Ubuntu 20.04 安装流程

本文以在一台带有GPU的物理机器安装ubuntu-server 20.04为例，安装基本的生产环境，包括后端开发所需的docker与docker-compose，以及深度学习所需的CUDA配置。
本文基本能够涵盖其他安装方式以及其他版本（18.04～24.04）的所遇到的问题，因此，**无论是安装desktop还是server版本、物理机还是虚拟机，都可以参考本文**。
通过虚拟机安装则可以跳过制作启动盘的步骤，直接使用现有的虚拟机工具安装，本文不过多介绍此部分内容。

1. 制作U盘启动盘

开始之前需要准备一个白盘，最好16G以上，至少也要4G，用于烧录系统镜像。

> 注意：提前备份U盘中的重要内容，烧录系统时U盘会格式化。

接着，我们需要下载系统镜像：

- ubuntu.com：官方网站，最稳定，速度也最快，但需要开代理。
- tuna：tsinghua的镜像，没有代理的时候可以使用，但亲测速度较慢。

之后，我们需要下载制作系统启动盘的工具，可根据自己的系统选择：

- Rufus：win系统可用，使用方法参考此篇文章（注意要选择自己需要的系统版本）
- balenaEtcher：mac系统可用，使用简单，根据软件提示操作即可。

完成上述操作之后，我们应该可以得到一个精致的启动盘。

2. 使用U盘安装ubuntu系统

首先，我们将U盘插到服务器上，要保证接口是连通主板的。启动服务器，然后连续按F12进入BIOS，有的主板是F2或者ESC，可根据开机时屏幕的提示按，或者都试一下。
进入BIOS之后，我们要选择启动顺序，将U盘顺序放到最前，或者有的主板可以直接选择使用U盘启动。选择好之后选择“保存并退出”，不出意外的话就可以进入安装流程了。

2.1 安装细节

多说无益，建议直接按照该视频的流程逐步安装，如果有懂的地方可以自己配置（比如硬盘等，每个人应该不太一样）。
tips:

- 如果你在**内网环境**安装时需要指定服务器地址，则先要确认好服务器的固定ip地址，所在网络的掩码以及网关的ip地址。
- 不建议在安装系统时配置apt换源，并且**建议不要急着更新系统依赖**，建议按照默认的源直接跳过更新，等安装完之后再配置apt源和更新系统，避免安装时出错。**如果你换源并更新，而网络又出现了问题，就很有可能导致需要重新安装，请谨慎决策。**

3. 配置系统网络

这一步骤对于使用物理机器十分重要，因为涉及远程使用服务器。虚拟机则会相对简单，可以通过虚拟机工具配置，但这些内容也同样重要，因为后续使用`apt`安装依赖时也要配置好网络才行。

3.1 配置系统网关

建议使用网络配置工具netplan，ubuntu系统安装好时会自带此工具，可以直接使用。ubuntu--server22.04配置文件地址如下：

```
sudo vim /etc/netplan/00-installer-config.yaml
```

配置在此处可以在系统启动时自动`apply`，属于永久配置。如果临时配置，可以使用以下指令：

```
sudo route add -net <目标网络> netmask <子网掩码> gw <网关IP>
```

配置要根据具体的系统和netplan版本而定，以下配置仅供参考：

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: no
      addresses: [192.168.1.100/24]
      routes:
        - to: default # 默认路由，一般用于连接外网
          via: 192.168.1.1 # 网关IP
        - to: 10.0.0.0/8 # 特定网络，一般用于连接内网的其他机器
          via: 192.168.1.2 # 网关IP
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]
```

配置好之后通过以下指令令配置生效：

```
sudo netplan try  # 测试配置（无响应则回滚）
# 或直接应用
sudo netplan apply
```

配置成功之后，可以通过以下指令查看配置情况：

```
ip route show
```

> tips：如果自己在远端（比如ssh）进行网络配置，担心配置错误导致后面连不上服务器，则可以通过`sudo netplan try`来测试配置是否正确，如果在计时器到期前未确认，配置将自动回滚。如果不确定是否正确强烈建议先`try`，笔者曾今因此往返跑了几趟，可谓刻骨铭心。

4. 配置基础工具

如果你已经顺利完成以上网络配置，接下来的操作我们可以移步到自己的电脑上，通过ssh连接操作。

```
ssh <username>@<hostname>
<username>@<hostname>'s password:
```

4.1 apt

在国内首先需要配置apt源，通过以下方式配置：

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

本文以阿里源为例（个人觉得比较快），配置常用的依赖以及docker镜像源，写入以下内容：

```
deb http://mirrors.aliyun.com/ubuntu/ focal main restricted universe
multiverse
deb http://mirrors.aliyun.com/ubuntu/ focal-security main restricted
universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ focal-updates main restricted universe
multiverse
deb http://mirrors.aliyun.com/ubuntu/ focal-backports main restricted
universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ focal-proposed main restricted
universe multiverse
deb [arch=amd64] http://mirrors.aliyun.com/docker-ce/linux/ubuntu focal
```

```
stable
# deb-src [arch=amd64] http://mirrors.aliyun.com/docker-ce/linux/ubuntu
focal stable
```

配置保存子后更新apt源地址，并更新现有依赖：

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

此步骤需要等待一段时间，一般网络只要没有问题，且正确配置了镜像，都不会出问题。
之后，我们来安装一些必要的基础工具：

```
sudo apt install build-essential  # 包括GCC等常见的工具
sudo apt install docker docker-compose  # 后端部署必备
```

4.2 Docker

Docker是生产环境中最重要的工具之一，基本ubuntu服务器都要安装这条🐋，此处对其配置进行相对详细介绍。

> 参考博客：推荐此文章，其涵盖了docker与docker-compose的安装流程，笔者基本就是按照其进行安装的，后来在其基础上又对镜像源进行了一些其他的配置。

4.2.1 Docker

前文虽然提到了可以使用apt安装docker，但那不是佳的配置方法，主要是镜像配置会比较麻烦。笔者建议参考以下方法，首先安装docker依赖：

```
sudo apt-get install ca-certificates curl gnupg lsb-release
```

接着，我们添加Docker官方的GPG密钥，这之后我们才能从dockerhub下载镜像：

```
curl -fsSL http://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg | sudo apt-
key add -
```

然后我们就可以先把镜像源配置上了（没错，还是阿里源）：

```
sudo add-apt-repository "deb [arch=amd64] http://mirrors.aliyun.com/docker-
ce/linux/ubuntu $(lsb_release -cs) stable"
```

完成了上述的操作之后，我们才正式安装docker。具体而言，我们安装的是docker-ce：

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

至此，docker基本就安装完成了。此外，为了避免以后每次使用docker质量需要sudo，我们可以将当前用户加入docker组：

```
sudo usermod -aG docker $USER
```

现在我们可以通过 `systemctl` 验证一下docker的安装情况：

```
sudo systemctl start docker
sudo systemctl status docker
sudo docker run hello-world
```

如果网络不出意外，我们就可以成功up测试镜像了。
也可以将docker加入自启动项，这样reboot之后就不用手动启动docker了：

```
sudo systemctl enable docker
```

此外还有一些工具可以安装（可选）：

```
sudo apt-get -y install apt-transport-https ca-certificates curl software-
properties-common
```

每次安装完新的工具或者对docker进行配置，需要重启docker，指令如下：

```
sudo systemctl restart docker
```

之后，如果还需要更改docker的配置，比如镜像，可以更改 `daemon.json` 配置文件，其路径一般在 `/etc/docker/daemon.json` ，需要sudo权限才能编辑，每次编辑完需要reload daemon文件并重启docker服务，指令如下：

```
sudo systemctl reload docker
sudo systemctl restart docker
```

4.2.2 Docker Compose

比较简单，直接上指令：

```
# github: https://github.com/docker/compose/releases/tag/v2.20.2
# 国内下载地址：https://gitee.com/smilezgy/compose/releases/tag/v2.20.2
sudo curl -SL \
https://github.com/docker/compose/releases/download/v2.20.2/docker-compose-linux-x86_64 \
-o /usr/local/bin/docker-compose

# 或者手动下载，上传到服务器后执行如下指令(use)
# 在 docker-compose-linux-x86_64 文件同一目录下执行
sudo cp docker-compose-linux-x86_64 /usr/local/bin/docker-compose

# 赋予权限
chmod +x /usr/local/bin/docker-compose
```

测试：

```
docker-compose --version
```

至此，docker与docker-compose基本就配置好了，镜像源也基本处于可用的状态。如果出现下载超时的情况，可以参考一下镜像源写入 `/etc/docker/daemon.json` 文件，源配置参考如下：

```
{
    "registry-mirrors": [
        "https://registry.docker-cn.com",
        "https://mirror.ccs.tencentyun.com",
        "https://docker.mirrors.ustc.edu.cn",
        "https://docker.m.daocloud.io",
        "https://docker.imgdb.de",
        "https://docker-0.unsee.tech",
        "https://docker.hlmirror.com",
        "https://docker.1ms.run",
        "https://func.ink",
        "https://lispy.org",
        "https://docker.xiaogenban1993.com"
    ]
}
```

建议试一下哪个能用，然后只写几个能用的，没必要写一大堆源。

5. N卡驱动与CUDA安装

5.1 N卡驱动

本文以1080Ti为例（别问为什么25年还是1080，服务器就长这样了）。
首先我们去官网下载驱动，在官网给出的信息列表根据自己的GPU型号和平台选择后点击搜索，建议将下载搜索到的最新版本。下载好之后获得的是一个`.run`文件，设置为可执行后直接通过`bash`运行：

```
chmod +x NVIDIA-Linux-x86_64-*.run
sudo ./NVIDIA-Linux-x86_64-*.run
```

安装需要一定时间，且安装完之后一般要重启，笔者建议重启前务必测试netplan配置，防止reboot后连不上机器。。。我们可以通过以下指令验证安装是否成功：

```
nvidia-smi
```

tips：这条指令用于查看显卡状态，深度学习时也比较常用，不过一般是用如下方式来监控：

```
watch -n 1 nvidia-smi
```

这条指令会每间隔1秒查看一次显卡状态，适合监控计算核心和显存的利用率。

5.2 CUDA安装

本文以CUDA 12.8为例，目前最新的CUDA版本是12.9，但英伟达宣布在CUDA 12.9开始不再对老型号的显卡进行支持，影响的范围包括经典的GTX系列、TITAN系列等，具体的型号此处不一一列举，建议读者在**安装前事先了解自己的显卡型号，以选择正确的版本**。一般来说，如果你安装了最新版本的显卡驱动，主要需要关注的就是12.8和12.9的区别。
了解到你的机器能支持的CUDA版本之后，直接在浏览器搜索对应版本（本文搜索的是`cuda 12.8`），进入NVIDIA的官方界面，选择操作系统->架构->发布版本->系统版本->安装类型。以本文的情况为例，选择：`Linux->x86_64->Ubuntu->20.04->deb (local)`。前面的选项请根据各自的版本，最后一个建议选择`local`。选好之后，直接复制官网给出的链接安装即可，例如本文的例子：

```
wget
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64/c
uda-ubuntu2004.pin
sudo mv cuda-ubuntu2004.pin /etc/apt/preferences.d/cuda-repository-pin-600
wget
https://developer.download.nvidia.com/compute/cuda/12.8.1/local_installers/c
uda-repo-ubuntu2004-12-8-local_12.8.1-570.124.06-1_amd64.deb
sudo dpkg -i cuda-repo-ubuntu2004-12-8-local_12.8.1-570.124.06-1_amd64.deb
sudo cp /var/cuda-repo-ubuntu2004-12-8-local/cuda-*-keyring.gpg
/usr/share/keyrings/
```

```
sudo apt-get update
sudo apt-get -y install cuda-toolkit-12-8
```

安装需要一段时间，并不短，耐性等待。安装完后

5.3 令Docker容器可以使用CUDA

此处推荐三篇文章：

- NVIDIA Container Toolkit
- 如何在Docker中搭建CUDA & CUDNN 开发环境
- docker-compose设置容器使用GPU
  总结就是要安装NVIDIA Container Toolkit，然后再`docker-compose`的配置文件中配置如下内容：

```
services:
  container-name:  # 替换成你的容器名称
    deploy:
      resources:
        reservations:
          devices:
            - driver: "nvidia"
              count: "all"
              capabilities: [ "gpu" ]
```

6. 深度学习环境

6.1 Miniconda

可以直接上官网输入其给的指令，这里给出我目前（2025-05-14）在官网查到的指令，适用于x64的linux系统：

```
mkdir -p ~/miniconda3
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -
O ~/miniconda3/miniconda.sh
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
rm ~/miniconda3/miniconda.sh
```

但笔者依然建议直接使用官网安装最新的稳定版本，此网站也不需要代理。安装时提示是否要设置自动启动conda（yes/no的选项），笔者建议选yes，这样他会帮我们自动配置`~/.bashrc`文件，比较方便。
如果一切顺利，命令行的左侧应该会出现`(base)`的提示，例如：

```
(base) sjj@pingpong-match-algo:~$
```

安装好之后，可以先配置一下镜像源

```
mkdir ~/.pip
cd ~/.pip
vim pip.conf
```

这将会创建一个全局的pip配置文件，我们写入以下内容（没错，还是aliyun）：

```
[global]
index-url = https://mirrors.aliyun.com/pypi/simple/

[install]
trusted-host = mirrors.aliyun.com
```

以下是一个配置虚拟环境的简单例子：

```
conda create -n torch python=3.9 -y
```

其中`-n torch`指定环境名为torch，`python=3.9`指定python版本，`-y`表示自动确认跳过交互。创建好之后启动环境，然后我们随便安装点依赖：

```
conda activate torch
pip install --upgrade pip
pip install torch torchvision torchaudio jupyter
```

如果安装一切顺利，我们就可以通过以下方式验证conda环境，以及之前安装的cuda是否能够使用了：

```
(torch) sjj@pingpong-match-algo:~$ python
Python 3.9.21 (main, Dec 11 2024, 16:24:11)
[GCC 11.2.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> torch.cuda.is_available()
True
```

如果输出了True，说明我们的torch和cuda环境都已经配置好了。

总结

通过以上步骤，我们便在带有GPU的物理机器上成功安装了ubuntu-server 20.04系统，并完成了后端开发与深度学习所需环境的搭建。无论是制作启动盘、系统安装，还是网络配置、工具安装，每一个环节都至关重要，只要严格按照流程操作，基本能解决常见的安装与配置问题。即便你使用的是其他版本的ubuntu系统，或者在虚拟机、desktop版本中进行安装，本文提供的思路和方法也具有较高的参考价值。希望这份指南能帮助你快速、顺利地搭建起理想的开发与运行环境，开启高效的开发与深度学习之旅。