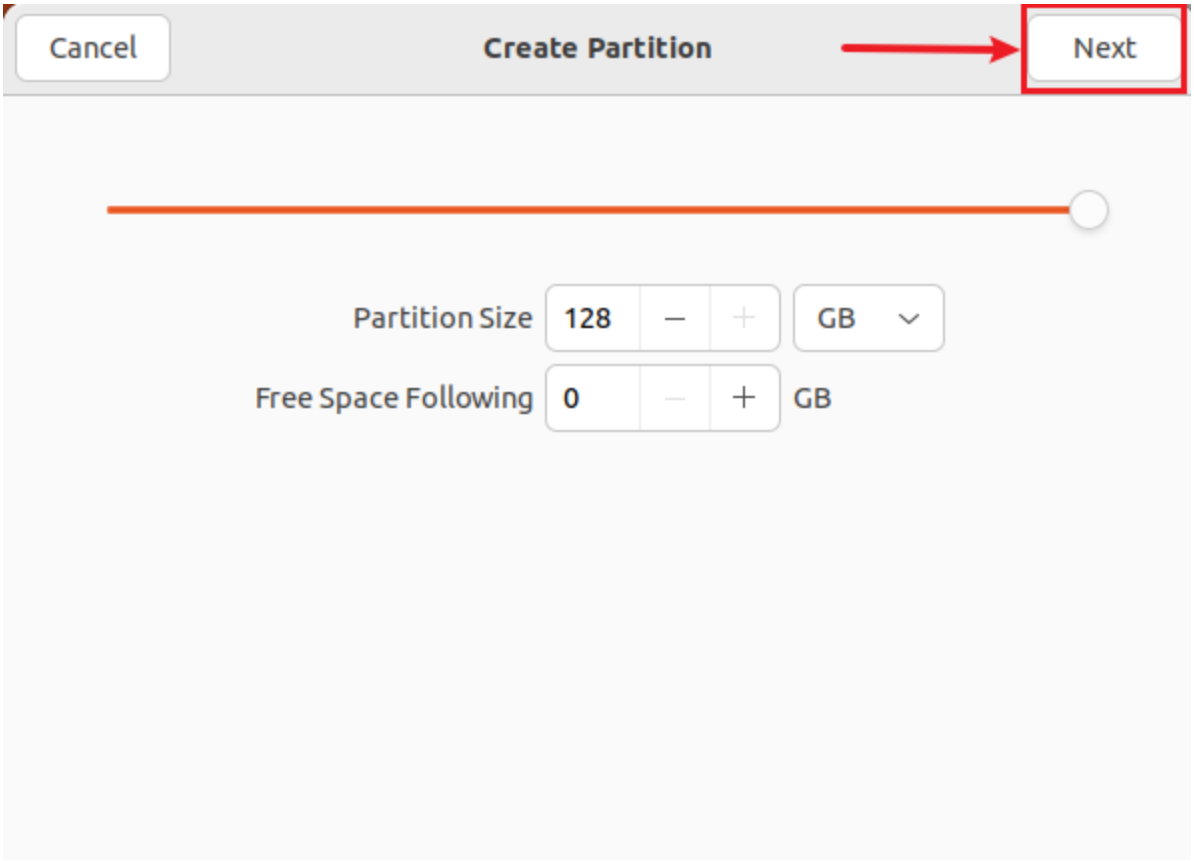
4. Click Format.



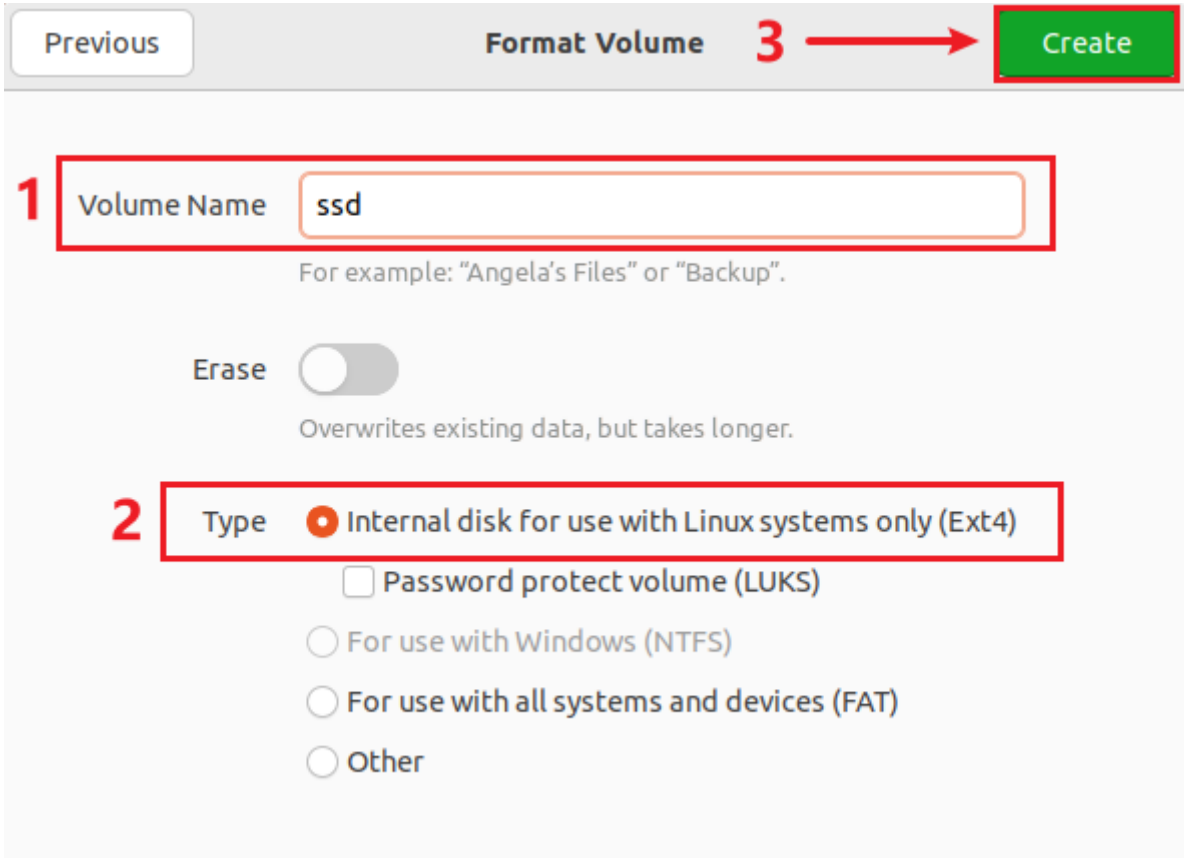
5. Click the plus sign to create a new partition.



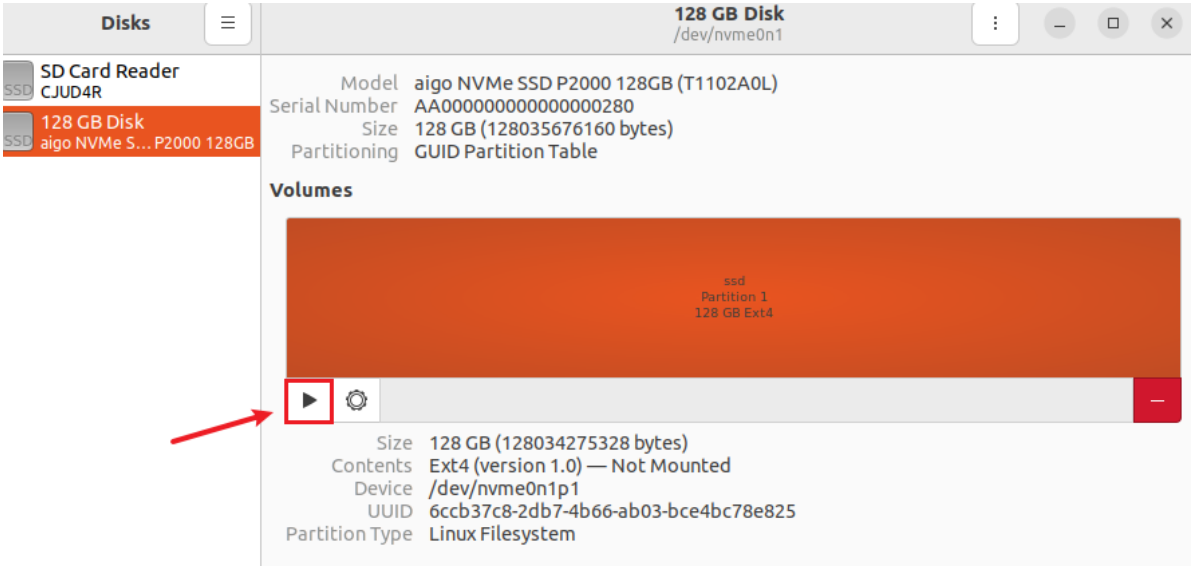
6. You can freely modify the partition size according to your needs. Here, I've defaulted to creating a 128GB partition. Click Next to proceed.



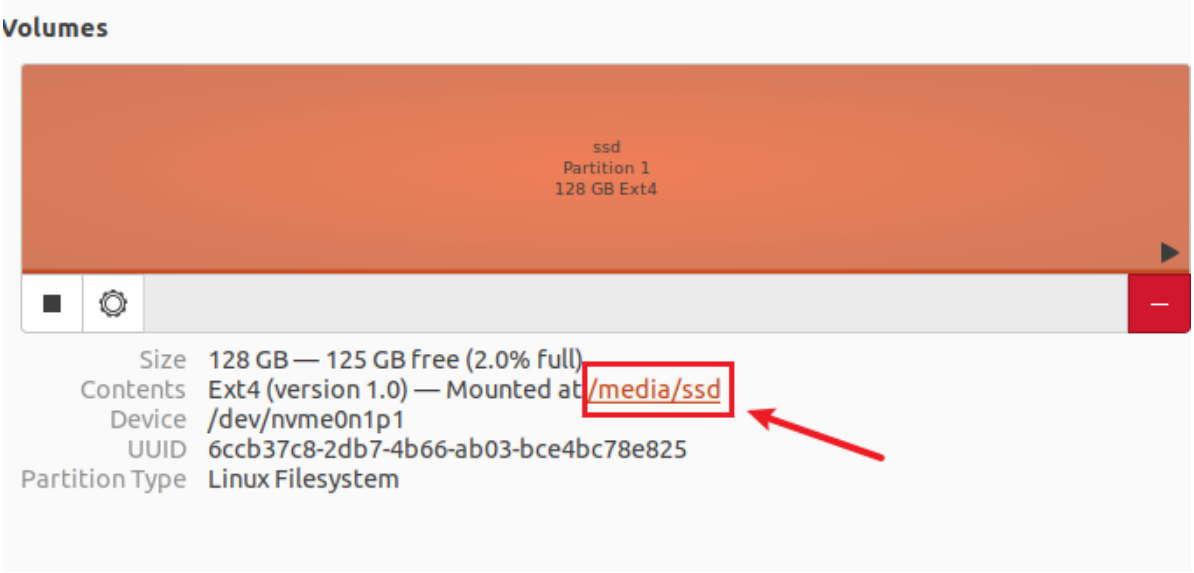
7. Enter the disk name. The disk format must be Ext4; do not change it. Then click Create to create the partition.



8. After successful creation, click the play button to mount the SSD.

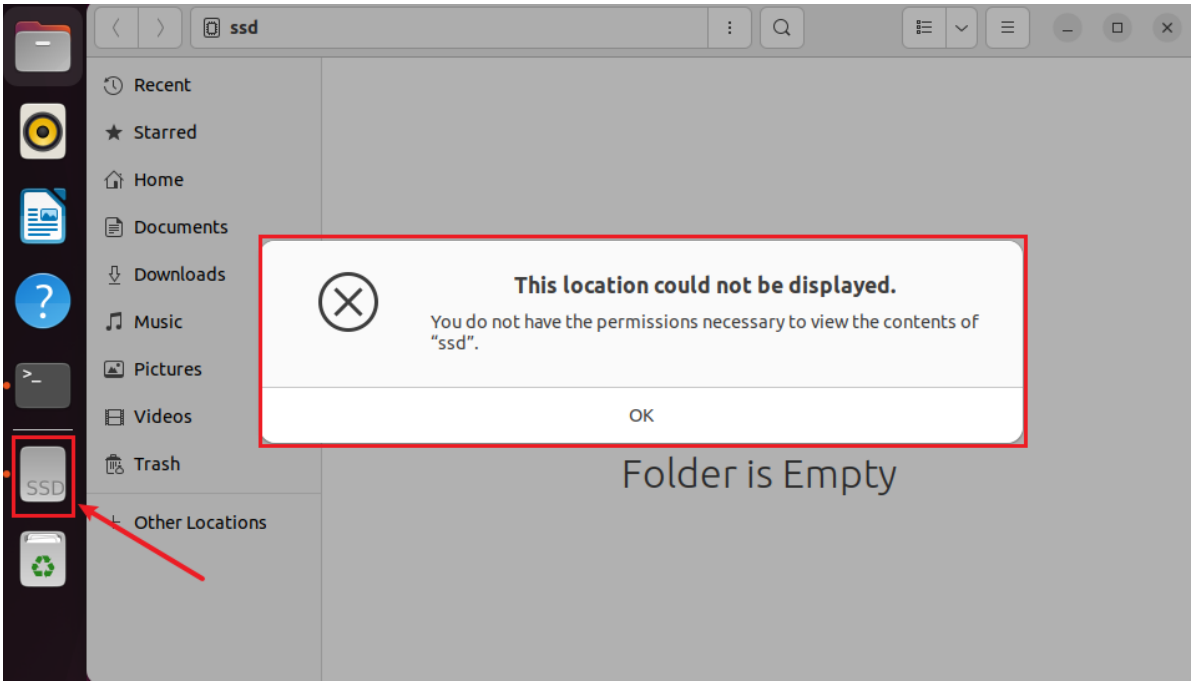


9. After successful mounting, you can see the SSD's mount path. In my case, it's located under /media/ssd.



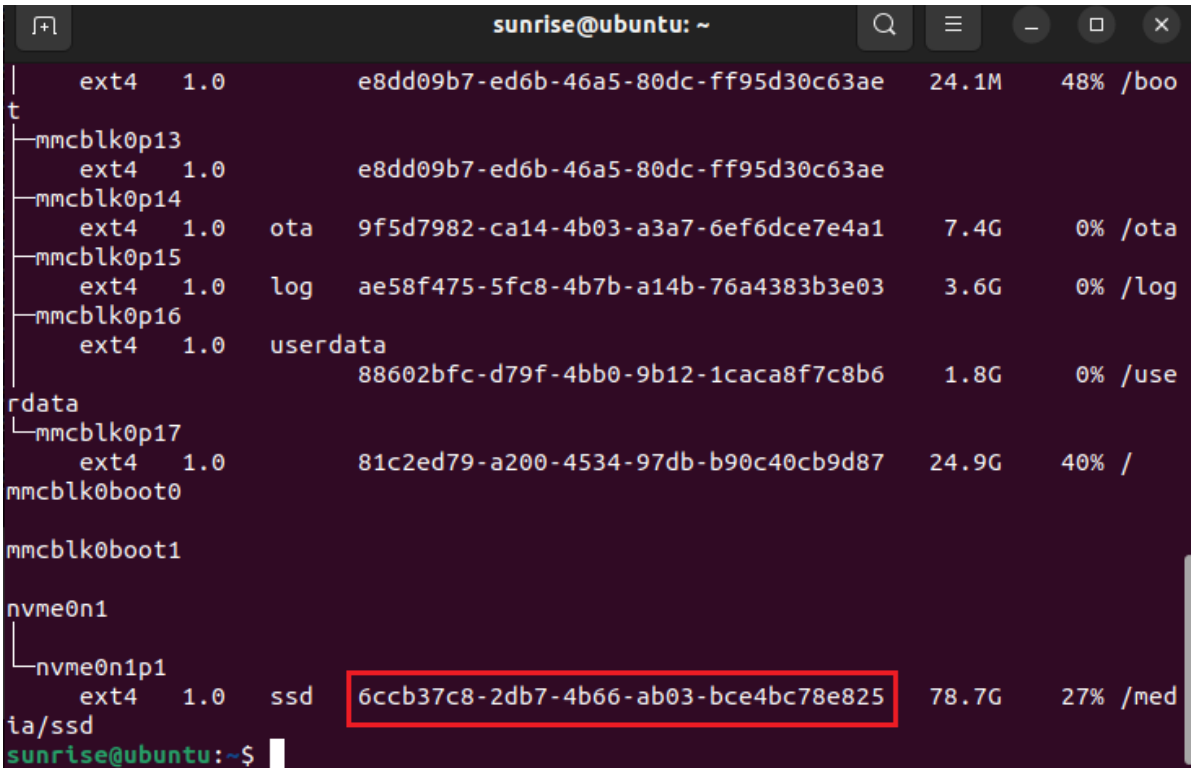
10. Then we can see an SSD icon in the left sidebar of the desktop. We can click to enter the disk's folder, but we'll find that we don't have read or write permissions. At this point, we need to open a terminal and execute the command to use the disk normally:

```
sudo chmod 777 /media/ssd #Here, /media/ssd needs to be modified according to
your SSD's mount path.
```



11. Finally, we need to set the hard drive to be automatically mounted at boot. First, enter the following command to query the SSD's UUID:

```
lsblk -f
```

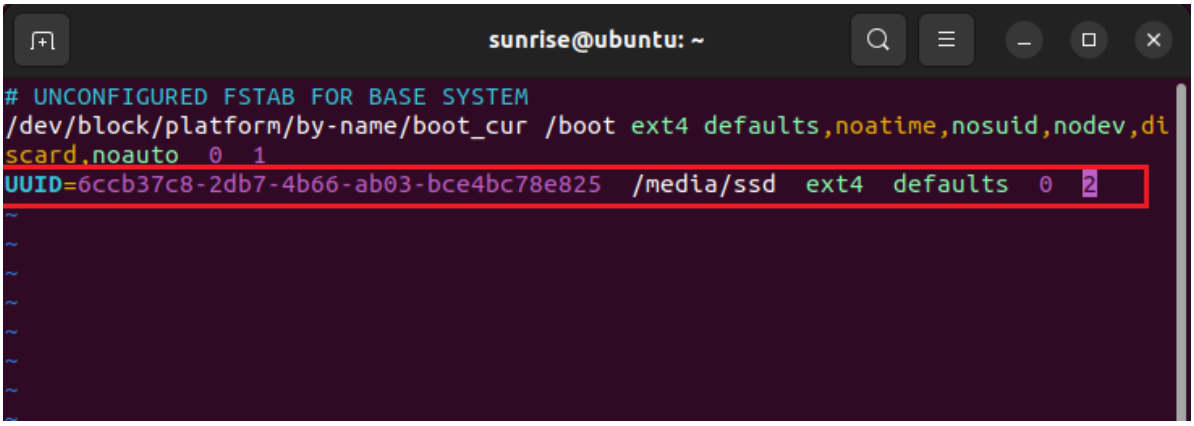


12. Open fstab

```
sudo vim /etc/fstab
```

Add the following text to the end, save and exit:

```
UUID=6ccb37c8-2db7-4b66-ab03-bce4bc78e825 /media/ssd ext4 defaults 0 2
#The content after UUID needs to be modified to your own value, which can be
found in the lsblk -f command above.
```



7. Changing the Ollama Model Download Path (Optional)

On Linux systems, the default download path for Ollama models is:

```
/usr/share/ollama/.ollama/models
```

1. First, stop the Ollama service:

```
systemctl stop ollama.service
```

2. Create a new folder on your SSD to store the models using the following commands:

```
sudo mkdir -p /media/ssd/ollama/models # The /media/ssd here needs to be modified
according to your SSD mount path
```

3. Then, add permissions to the model folder:

```
sudo chown -R sunrise:sunrise /media/ssd/ollama/models
sudo chmod -R 777 /media/ssd/ollama/models
```

4. Next, we open ollama.service

```
sudo vim /etc/systemd/system/ollama.service
```

Add a new line under [Service]: Environment, then save and exit:

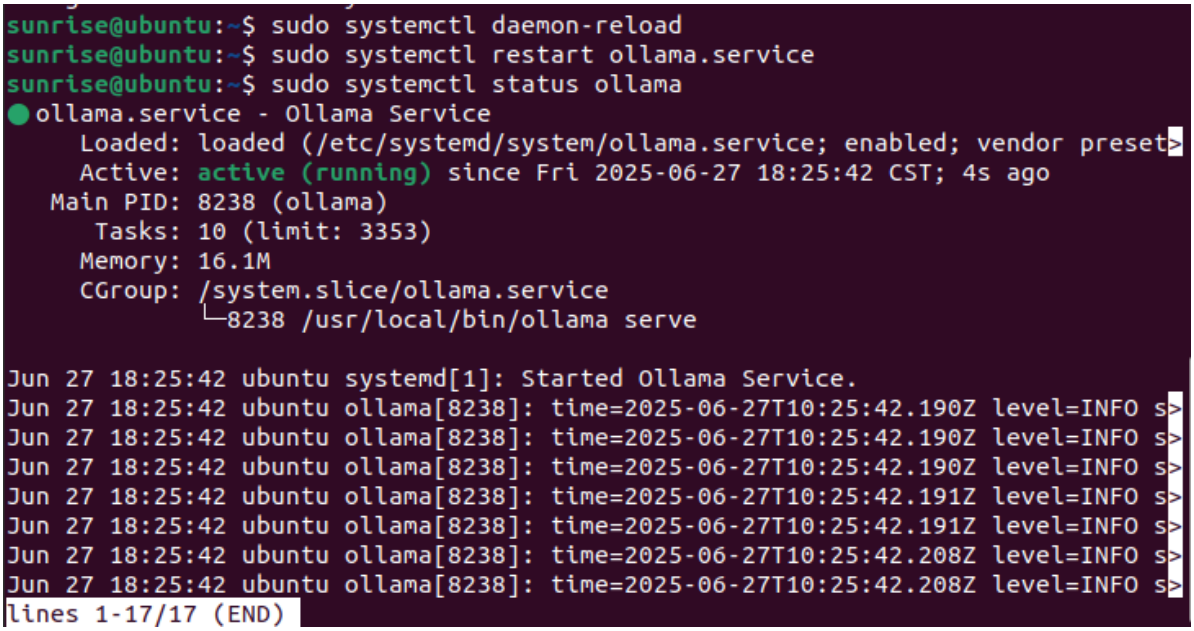
```
Environment="OLLAMA_MODELS=/media/ssd/ollama/models" # Remember to change the
path!!!
```

5. Execute the following command to refresh the configuration:

```
sudo systemctl daemon-reload
```

6. Restart the ollama service and check its status:

```
sudo systemctl restart ollama.service
sudo systemctl status ollama
```



At this point, the ollama model download path has been modified. New models pulled using ollama will be downloaded to the ollama/models folder on the SSD, reducing the space occupied by the eMMC.

8. Setting up the LargeModel Environment (Must Read!!!)

The AI large-model multimodal applications in later chapters of this tutorial all require the LargeModel package to function properly.

1. Locate the largemodel.zip compressed file in the source code folder of your cloud drive. You can use a transfer tool like WinSCP to place the compressed file in your home directory, which is your ~ directory. Then run the command to decompress it:

```
unzip largemodel.zip
```

2. Run the installation script:

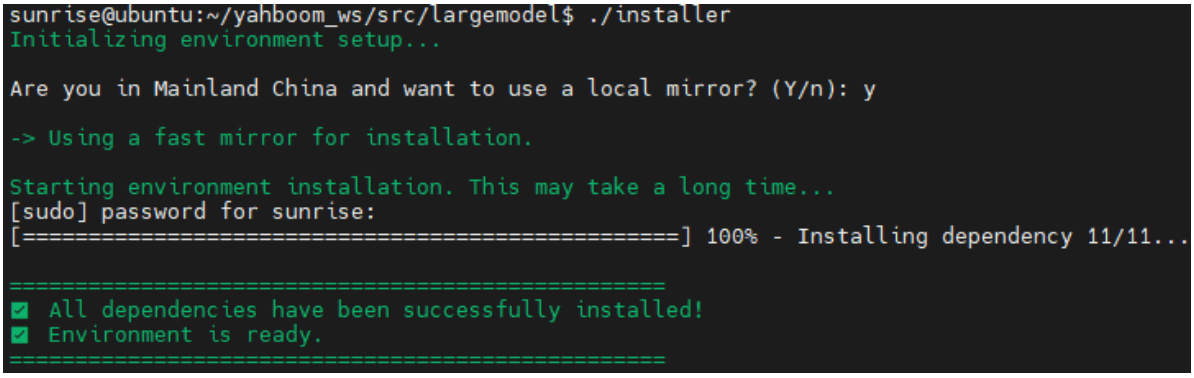
After decompression, first navigate to the LargeModel package:

```
cd ~/yahboom_ws/src/largemodel/
```

Then run the installation script:

```
chmod +x installer #Add script execution permissions
./installer
```

When running the script, it will ask if you are using a mainland China network. If so, enter 'Y' to switch to the Alibaba Cloud mirror source, which will speed up the environment download.



3. Install the colcon tool

```
sudo apt update
sudo apt install python3-colcon-common-extensions
```

4. Add environment variables
   Enter the following commands:

```
echo 'source /opt/ros/humble/setup.bash' >> ~/.bashrc
echo 'source ~/yahboom_ws/install/setup.bash' >> ~/.bashrc
```

Then enter:

```
source ~/.bashrc
```

5. Enter the workspace to compile

```
cd ~/yahboom_ws
colcon build && source ~/yahboom_ws/install/setup.bash
```

6. Start the program:

```
ros2 launch largemodel largemodel_control.launch.py
```

This command is in voice mode and requires the AI large model voice module of the Yaboom RDK S100(P) to be connected. It detects the ttyUSB0 port number by default. For text input mode, please refer to subsequent tutorials.

References

> **Ollama**

Official Website: https://ollama.com/

GitHub: https://github.com/ollama/ollama