Linux环境配置

1. 更新本地源
2. 安装显卡
3. 配置语言环境

   更新系统版本
4. 安装百度输入法

   按照百度输入法说明书安装即可
5. 安装matlab

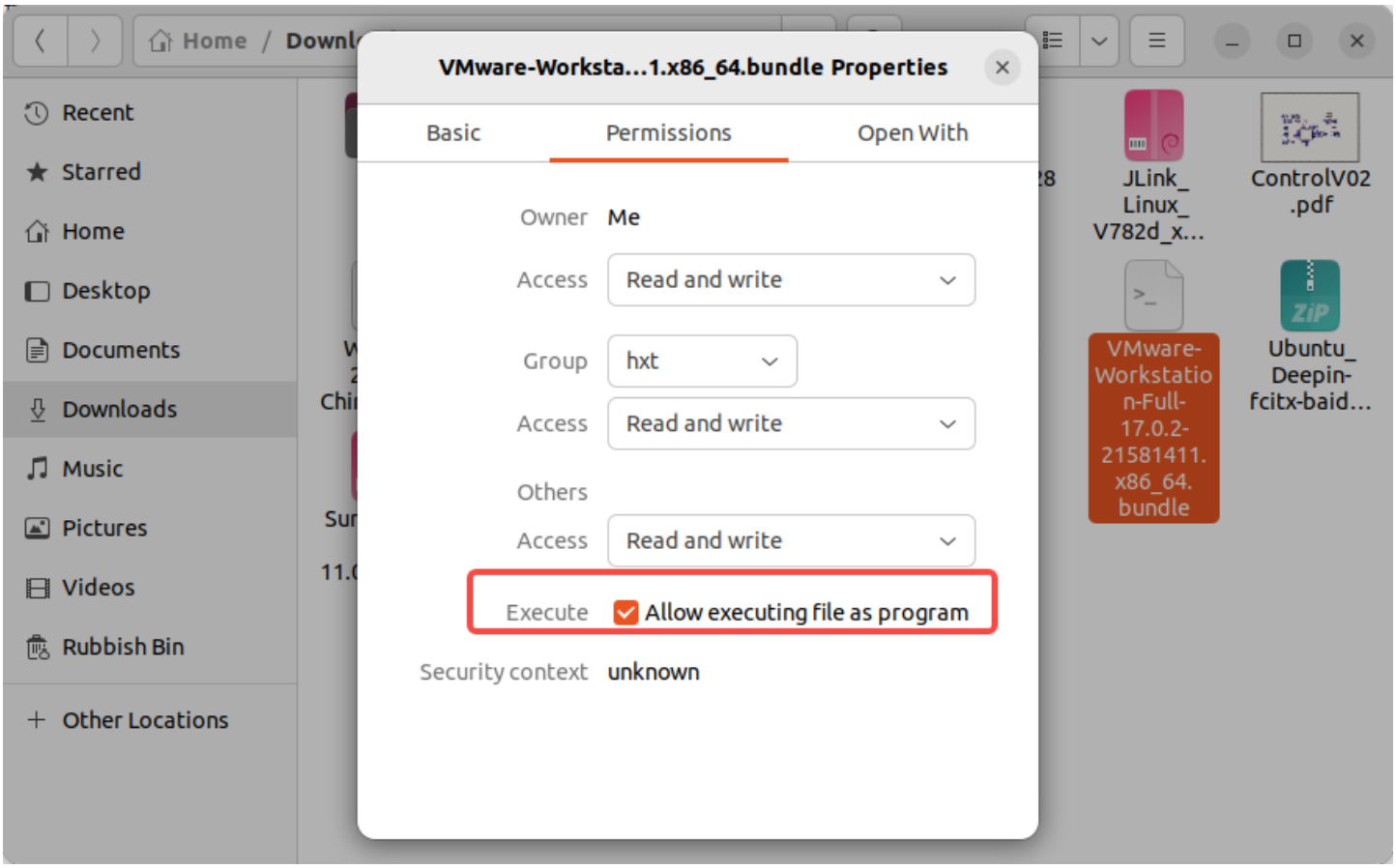
   https://zhuanlan.zhihu.com/p/572662952

matlab窗口设置大小

> 🍞 s = settings;
>
> s.matlab.desktop.DisplayScaleFactor
>
> s.matlab.desktop.DisplayScaleFactor.PersonalValue = 2.0; % 适当调整缩放比例

6. 安装虚拟机器

   配置成可执行文件



7. 安装gcc

   sudo apt-cache search arm-none

   sudo apt install gcc-arm-none-eabi

配置ssh密钥

1. Open a terminal.
2. Run `ssh-keygen -t` followed by the key type and an optional comment. This comment is included in the `.pub` file that's created. You may want to use an email address for the comment.

   For example, for ED25519:

   ```
   ssh-keygen -t ed25519 -C "<comment>"
   ```

   For 2048-bit RSA:

   ```
   ssh-keygen -t rsa -b 2048 -C "<comment>"
   ```

3. Press `Enter`. Output similar to the following is displayed:

   ```
   Generating public/private ed25519 key pair.
   Enter file in which to save the key (/home/user/.ssh/id_ed25519):
   ```

4. Accept the suggested filename and directory, unless you are generating a deploy key or want to save in a specific directory where you store other keys.

   You can also dedicate the SSH key pair to a specific host.

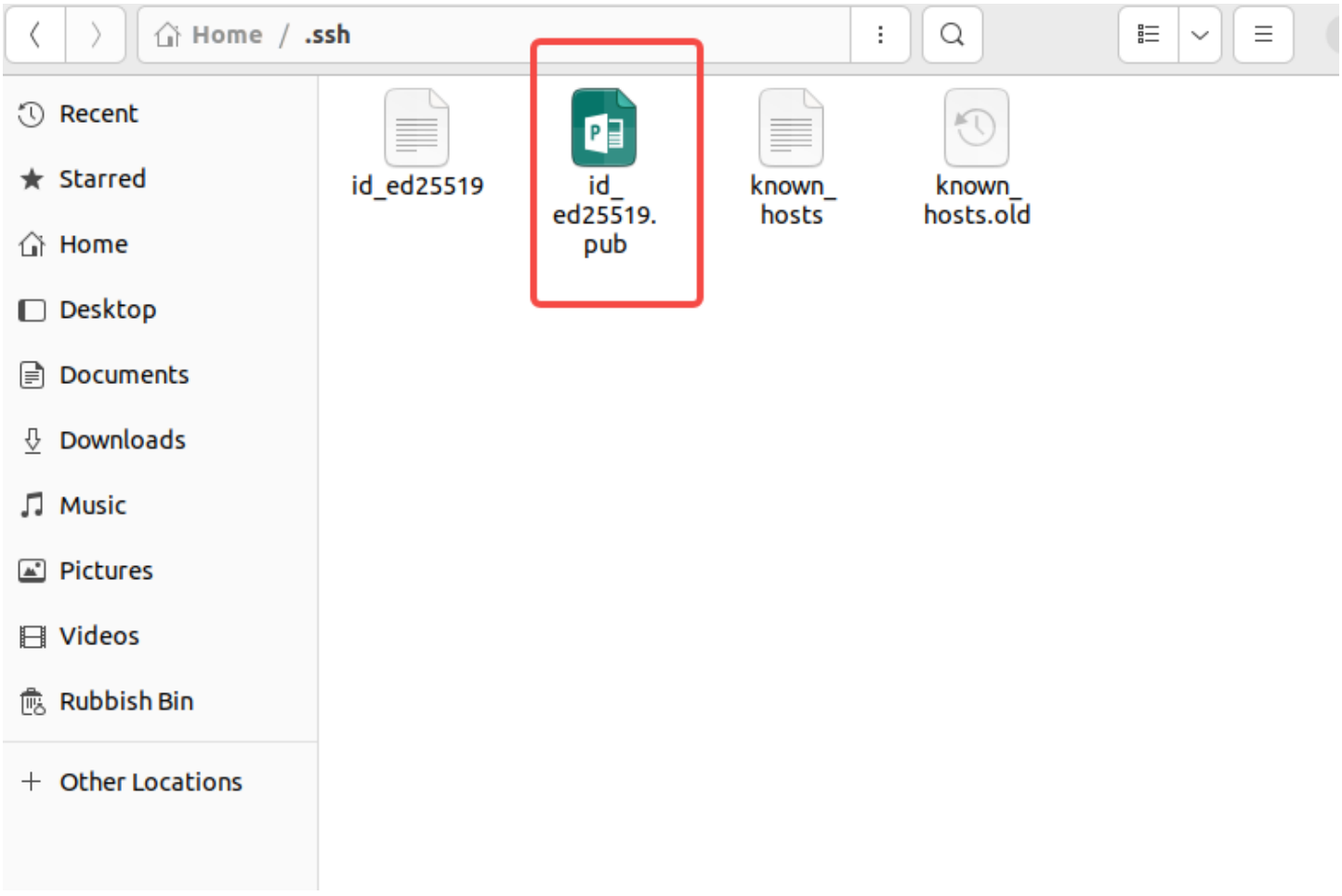
5. Specify a passphrase:

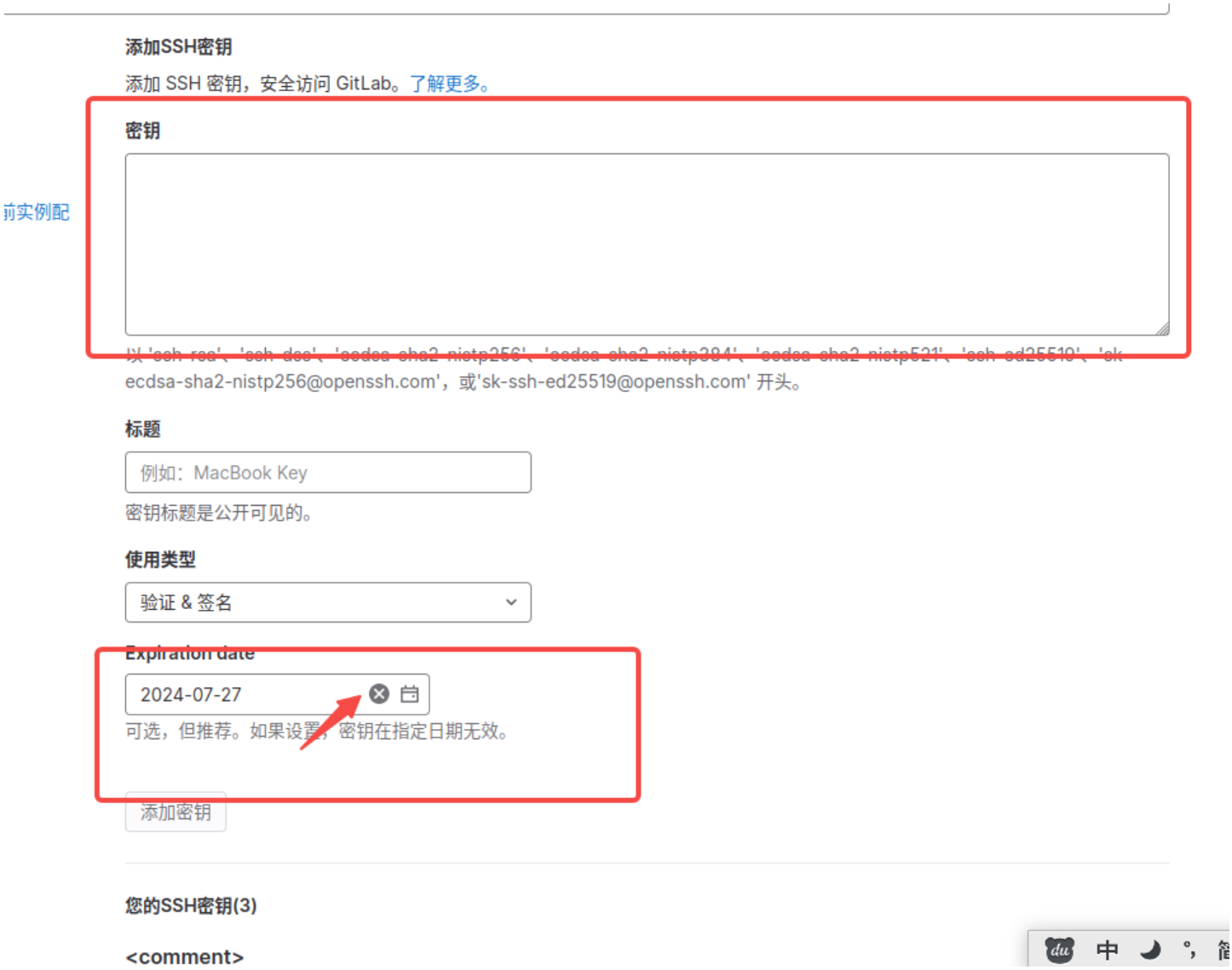
   ```
   Enter passphrase (empty for no passphrase):
   Enter same passphrase again:
   ```

   A confirmation is displayed, including information about where your files are stored.

A public and private key are generated. Add the public SSH key to your GitLab account and keep the private key secure.

将是公钥添加到





8. 配置Python3

   黄锡涛 要在Ubuntu上安装Python，可以按照以下步骤进行操作：

   1. 打开终端。
   2. 更新软件包列表：`sudo apt update`。
   3. 安装Python：`sudo apt install python3`。
   4. 确认安装结果：输入 `python3` 命令后，应显示Python解释器的版本信息。

   另外，您也可以使用以下命令安装pip，这是Python包管理工具：
   `sudo apt install python3-pip`
   安装完成后，您可以使用 `pip3` 命令来安装Python的第三方库。
   希望这对您有所帮助！sudo dpkg -r sunloginclient

   python3 -m pip install --user pipx

9. 配置gitlab环境

If you do not have an existing SSH key pair, generate a new one:

1. Open a terminal.
2. Run `ssh-keygen -t` followed by the key type and an optional comment. This comment is included in the `.pub` file that's created. You may want to use an email address for the comment.

   For example, for ED25519:

   ```
   ssh-keygen -t ed25519 -C "<comment>"
   ```

   For 2048-bit RSA:

   ```
   ssh-keygen -t rsa -b 2048 -C "<comment>"
   ```

3. Press `Enter`. Output similar to the following is displayed:

   ```
   Generating public/private ed25519 key pair.
   Enter file in which to save the key (/home/user/.ssh/id_ed25519):
   ```

4. Accept the suggested filename and directory, unless you are generating a deploy key or want to save in a specific directory where you store other keys.

   You can also dedicate the SSH key pair to a specific host.

5. Specify a passphrase:

   ```
   Enter passphrase (empty for no passphrase):
   Enter same passphrase again:
   ```

   A confirmation is displayed, including information about where your files are stored.

A public and private key are generated. Add the public SSH key to your GitLab account and keep the private key secure.

Create a GPG key

If you don't already have a GPG key, create one:

1. Install GPG for your operating system. If your operating system has `gpg2` installed, replace `gpg` with `gpg2` in the commands on this page.
2. To generate your key pair, run the command appropriate for your version of `gpg`:

   ```
   # Use this command for the default version of GPG, including
   # Gpg4win on Windows, and most macOS versions:
   gpg --gen-key

   # Use this command for versions of GPG later than 2.1.17:
   gpg --full-gen-key
   ```

3. Select the algorithm your key should use, or press `Enter` to select the default option, `RSA and RSA`.
4. Select the key length, in bits. GitLab recommends 4096-bit keys.
5. Specify the validity period of your key. This value is subjective, and the default value is no expiration.
6. To confirm your answers, enter `y`.
7. Enter your name.
8. Enter your email address. It must match a verified email address in your GitLab account.
9. Optional. Enter a comment to display in parentheses after your name.
10. GPG displays the information you've entered so far. Edit the information or press `O` (for `Okay`) to continue.
11. Enter a strong password, then enter it again to confirm it.
12. To list your private GPG key, run this command, replacing `<EMAIL>` with the email address you used when you generated the key:

12. To list your private GPG key, run this command, replacing `<EMAIL>` with the email address you used when you generated the key:

```
gpg --list-secret-keys --keyid-format LONG <EMAIL>
```

13. In the output, identify the `sec` line, and copy the GPG key ID. It begins after the `/` character. In this example, the key ID is `30F2B65B9246B6CA`:

```
sec   rsa4096/30F2B65B9246B6CA 2017-08-18 [SC]
      D5E4F29F3275DC0CDA8FFC8730F2B65B9246B6CA
uid                 [ultimate] Mr. Robot <your_email>
ssb   rsa4096/B7ABC0813E4028C0 2017-08-18 [E]
```

14. To show the associated public key, run this command, replacing `<ID>` with the GPG key ID from the previous step:

```
gpg --armor --export <ID>
```

15. Copy the public key, including the `BEGIN PGP PUBLIC KEY BLOCK` and `END PGP PUBLIC KEY BLOCK` lines. You need this key in the next step.

Associate your GPG key with Git

After you create your GPG key and add it to your account, you must configure Git to use this key:

1. Run this command to list the private GPG key you just created, replacing `<EMAIL>` with the email address for your key:

```
gpg --list-secret-keys --keyid-format LONG <EMAIL>
```

2. Copy the GPG private key ID that starts with `sec`. In this example, the private key ID is `30F2B65B9246B6CA`:

```
sec   rsa4096/30F2B65B9246B6CA 2017-08-18 [SC]
      D5E4F29F3275DC0CDA8FFC8730F2B65B9246B6CA
uid                 [ultimate] Mr. Robot <your_email>
ssb   rsa4096/B7ABC0813E4028C0 2017-08-18 [E]
```

3. Run this command to configure Git to sign your commits with your key, replacing `<KEY ID>` with your GPG key ID:

```
git config --global user.signingkey <KEY ID>
```

这里给出一个配置文件（$HOME/.gitconfig）的样例。

```
[user]
        name = user.name
        email = user.email
        signingkey = user.gpg.seckey
[core]
        editor = vim
[init]
        defaultBranch = main
[color]
        status = auto
        interactive = auto
        branch = auto
        diff = auto
        ui = auto
[commit]
        gpgsign = true
```

可以将其中的`user.name`、`user.email`和`user.gpg.seckey`修改为你自己的，并保存在$HOME/.gitconfig文件中。