

**《****通用工具操作实训》**

**实训报告**

|  |  |
| --- | --- |
| **姓 名:** | **张舒婷** |
| **学 号:** | **2409050079** |
| **专 业­­­­­­­­­­­­­­­­­­­:** | **网络空间安全** |
| **班 级:** | **网安2402** |
| **指导教师:** | **王海东** |
| **职 称­­­­­­­­­­­­­­­­­­­:** | **讲师** |

**前沿交叉学院**

**2025年5月**

**《通用工具操作实训》评审表**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 |  | |  |  | | | | 学 号 | | |  | | | |
| 专业班级 | | |  | | | |
| 实训内容 |  | | | | | | | | | | | | | |
|
| 评  审  意  见 | 项目 | | 具体要求 | | | 单项分值 | 分项评分分数参考区间 | | | | | | | 得分 |
| A | | B | C | | D | E |
| 平时表现 | | 学习态度认真，自主学习相关方法、总结归纳相关技术的能力优秀。积极与团队成员合作，积极参与讨论分析，团队关系融洽，能按时完成团队分配的工作。明确团队角色和分工，积极与团队成员合作，团队工作进展顺利。 | | | 20 | 18 | | 16 | 14 | | 12 | <12 |  |
| 实操成绩 | | Git 版本控制（15 分）：基础操作（5 分）分支管理（5 分）远程协作（5 分）；  Docker（15 分） ：镜像操作（5 分），容器管理（5 分），数据与编排（5 分）；  攻防工具选修（20 分，任选1项完成）。 | | | 50 | 45 | | 40 | 35 | | 30 | <30 |  |
| 文档质量 | | 文档规范，结构合理，文献工作量饱满，格式正确。 | | | 10 | 9 | | 8 | 7 | | 6 | <6 |  |
| 实验原理正确，步骤完整，实验结果充分，结论有效。方案合理可行，效果好，有创新意识。工作量饱满，完成质量高。 | | | 20 | 18 | | 16 | 14 | | 12 | <12 |  |
| 评审成绩：  □优秀（100-90） □良好（89-80） □中等（79-70） □及格（69-60） □不及格（<60） | | | | | | | | | | | | | |
| 指导教师签名 | |  | | 职称 |  | 时间 | | | 年 月 日 | | | | | |

**目录**

[摘要 4](#_Toc202436000)

[第1章 绪论 5](#_Toc202436001)

[1.1 实训背景与意义 5](#_Toc202436002)

[1.2 实训内容 5](#_Toc202436003)

[第2章 git 5](#_Toc202436004)

[2.1基本原理 5](#_Toc202436005)

[2.2实验步骤 6](#_Toc202436006)

[2.3实验结果与分析 9](#_Toc202436007)

[第3章 Docker 9](#_Toc202436008)

[3.1基本原理 9](#_Toc202436009)

[3.2实验步骤 9](#_Toc202436010)

[3.3实验结果与分析 11](#_Toc202436011)

[第4章 Wireshark 12](#_Toc202436012)

[4.1基本原理 12](#_Toc202436013)

[4.2实验步骤 12](#_Toc202436014)

[4.3实验结果与分析 15](#_Toc202436015)

[**第5章 总结** 16](#_Toc202436016)

**摘要**

本实验报告旨在总结和记录在通用工具操作实训中的学习内容和实践经验。实训内容主要覆盖开发和网络攻防通用工具两个方向，包括Git的掌握、Docker的掌握、网络协议分析工具（如Wireshark）的使用。通过本次实训掌握了这些工具的基本操作和高级功能，提升了在实际项目中的应用能力，为后续的软件开发和网络安全工作奠定了坚实基础。

关键词：Git，Docker, Wiresh ，实训

# 第1章 绪论

1.1 实训背景与意义

随着信息技术的快速发展，软件开发、网络通信和安全防护等领域对高效工具的需求日益增长。掌握通用工具的操作技能不仅能提高工作效率，还能为复杂问题的解决提供技术支持。本次实训旨在通过实际操作，帮助学生熟练掌握Git、Docker和Wireshark等工具的使用方法，培养其在实际项目中的工具应用能力和问题解决能力。这些技能对于从事软件开发、网络安全和系统维护等职业的学生具有重要的实践意义。

1.2 实训内容

本次实训主要围绕现代软件开发中的关键技术展开，重点包括Git版本控制,Docker容器化技术的学习与实践和wireshark的抓包操作的掌握。在Git部分，实训内容涵盖版本管理的基本原理、常用命令操作、分支管理策略以及团队协作开发流程，旨在帮助掌握高效的代码版本控制方法。Docker部分则涉及容器化技术的核心概念、镜像构建与管理、容器部署及网络配置，通过实际案例练习，培养应用容器化的部署能力。此外，实训还深入网络分析领域，借助Wireshark工具开展网络抓包实验，包括协议解析、流量监控及故障诊断等实践环节，使学员能够全面理解网络通信机制。实训采用理论讲解与动手实操相结合的方式，通过阶段性任务和团队协作项目，逐步提升学员的技术实践能力和问题解决能力，最终完成可交付的代码仓库、容器化应用及完整的实训报告。

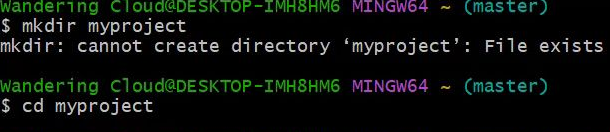
第2章 git

2.1基本原理

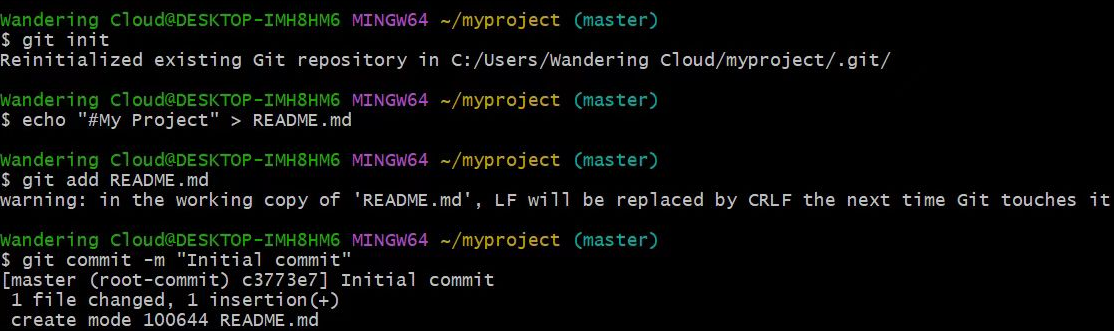
在现代软件开发的复杂协作场景中，Git 作为一款卓越的分布式版本控制系统，其核心竞争力源于对工作目录、暂存区、本地仓库及远程仓库这四个关键区域的精妙设计与协同运作。这四个区域相互衔接又各司其职，共同构建起一套高效、灵活且可靠的代码管理体系，为开发者提供了从个体编辑到团队协作的全流程支持。其中，工作目录作为开发者直接接触的 "前线阵地"，是代码文件实际存在并进行编辑的物理空间。在这里，每一个文件的创建、修改或删除都会被 Git 实时感知而已被管理的文件若发生变动，则会标记为 "已修改"，清晰呈现与最近一次提交版本的差异；暂存区扮演着 "筛选与准备" 的过渡角色，它像一个精心设计的缓冲区，承接工作目录中的变更并为提交操作做好铺垫。作为过渡区域，它通过 git add命令收纳工作目录的变更，为提交做准备；本地仓库存储完整版本历史，git commit命令将暂存区内容固化为版本记录；而远程仓库作为团队协作的 "中央枢纽"，通常部署在服务器上（如 GitHub、GitLab 等平台），是多人共享代码、同步进度的核心载体。它用于多人协作时的代码同步，借助git push、git pull等命令，实现本地与远程仓库的交互，支持并行开发与版本回溯 。

2.2实验步骤

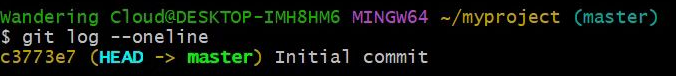
创建一个新的文件夹并进入：

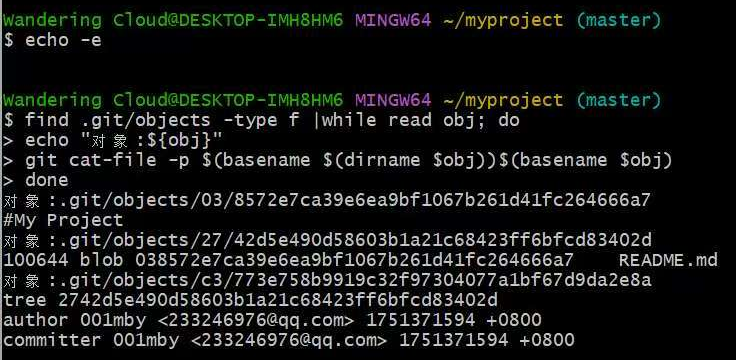


初始化git仓库，使用git add将其加入暂存区，再通过git commit提交到本地仓库：

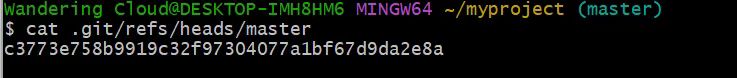


用git log查看提交历史





执行git commit时，master分支如何更新

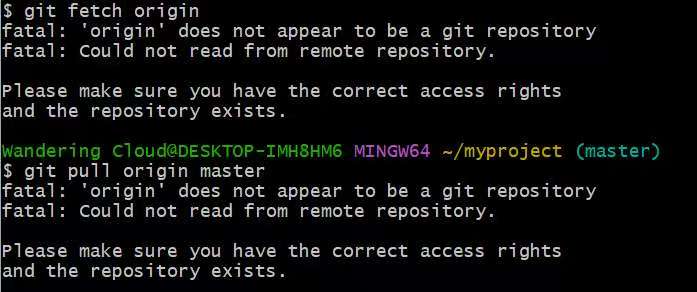


通过git remote命令添加远程仓库，git remote -v查看参考信息



首先查看当前分支（确认是否是master）将本地master分支推送到远程origin仓库 再用git push命令将本地版本库的提交推

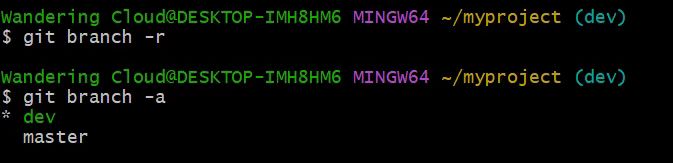
完成从本地推送到远程，从远程拉取更新的完整流程



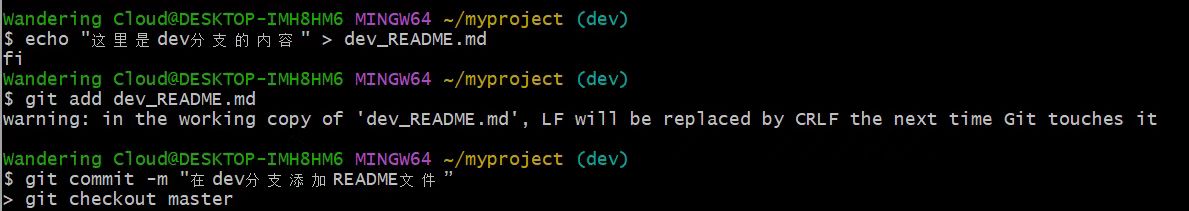
URL分别为HTTPS和SSH两种格式的远程仓库

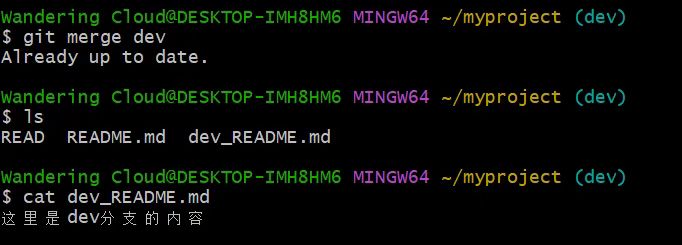
（3）

使用git checkout -b dev创建并切换到新分支dev,用git branch查看当前分支（带\*的为当前分支）

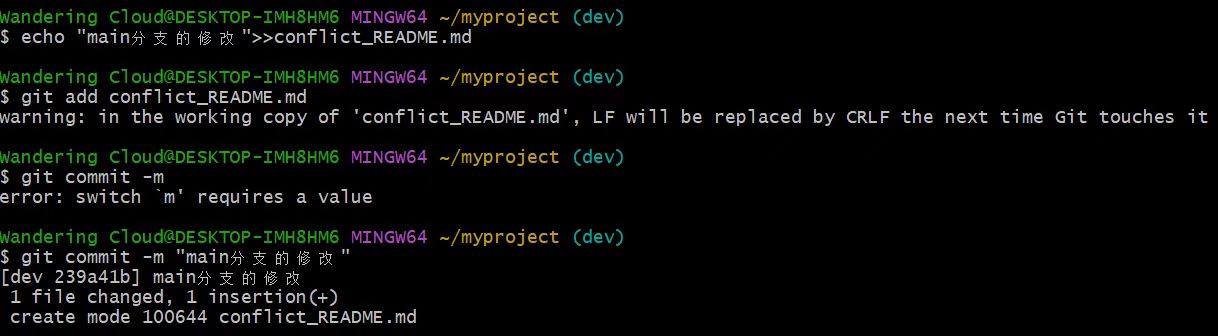
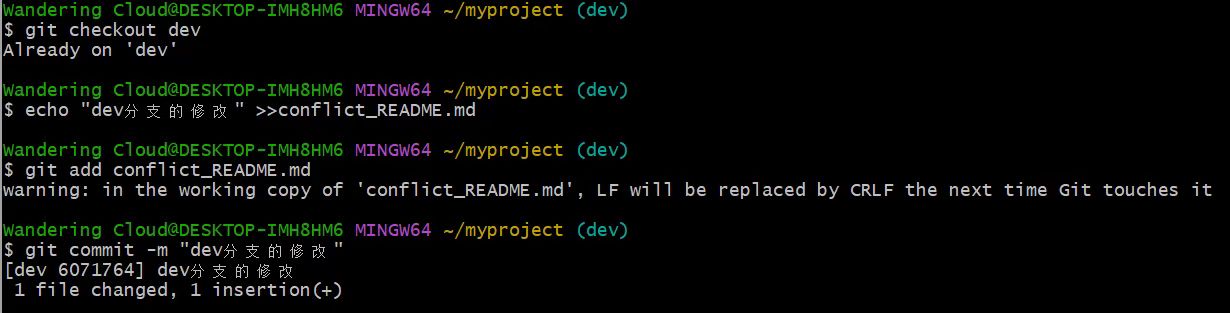
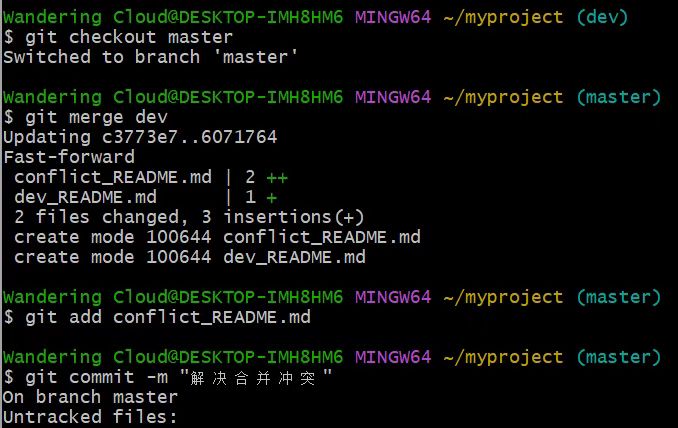
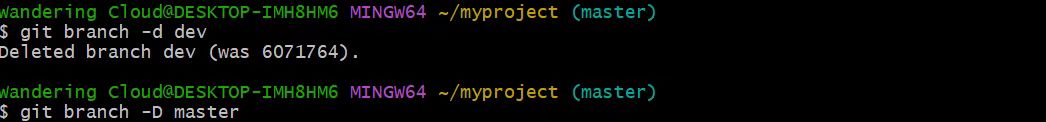


在dev分支添加文件



用git merge命令将dev分支合并到main分支，并查看文件

模拟并解决合并冲突。在main中修改文件，切换到dev分支并修改同一文件，切换回main分支尝试合并，出现冲突，手动编辑conflict\_README.md解决冲突

2.3实验结果与分析

通过对Git实操，我清晰看到工作目录、暂存区、本地仓库的交互。新建文件是“未跟踪”状态，采用git add能把修改放到暂存区，再执行git commit -m提交到本地仓库，借助git log能查看提交历史。清晰的展现了文件在三个区域的状态变化。在远程仓库交互时，分别采用 git push把本地代码推上去，git pull 拉取更新和git remote配置、查看远程仓库。在分支管理里，使用git checkout -b 新建并切换分支，不同分支开发互不影响。通过git branch系列命令查看分支，在多分支并行开发，体验切换时工作目录变化。合并分支用git merge，遇到冲突就手动改文件，再git add ，git commit 解决。Git 让我学会版本控制，团队开发时能管好代码，还能回退历史版本。

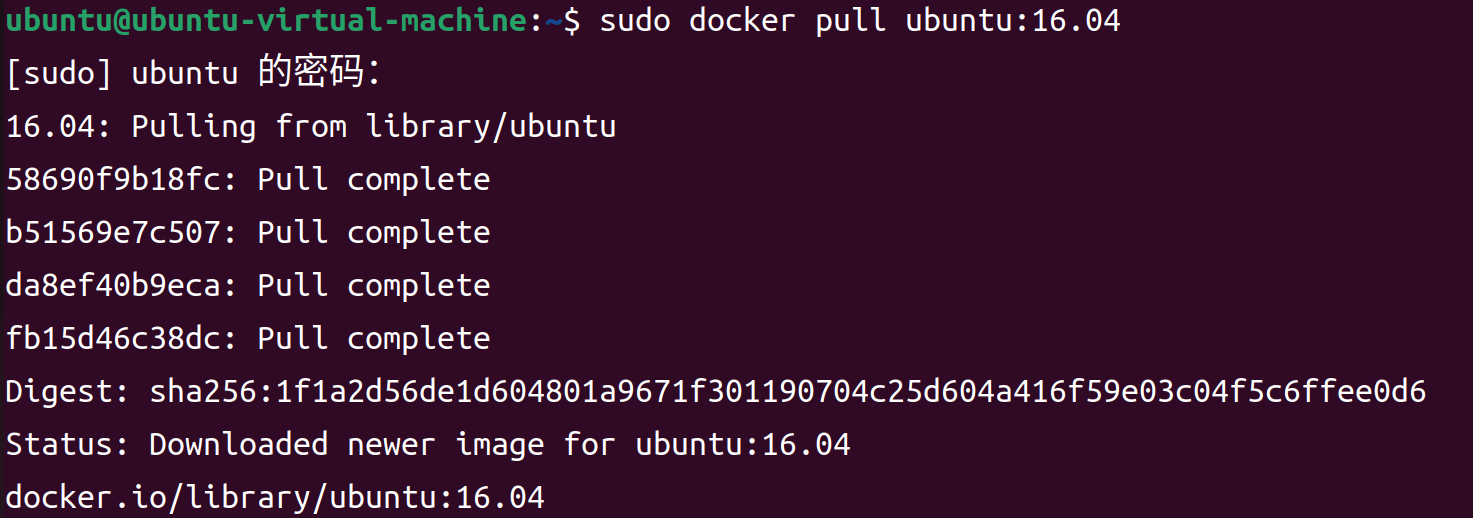
第3章 Docker

3.1基本原理

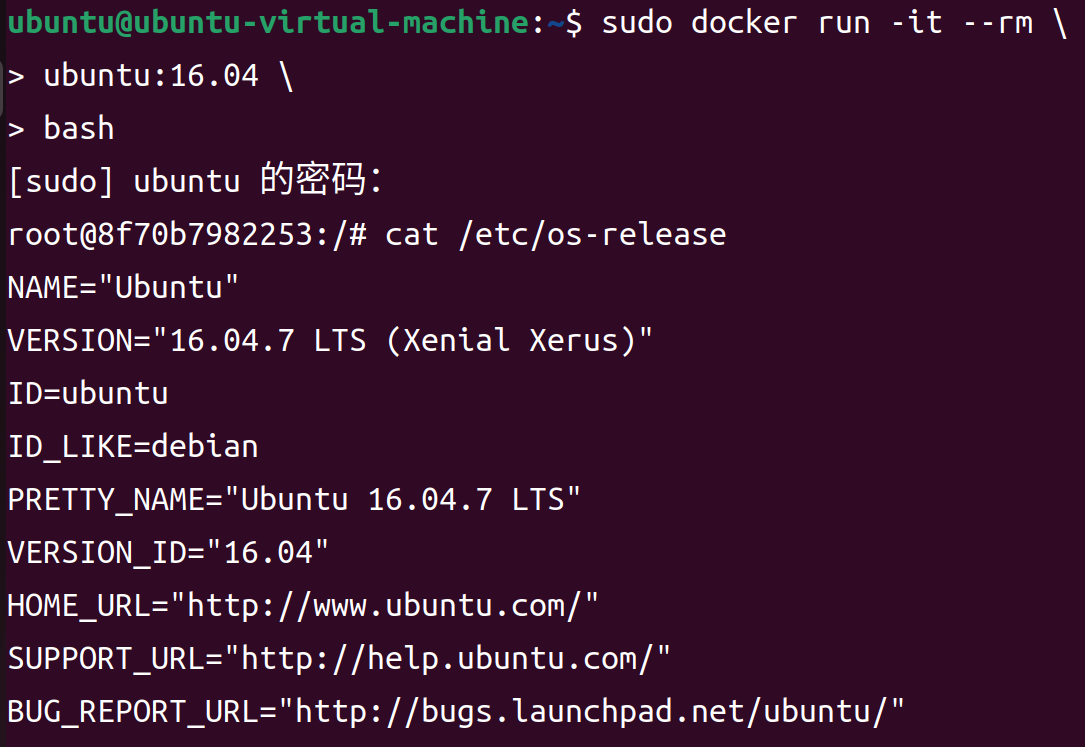
Docker 是基于操作系统级虚拟化的容器化技术，核心靠镜像、容器、仓库三大组件协同工作。​其中，镜像为只读模板，含应用及所有依赖，采用分层存储，可增量构建和共享复用，每层对应 Dockerfile 一条指令。​容器是镜像的运行实例，通过添加可写层实现修改隔离，用写时复制机制保证轻量，秒级启动。借助 namespace 隔离资源，cgroups 限制 CPU、内存等，实现多容器并行不干扰。​仓库用于集中存储和分发镜像，支持增量同步，方便团队协作。​其底层整合 Linux 的 namespace、cgroups 和 UnionFS 技术，实现隔离、资源控制和高效存储，让应用 “一次构建，到处运行”。

3.2实验步骤

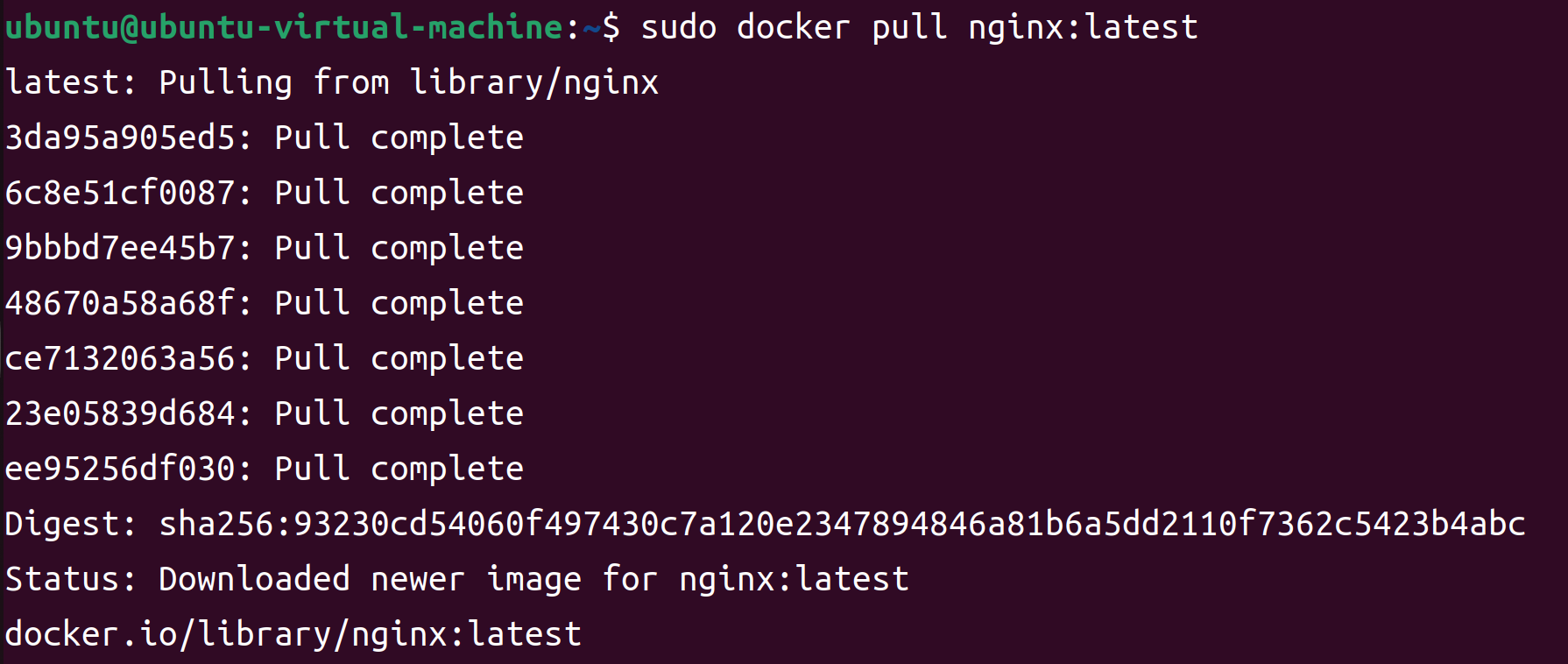
通过使用docker pull,拉取了一个镜像名为ubuntu:16:04

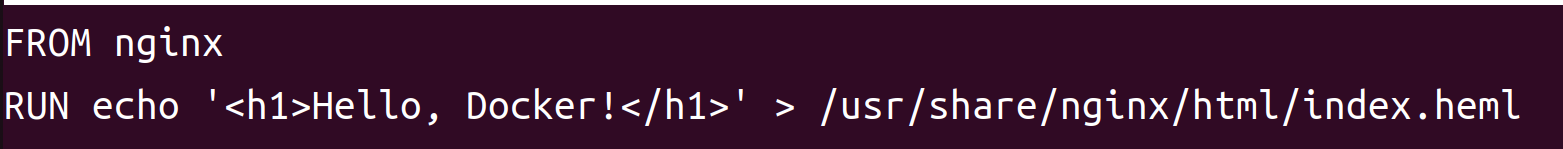
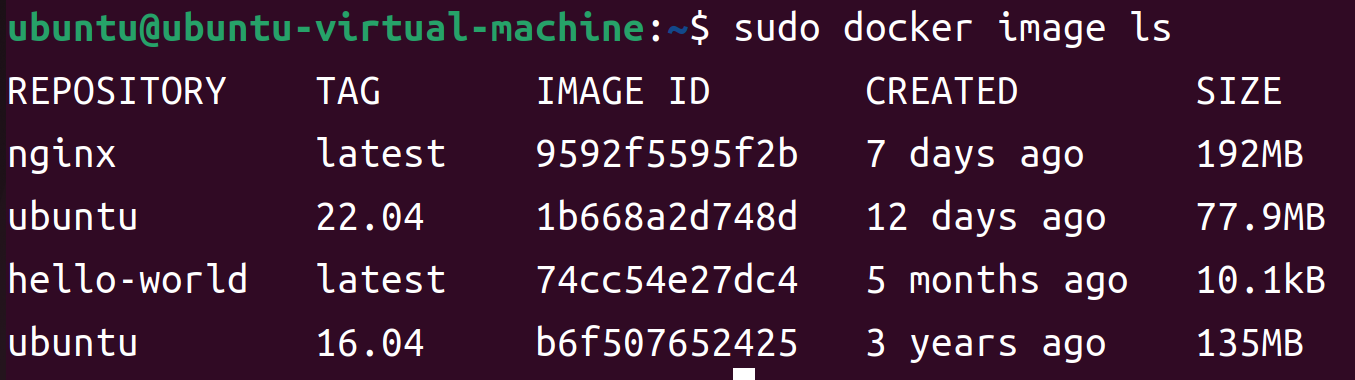


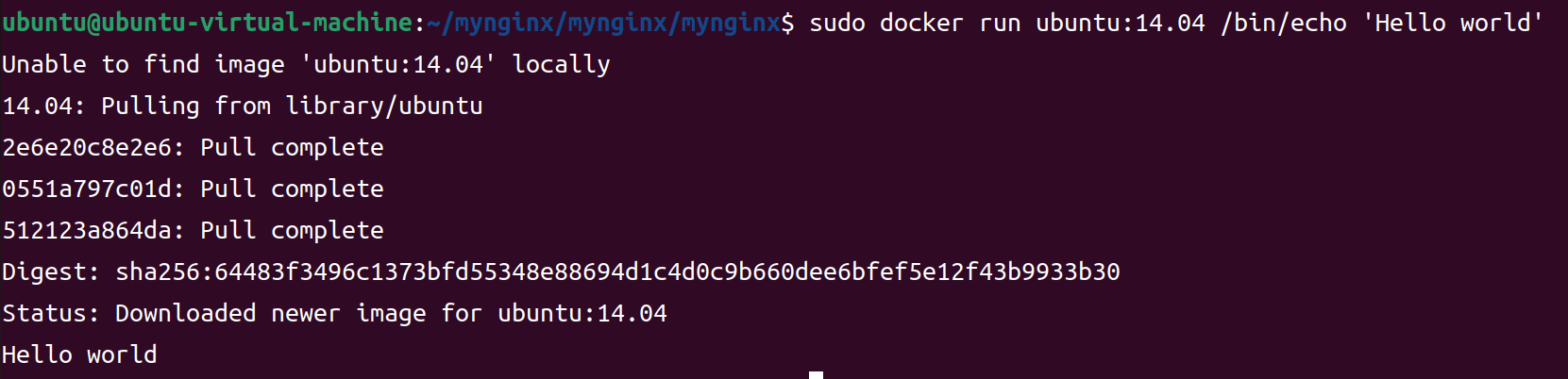
以ubuntu镜像为基础，启动bash并进行交互操作。在这里，我们执行了cat/etc/os-release命令



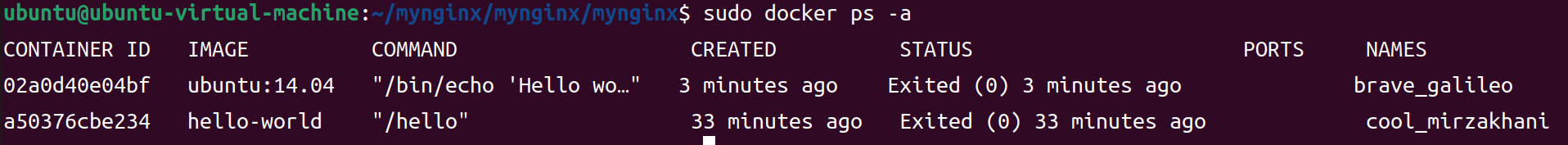
拉取最新版本的Nginx镜像

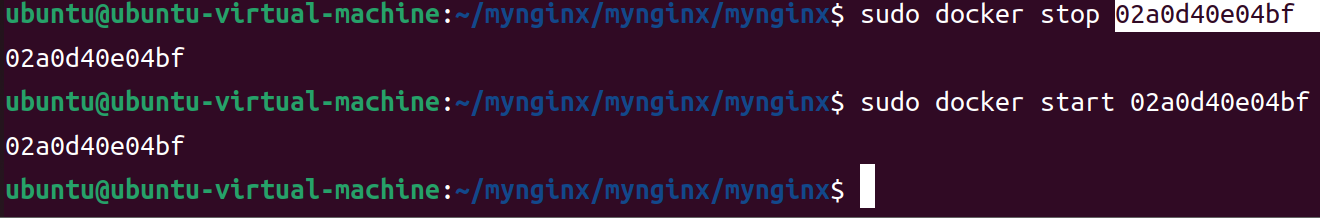


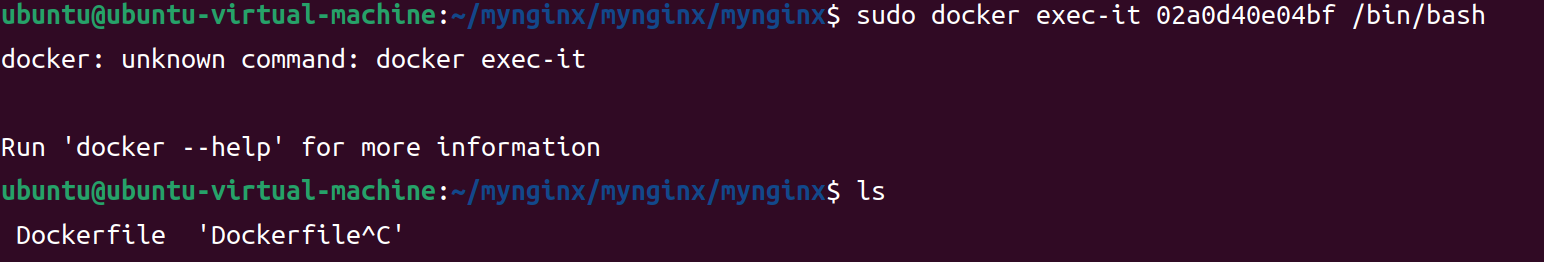
通过docker image命令查看本地已有的镜像列表

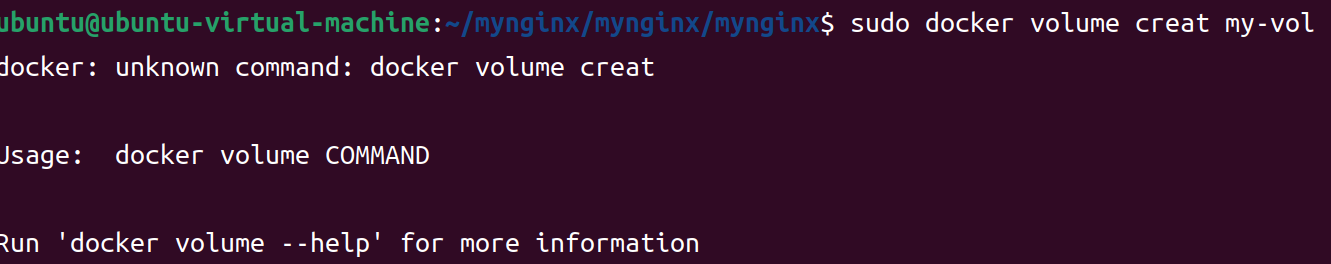
创建一个ubuntu容器

使用docker ps -a查看所有容器



使用docker stop和docker stop

使用docker exec-it进入容器的交互式终端，并在容器中执行查找命令

使用docker volume creat创建数据卷

3.3实验结果与分析

Docker实验围绕镜像，容器，仓库核心概念展开。从Docker Hub拉镜像，如 docker pull nginx，用docker images看本地镜像，了解镜像信息。编写Dockerfile定义基础环境，安装软件。容器操作方法的过程：docker run 启动容器，docker ps 看运行的容器，docker stop 、docker start 控制启停，进容器终端能执行命令。网络方面，默认的 bridge 模式能让容器连外网，还能自己建网络，让容器互相通信。数据管理用数据卷和绑定挂载，存数据不怕容器删了丢失。Docker Compose更方便，写一个docker-compose.yml ，一键启动多个服务，部署应用。Docker 把环境打包，让应用到处跑都一样，适合快速部署了。

第4章 Wireshark

4.1基本原理

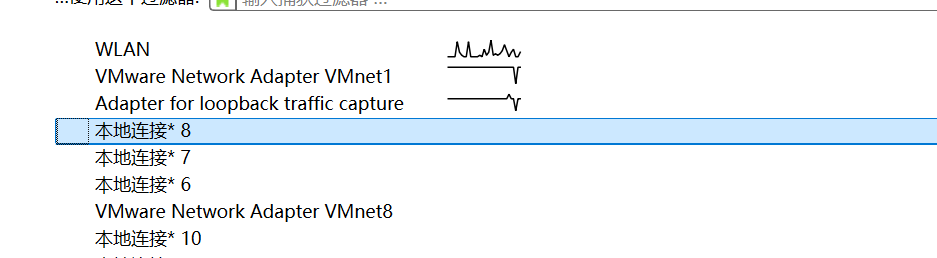
Wireshark 是一款网络数据包分析工具，核心靠数据包捕获、解析与展示三大功能实现网络监控。​它通过网卡进入 “混杂模式” 捕获流经的网络数据包，包括目标地址并非本机的流量。捕获后，借助内置的协议解析库，按 TCP/IP 等协议规范拆解数据包结构，提取源地址、目的地址、端口、数据内容等信息。​

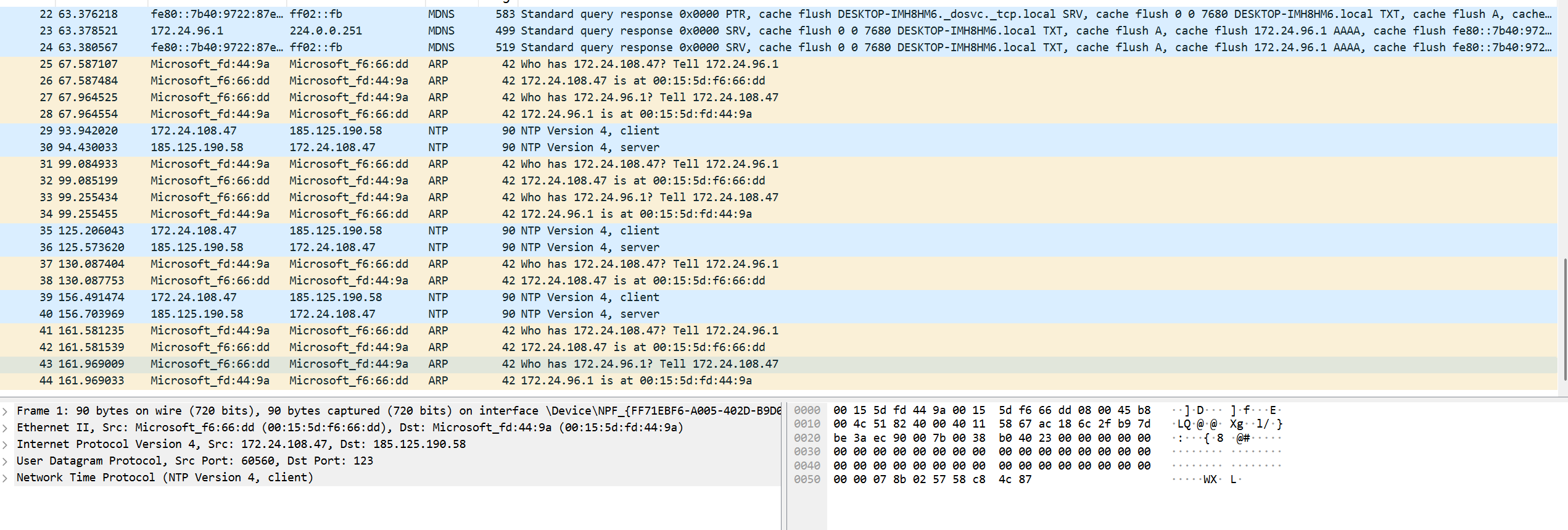
解析后的内容以列表、详情和十六进制等多视图展示，支持按协议、地址等筛选，方便定位问题。其底层依赖 libpcap 等抓包库与系统接口交互，实现跨平台的数据包捕获与分析，助力排查网络故障、分析协议交互。

4.2实验步骤

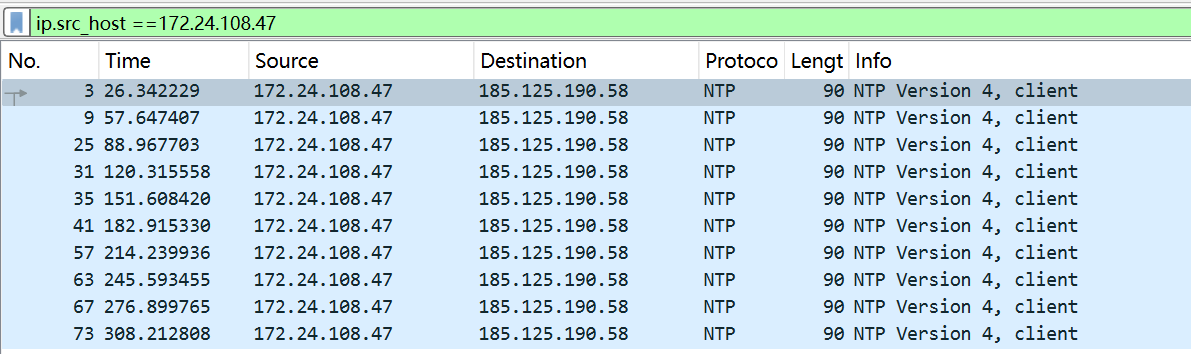
（1）基本操作

在启动的Wireshark操作界面上直接双击要捕获的数据包网络接口

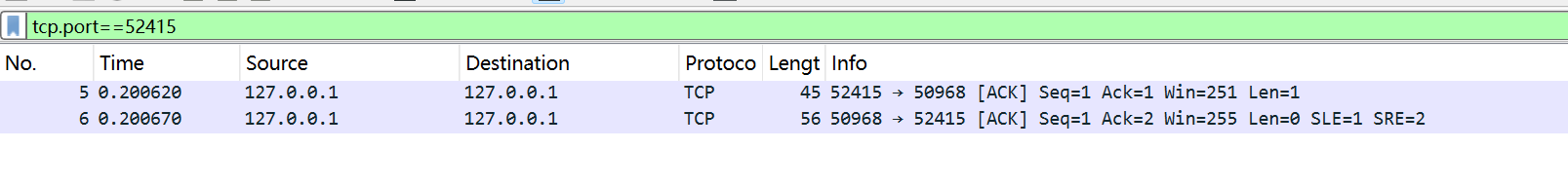
使用start按钮开始捕获，通过stop按钮停止捕获



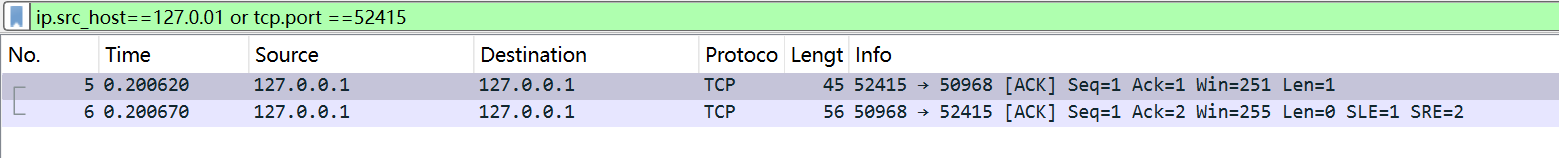
使用ip.src\_host过滤掉了源IP地址为172.24.108.47的数据包



使用tcp.port过滤tcp端口为5241的数据包

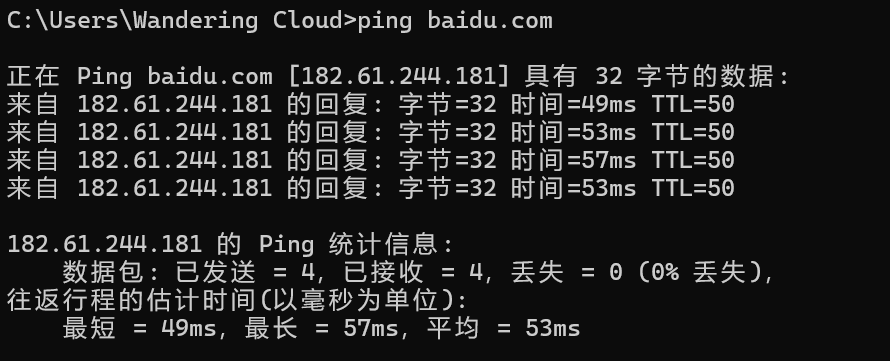


为了过滤出两种不同的数据包，可以使用Ip.src\_host==or tcp.port==过滤掉源IP地址为127.0.01的数据包或过滤tcp端口为52415的包

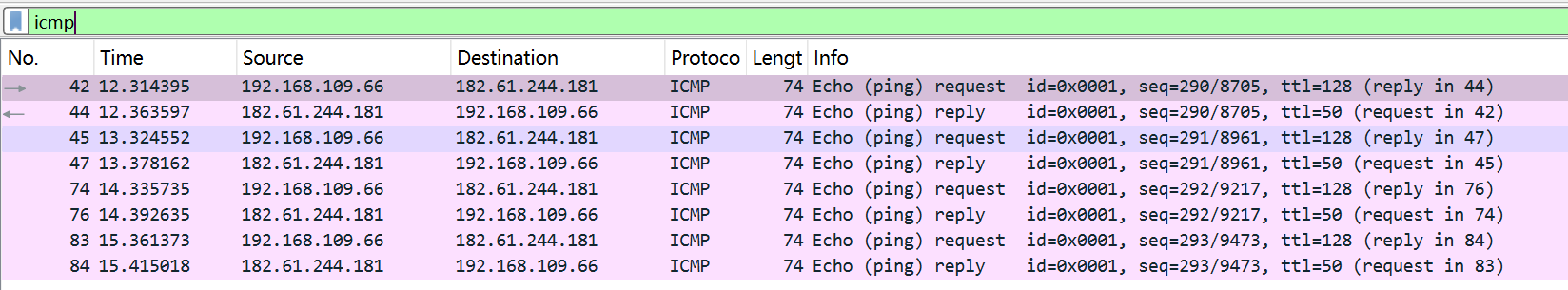


（2）数据包分析方面

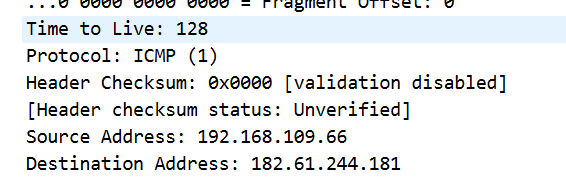
使用ping生成网络流量



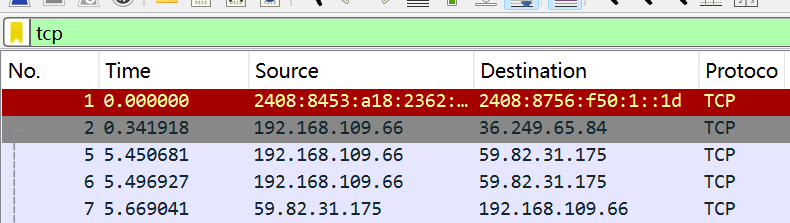
查看ping产生的ICMP包

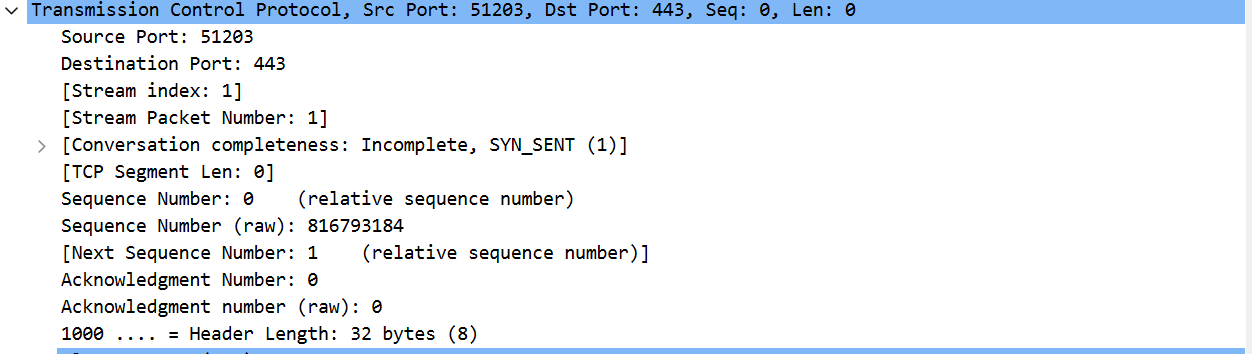


在IP包中查看地址，其中源IP地址是192.168.109.66，目标IP地址是182.61.244.181，TTL值是128和服务类型是1

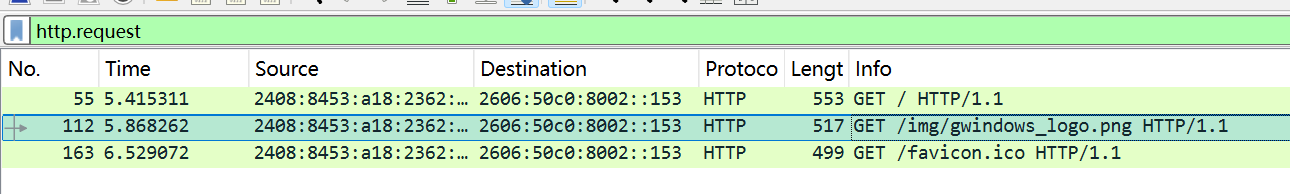


过滤出TCP包，可以查看端口号，序列号，标志位



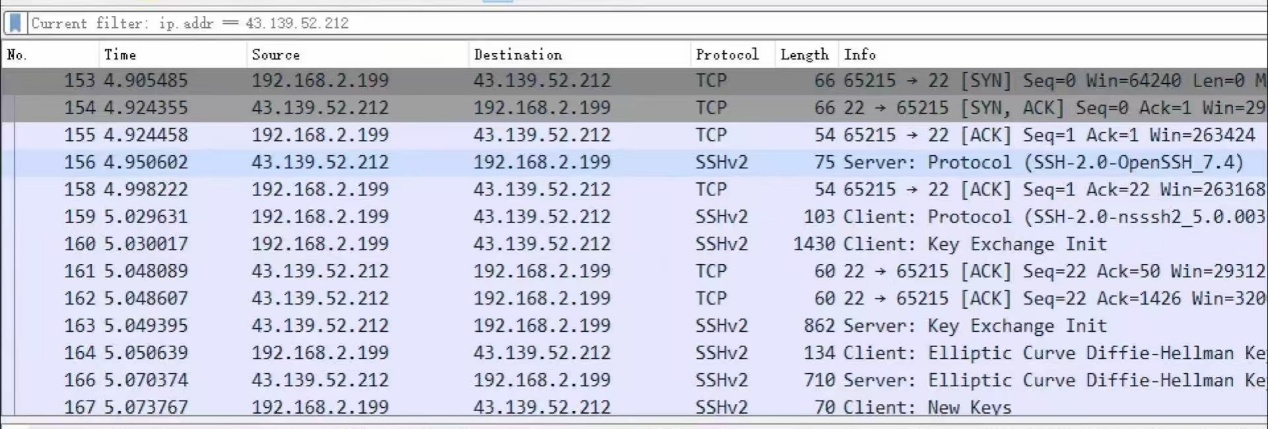


在HTTP请求查看请求方法，其中GET就是请求方法

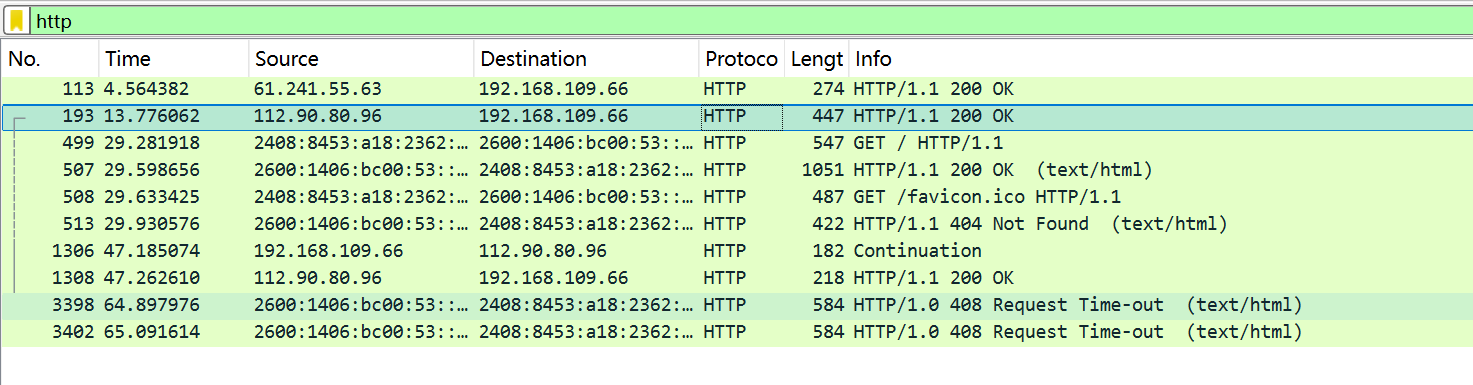


（3）协议分析

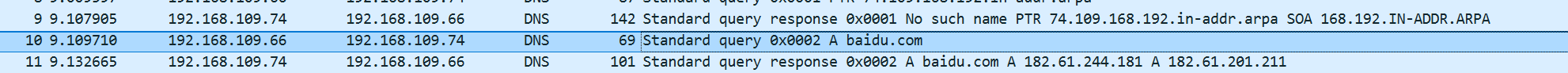
在TCP协议中，了解到了三次握手，我的电脑IP地址192.168.2.199与云服务器43.139.52.212了进行三次握手



HTTP请求：GET / HTTP/1.1是，HTTP响应：HTTP/1.1 200 OK



DSN请求包和响应包



4.3实验结果与分析

在Wireshark实操中，我会启动软件选网卡抓包，用过滤器筛特定IP、端口的包。看数据包列表，能懂编号、时间戳、源地址、协议这些字段，点进去看详情，从物理层到应用层，逐层查信息，像以太网帧的MAC地址、IP包的地址、TCP包的端口和标志位，HTTP请求里的网址、请求头都能看到。分析协议时，能认出TCP三次握手、四次挥手，看HTTP响应状态码，HTTPS虽加密但能看SSL握手，DNS 能查域名解析过程。抓包能帮我找网络问题，比如TCP重传影响速度，也能验证应用通信对不对。Wireshark让我看透网络数据，以后排查问题更有办法了。

第5章 总结

通过本次通用工具操作实训，我系统学习了Git、Docker、Wireshark和Burp Suite等工具的核心功能与实际应用，掌握了从基础操作到高级分析的完整流程。在Git部分，我深入理解了工作目录、暂存区与本地仓库的交互机制，熟练掌握了分支管理、远程仓库同步及冲突解决等关键操作。通过实践，我认识到Git在版本控制和团队协作中的高效性，尤其是分支的灵活运用能够显著提升开发效率。Docker实验让我熟悉了镜像与容器的生命周期管理，包括镜像构建、容器部署、网络配置和数据持久化。通过Docker Compose，我体验了多容器应用的便捷编排，理解了容器化技术对开发环境一致性和服务隔离的重要性。网络分析工具Wireshark的使用让我掌握了数据包捕获与协议分析的核心技能。我成功解析了TCP三次握手、HTTP请求响应等典型通信过程，并学会了通过过滤器精准定位问题。Burp Suite的实践则增强了我在Web安全测试中的实战能力，尤其是请求拦截与修改的灵活应用。本次实训不仅提升了我的工具操作能力，更让我认识到这些技术在现代化开发、运维和安全领域的核心价值。未来，我将进一步结合理论知识与实践需求，探索工具在复杂场景下的深度应用，为职业发展奠定坚实基础。