

虚拟化实验一

戴亮 陈箫彤 孙泰利

一、常规方法

1.1 基础环境配置

1. 使用 vmware 创建一台虚拟机，使用 linux 镜像为 centos7，配置如下：



2. 配置静态 IP 地址：

```
[root@master ~]# cat /etc/sysconfig/network-scripts/ifcfg-ens33
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens33
UUID=f6425d8d-2467-4d7f-b48f-9f6a9c8d260c
DEVICE=ens33
ONBOOT=yes
IPADDR=192.168.1.10
NETMASK=255.255.255.0
GATEWAY=192.168.1.2
DNS1=114.114.114.114
[root@master ~]#
```

3. 服务器关闭防火墙和 selinux:

```
[root@master ~]# systemctl stop firewalld
[root@master ~]# systemctl disable firewalld
Removed symlink /etc/systemd/system/multi-user.target.wants/firewalld.service.
Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
[root@master ~]#
[root@master ~]# vi /etc/selinux/config
[root@master ~]#
[root@master ~]# cat /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of three values:
#   targeted - Targeted processes are protected
```

4. 关闭 ipv6、swap 分区和 NetworkManager:

```
[root@master ~]# vi /etc/sysctl.conf
[root@master ~]# sysctl -p
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
[root@master ~]#

[root@master ~]#
[root@master ~]# systemctl stop NetworkManager
[root@master ~]# systemctl disable NetworkManager
Removed symlink /etc/systemd/system/multi-user.target.wants/NetworkManager.service.
Removed symlink /etc/systemd/system/dbus-org.freedesktop.nm-dispatcher.service.
Removed symlink /etc/systemd/system/network-online.target.wants/NetworkManager-wait-online.service.
[root@master ~]#
```

5. 在三台节点上将桥接的 ipv4 流量传递到 iptables 的链：

```
Last login: Thu Mar 13 03:10:03 2025
[root@master ~]# cat > /etc/sysctl.d/k8s.conf << EOF
> net.bridge.bridge-nf-call-ip6tables = 1
> net.bridge.bridge-nf-call-iptables = 1
> EOF
[root@master ~]#
[root@master ~]#
[root@master ~]# sysctl -p
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
[root@master ~]# clear
```

6. 配置 centos 和 epel 的 yum 源（三台节点都配置）：

```
[root@master ~]# wget -O /etc/yum.repos.d/CentOS-Base.repo https://mirrors.aliyun.com/repo/Centos-7.repo
--2025-03-12 19:30:51-- https://mirrors.aliyun.com/repo/Centos-7.repo
Resolving mirrors.aliyun.com (mirrors.aliyun.com)... 117.21.233.111, 117.21.233.106, 117.21.229.249
Connecting to mirrors.aliyun.com (mirrors.aliyun.com)|117.21.233.111|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2523 (2.5K) [application/octet-stream]
Saving to: '/etc/yum.repos.d/CentOS-Base.repo'

100%[=====>] 2,523      --.-K/s   in 0s

2025-03-12 19:30:51 (1.33 GB/s) - '/etc/yum.repos.d/CentOS-Base.repo' saved [2523/2523]
```

```
[root@master ~]# wget -O /etc/yum.repos.d/epel.repo https://mirrors.aliyun.com/repo/epel-7.repo
--2025-03-12 19:32:42-- https://mirrors.aliyun.com/repo/epel-7.repo
Resolving mirrors.aliyun.com (mirrors.aliyun.com)... 117.68.48.74, 117.68.48.73, 117.68.48.110
Connecting to mirrors.aliyun.com (mirrors.aliyun.com)|117.68.48.74|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 664 [application/octet-stream]
Saving to: '/etc/yum.repos.d/epel.repo'

100%[=====]

2025-03-12 19:32:42 (443 MB/s) - '/etc/yum.repos.d/epel.repo' saved [664/664]
```

```
[root@master ~]# yum clean all && yum repolist
Loaded plugins: fastestmirror
Cleaning repos: base epel extras updates
Cleaning up list of fastest mirrors
Loaded plugins: fastestmirror
Determining fastest mirrors
 * base: mirrors.aliyun.com
 * extras: mirrors.aliyun.com
 * updates: mirrors.aliyun.com
base                                3.6 kB 00:00:00
epel                                4.5 kB 00:00:00
extras                              2.9 kB 00:00:00
updates                             2.9 kB 00:00:00
(1/7): base/7/x86_64/group.gz      153 kB 00:00:00
(2/7): epel/x86_64/group           399 kB 00:00:00
(3/7): extras/7/x86_64/primary.db  253 kB 00:00:00
(4/7): epel/x86_64/updatesinfo     1.3 MB 00:00:00
(5/7): base/7/x86_64/primary.db    6.1 MB 00:00:04
(6/7): epel/x86_64/primary.db      6.7 MB 00:00:07
(7/7): updates/7/x86_64/primary.db 2.7 MB 00:00:24
repo id                             status
base/7/x86_64                      13,791
epel/x86_64                        526
extras/7/x86_64                    6,173
updates/7/x86_64
repolist: 38,502

repo name
CentOS-7 - Base - mirrors.aliyun.com
Extra Packages for Enterprise Linux 7 - x86_64
CentOS-7 - Extras - mirrors.aliyun.com
CentOS-7 - Updates - mirrors.aliyun.com
```

1.2 安装 docker 环境

- (1) 配置 docker 的网络源，并安装 docker：

```
[root@master ~]# yum-config-manager --add-repo https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
Loaded plugins: fastestmirror
adding repo from: https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
grabbing file https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo to /etc/yum.repos.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
[root@master ~]#
[root@master ~]# yum install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
* base: mirrors.aliyun.com
* extras: mirrors.aliyun.com
* updates: mirrors.aliyun.com
docker-ce-stable
```

2. 启动 docker 并设置开机自启动：

```
[root@master ~]# systemctl start docker
[root@master ~]# systemctl enable docker --now
[root@master ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Sat 2025-06-07 16:17:34 CST; 1min 20s ago
     Docs: https://docs.docker.com
    Main PID: 1006 (dockerd)
    CGroup: /system.slice/docker.service
            └─1006 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jun 07 16:17:55 master dockerd[1006]: time="2025-06-07T16:17:55.694765121+08:00" level=info msg="ignoring event" cor
Jun 07 16:17:55 master dockerd[1006]: time="2025-06-07T16:17:55.834723575+08:00" level=info msg="ignoring event" cor
Jun 07 16:17:55 master dockerd[1006]: time="2025-06-07T16:17:55.968578641+08:00" level=info msg="ignoring event" cor
Jun 07 16:17:57 master dockerd[1006]: time="2025-06-07T16:17:57.002563780+08:00" level=info msg="ignoring event" cor
Jun 07 16:17:57 master dockerd[1006]: time="2025-06-07T16:17:57.062802935+08:00" level=info msg="ignoring event" cor
Jun 07 16:17:57 master dockerd[1006]: time="2025-06-07T16:17:57.063868554+08:00" level=info msg="ignoring event" cor
Jun 07 16:17:58 master dockerd[1006]: time="2025-06-07T16:17:58.047304590+08:00" level=info msg="ignoring event" cor
Jun 07 16:17:58 master dockerd[1006]: time="2025-06-07T16:17:58.055166302+08:00" level=info msg="ignoring event" cor
Jun 07 16:17:59 master dockerd[1006]: time="2025-06-07T16:17:59.077089674+08:00" level=info msg="ignoring event" cor
Jun 07 16:17:59 master dockerd[1006]: time="2025-06-07T16:17:59.080002721+08:00" level=info msg="ignoring event" cor
Hint: Some lines were ellipsized, use -l to show in full
```

1.3 安装大模型

(1) 创建 ollama 配置文件：

```
[root@master ~]#
[root@master ~]# mkdir ~/ollama
"models": [
  {
    "name": "llama3",
mkdir: cannot create directory '/root/ollama': File exists
[root@master ~]# cat <<EOF > ~/ollama/config.json
> {
>   "models": [
>     {
>       "name": "llama3",
>       "model": "ollama/llama3:latest"
>     },
>     {
>       "name": "mistral",
>       "model": "ollama/mistral:latest"
>     }
>   ],
>   "system_prompt": "你是一个有帮助的AI助手",
>   "temperature": 0.7
> }
> EOF
[root@master ~]#
```

(2) 运行 ollama 容器：

```
[root@master ~]# docker run -d \
> --name ollama \
> -p 11434:11434 \
> -v ~/ollama:/root/.ollama \
> ollama/ollama
Unable to find image 'ollama/ollama:latest' locally
latest: Pulling from ollama/ollama
13b7e930469f: Pull complete
97ca0261c313: Pull complete
f6a9ed9582e4: Pull complete
f6b71baa717c: Pull complete
Digest: sha256:2ea3b768a8f2dcd4d910f838d79702bb952089414dd578146619c0a939647ac6
Status: Downloaded newer image for ollama/ollama:latest
```

(3) 通过 docker ps 查看容器运行情况：

```
[root@master ~]# docker ps | grep ollama
489ded4fdb16   ollama/ollama   "/bin/ollama serve"   35 seconds ago   Up 34 seconds   0.0.0.0:11434->11434/tcp   ollama
[root@master ~]#
```

(4) 通过浏览器访问 IP+ 11434 验证是否安装成功：

< > ↺ ⏏ ↻  http://192.168.1.10:11434

Ollama is running

(5) 通过进入 ollama 容器，拉取大模型到本地，这里拉取阿里千问的 1.8B 大模型进行测试：

```
[root@master ~]# docker exec ollama ollama pull qwen:1.8B
pulling manifest
pulling 1296b084ed6b: 11%  | 125 MB/1.1 GB 987 KB/s 16m47s
```

```
[root@master ~]# docker exec ollama ollama pull qwen:1.8B
pulling manifest
pulling 1296b084ed6b: 100%  1.1 GB
pulling 41c2cf8c272f: 100%  7.3 KB
pulling 1da0581fd4ce: 100%  130 B
pulling f02dd72bb242: 100%  59 B
pulling 9ece4a97bfb6: 100%  483 B
verifying sha256 digest
writing manifest
success
[root@master ~]#
```

(6) 与大模型进行对话测试：

```
[root@master ~]#  
[root@master ~]# docker exec -it ollama ollama run qwen:1.8B  
>>> 你好，请介绍一下自己  
很荣幸为您介绍我自己，我是一名AI助手，具有以下特点：  
  
1. 语音识别：我可以通过语音与用户进行交互，提供查询、提醒、对话等功能。  
  
2. 自动学习：我能够通过不断的学习和更新知识库，提高自身处理复杂任务的能力。  
  
3. 多语言支持：除了中文以外，我还具备对其他多种语言的支持能力，如英文、法文等。  
  
4. 高效的数据处理和分析：作为AI助手，我具备强大的数据处理和分析功能。我可以从各种来源获取大量实时或历史数据，并通过深度学习、自然语言处理等技术手段对这些数据进行深入挖掘和分析，从而提供更加准确和有价值的数据报告和决策支持。  
  
在与用户的交互过程中，我不仅能快速响应用户的需求，还能提供个性化的服务体验，如根据用户的喜好推荐相关的商品或信息；根据用户的使用习惯提醒相关的任务或活动；或者根据用户的疑问进行详细的解答或解释等。  
  
总之，作为AI助手，我具备强大的数据处理和分析能力，能够从各种来源获取大量实时或历史数据，并通过深度学习、自然语言处理等技术手段对这些数据进行深入挖掘和分析，从而提供更加准确和有价值的数据报告和决策支持。  
  
>>> 你是什么模型？  
我是由阿里云研发的预训练语言模型，名为通义千问（QwenLiU）。通义千问是阿里云自主研发的语言模型系列之一，该系列旨在利用大量的文本数据，进行大规模的语言模型推理和学习，从而实现多种语言的理解、生成和分析等应用。  
  
通义千问的主要优势在于其强大的自然语言处理能力，可以实现对文本数据的深度理解和自动推理。此外，通义千问还具有很强的语言生成能力，可以根据用户的需求和指令，自动生成各种类型的文本内容，如新闻报道、小说故事、科技报告、商业信函等。  
  
总之，通义千问作为阿里云自主研发的语言模型系列之一，以其强大的自然语言处理能力和语言生成能力，可以实现对多种语言的理解、生成和分析等应用。这种强大的技术手段，使得通义千问能够在各种复杂的语言应用场景中，发挥出关键的实用价值。  
  
>>> Send a message (/? for help)
```

二、进阶方法（手动基于 ollama 的镜像构建自带大语言的镜像）

(1)、安装 docker

通过 apt 软件管理工具完成 docker 安装 (已提前换源)

```
master@master: ~  
master@master:~$ sudo apt install docker  
docker-buildx      docker-compose      docker-doc          docker-registry  
docker-clean       docker-compose-v2   docker.io  
master@master:~$ sudo apt install docker.io  
[sudo] master 的密码：  
正在读取软件包列表... 完成  
正在分析软件包的依赖关系树... 完成  
正在读取状态信息... 完成  
将会同时安装下列软件：  
  containerd pigz runc ubuntu-fan  
建议安装：  
  aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite debootstrap  
  docker-buildx docker-compose-v2 docker-doc rinse zfs-fuse | zfsutils  
下列【新】软件包将被安装：  
  containerd docker.io pigz runc ubuntu-fan  
升级了 0 个软件包，新安装了 5 个软件包，要卸载 0 个软件包，有 173 个软件包未被升级。  
需要下载 78.8 MB 的归档。  
解压缩后会消耗 299 MB 的额外空间。  
您希望继续执行吗？ [Y/n] y  
获取:1 http://mirrors.tuna.tsinghua.edu.cn/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]  
获取:2 http://mirrors.tuna.tsinghua.edu.cn/ubuntu noble-updates/main amd64 runc amd64 1.2.5-0ubuntu1-24.04.1 [8,043 kB]
```

dockerhub 被墙了, 对 docker 的仓库进行换源

```
master@master: /etc/docker
正在解压 ubuntu-fan (0.12.16) ...
正在设置 ubuntu-fan (0.12.16) ...
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service -> /usr/lib/systemd/system/ubuntu-fan.service.
正在设置 runc (1.2.5-0ubuntu1~24.04.1) ...
正在设置 pigz (2.8-1) ...
正在设置 containerd (1.7.27-0ubuntu1~24.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service -> /usr/lib/systemd/system/containerd.service.
正在设置 docker.io (27.5.1-0ubuntu3~24.04.2) ...
info: 从 100 到 999 中选择 GID...
info: 正在添加组 "docker" (GID 133)...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service -> /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket -> /usr/lib/systemd/system/docker.socket.
正在处理用于 man-db (2.12.0-4build2) 的触发器 ...
master@master:~$ docker --version
Docker version 27.5.1, build 27.5.1-0ubuntu3~24.04.2
master@master:~$ sudo mkdir -p /etc/docker
master@master:~$ cd /etc/docker
master@master:/etc/docker$ vim daemon.json
master@master:/etc/docker$ sudo vim daemon.json
master@master:/etc/docker$
```

```
master@master: /etc/docker
"registry-mirrors":[
  "https://docker.hpcloud.cloud",
  "https://docker.m.daocloud.io",
  "https://docker.unsee.tech",
  "https://docker.ipanel.live",
  "https://mirrors.ustc.edu.cn",
  "https://docker.chenby.cn",
  "https://mirror.azure.cn",
  "https://dockerpull.org",
  "https://dockerhub.icu",
  "https://hub.rat.dev",
  "https://proxy.ipanel.live",
  "https://docker.ipanel.top",
  "https://docker.m.daocloud.io",
  "https://docker.jnc.fun",
  "https://docker.ketches.cn"
]
```

(2)、构建镜像

Dockehub 中的大模型镜像需要使用 Docker Model Runner 插件或其他第三方的容器运行。由于 Docker Model Runner 还在测试阶段，所以不考虑用 dockerhub 中提供的大模型镜像。

我们采用通过构建一个基于 ollama 且运行大模型的镜像，来完成实验。

先基于 ollama 镜像，配置一个包含 qwen:0.5b 大模型的镜像，编写其 dockerfile 文件，以及启动脚本。

(3)、创建并运行容器

使用前面构建好的镜像创建容器并运行

```
docker@docker:~$ docker run -it -p 11434:11434 ollama-tinyllama
Successfully tagged ollama-tinyllama:latest
docker@docker:~$ docker run -it -p 11434:11434 ollama-tinyllama
Couldn't find '/root/.ollama/id_ed25519'. Generating new private key.
Your new public key is:
ssh-ed25519 AAAAC3NzaC1lZD11NTE5AAAAIckJSv+polDqTDleWqsP36ophkYxamSfdRw2195chB8
time=2025-06-10T10:39:47.316Z level=INFO source=routes.go:1234 msg="server config" env="map[CUDA_VISIBLE_DEVICES: GPU_DEVICE_ORDINAL: HIP_VISIBLE_DEVICES: HSA_OVERRIDE_GFX_VERSION: HTTPS_PROXY: HTTP_PROXY: NO_PROXY: OLLAMA_CONTEXT_LENGTH:4096 OLLAMA_DEBUG:INFO OLLAMA_FLASH_ATTENTION:false OLLAMA_GPU_OVERHEAD:0 OLLAMA_HOST:http://0.0.0.0:11434 OLLAMA_INTEL_GPU:false OLLAMA_KEEP_ALIVE:5m0s OLLAMA_KV_CACHE_TYPE: OLLAMA_LLM_LIBRARY: OLLAMA_LOAD_TIMEOUT:5m0s OLLAMA_MAX_LOADED_MODELS:0 OLLAMA_MAX_QUEUE:512 OLLAMA_MODELS:/root/.ollama/models OLLAMA_MULTIUSER_CACHE:false OLLAMA_NEW_ENGINE:false OLLAMA_NOHISTORY:false OLLAMA_NOPRUNE:false OLLAMA_NUM_PARALLEL:0 OLLAMA_ORIGINS:[http://localhost https://localhost http://localhost:* https://localhost:* http://127.0.0.1 https://127.0.0.1 http://127.0.0.1:* https://127.0.0.1:* http://0.0.0.0 https://0.0.0.0 http://0.0.0.0:* https://0.0.0.0:* app://* file://* tauri://* vscode-webview://* vscode-file://*] OLLAMA_SCHED_SPREAD:false ROCR_VISIBLE_DEVICES: http_proxy: https_proxy: no_proxy:]"
time=2025-06-10T10:39:47.317Z level=INFO source=images.go:479 msg="total blobs: 0"
time=2025-06-10T10:39:47.317Z level=INFO source=images.go:486 msg="total unused blobs removed: 0"
```

容器顺利启动, 正常运行

(4)、测试与大模型交互

接下来测试与大模型的简单交互。为了方便, 这边直接使用 js 请求 ollama 接口。

```
Object { id: "chatcmpl-566", object: "chat.completion", created: 1749552956, model: "tinyllama", system_fingerprint: "fp_ollama", choices: (1) [~], usage: {~} }
  choices: Array (1) [~]
    0: Object { index: 0, message: {~}, finish_reason: "stop" }
      finish_reason: "stop"
      index: 0
      message: Object { role: "assistant", content: "That information is proprietary and not available through human contact or interaction. The only public-facing likeness of me is my brand image and social media profiles, which are subject to change based on my interactions with customers. I have no physical self or personality traits that would allow for identification by humans." }
        content: "That information is proprietary and not available through human contact or interaction. The only public-facing likeness of me is my brand image and social media profiles, which are subject to change based on my interactions with customers. I have no physical self or personality traits that would allow for identification by humans."
        role: "assistant"
        <prototype>: Object {~}
      <prototype>: Object {~}
    length: 1
    <prototype>: Array []
  created: 1749552956
  id: "chatcmpl-566"
```

没有问题, 容器中的大模型运行正常, 完成实验.