

**《****通用工具操作实训》**

**实训报告**

|  |  |
| --- | --- |
| **姓 名:** | **杨颖琦** |
| **学 号:** | **2409050073** |
| **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 |  | | 学院 |  | | | | 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 7](#_Toc22121)

[1 绪论 8](#_Toc1672)

[1.1 实训背景与意义 8](#_Toc8088)

[1.2 实训内容 8](#_Toc15501)

[2 Git 9](#_Toc16368)

[2.1 基本原理 9](#_Toc12019)

[2.2 实验步骤 9](#_Toc24865)

[2.3 实验结果与分析 10](#_Toc20188)

[3 Docker 17](#_Toc11850)

[3.1 基本原理 9](#_Toc12019)

[3.2 实验步骤 9](#_Toc24865)

[3.3 实验结果与分析 10](#_Toc20188)

[4 选修内容 24](#_Toc20398)

[4.1 基本原理 9](#_Toc12019)

[4.2 实验步骤 9](#_Toc24865)

[4.3 实验结果与分析 10](#_Toc20188)

[5 总结 42](#_Toc3076)

[参考文献 43](#_Toc18723)

**摘要**

学生作业管理系统是对学生作业资源进行管理的系统。随着信息技术的发展，教育领域正在经历一场深刻的变革。特别是在学生作业管理方面，传统的管理方式已经无法满足现代教育的需求。因此，开发一款高效、便捷的学生作业管理系统显得尤为重要。本文将探讨学生作业管理系统的功能、设计、实现及效果评估。

本次开发的学生作业管理系统就提供了一个操作的平台，可以将信息进行分类管理，并以学生作业信息所涉及的具体方面作为模块划分的依据。具体内容包括:发布作业、查看作业、提交作业、批改作业、用户管理、角色管理、菜单管理和注册、退出模块等，并在这些模块中完成了添加、修改、删除、查询等功能。

关键词：Java，Mysql，学生作业管理

# **第1章 绪论**

## 1.1 实训背景与意义

随着信息技术的不断发展，计算机在各个领域的应用愈发广泛和深入。在教学领域，培养学生全面且扎实的计算机操作技能是重要目标。通用工具操作实训作为培养学生实践技能的重要基础课程内容，为学生以后学习更为复杂的专业知识奠定坚实的基础，弥补了传统理论教学与解决实际问题之间的差距，使学生更好的适应未来职业岗位的要求。同时，在实际的计算机行业中，无论是互联网企业的快速迭代开发模式，还是传统企业数字化转型过程中的信息化建设，都依赖于多种计算机通用工具的协同使用。掌握这些工具的操作，是从业者能够顺利开展工作、提升工作效率的关键所在，也是企业在招聘和衡量人才时的重要考量因素。

通过计算机通用工具操作实训，我们能够亲手操作各类常见工具。在实际操作中，我们深入了解工具的功能特点、操作流程以及参数设置，从而熟练掌握这些工具的使用方法，这种实操技能的提升将为我们在后续更复杂的计算机任务处理中打下坚实基础。同时，计算机通用工具在使用过程中往往会遇到各种问题。在实训中，我们才有机会直面这些常见问题，并通过自己的思考和探索，尝试运用所学知识以及查阅资料等方式去解决。这种解决问题的过程能够极大地锻炼学生的应变能力和独立思考能力，使我们在未来面对复杂的计算机技术难题时能够冷静应对，迅速找到解决方案。此外，在许多计算机项目中，团队成员需要分工协作，共同运用各种工具完成任务。通过实训培养的团队合作意识和协作能力，能够使学生更好地融入这样的团队工作环境，实现团队目标。

## 1.2 实训内容

实训内容覆盖开发和网络攻防通用工具两个方向。主要内容包括：

（1）系统掌握 Git 的基本原理和常用操作，包括文件状态管理、工作区与版本库的交互、远程仓库协作以及分支管理等核心功能。实验过程中，学生需结合理论学习进行实操，在实践中理解 Git 的工作机制，为今后在软件开发项目中进行版本控制和团队协作打下坚实基础。

（2）系统掌握 Docker 的基本原理和常用操作，包括镜像构建、容器管理、网络配置、数据持久化以及多容器应用部署等核心功能。实验过程中，学生需结合理论学习进行大量实操，在实践中理解 Docker 的工作机制，为今后在软件开发和运维中使用 Docker 进行容器化部署打下坚实基础。

（3）系统掌握 Wireshark 的基本原理和常用操作，包括数据包捕获、协议分析、高级过滤和统计分析等核心功能，为今后在网络运维、网络安全分析等领域的工作打下坚实基础。

**第2章 Git**

2.1 基本原理

Git是分布式版本控制系统，每个开发者的电脑上都有整个仓库的副本，包括所有的历史记录和版本信息。在本地计算机上有一个.git隐藏文件夹，这个文件夹就是本地仓库，用于存储项目的版本历史、配置信息等。同时，通常会有远程仓库，如GitHub上的仓库，用于团队协作和代码共享。

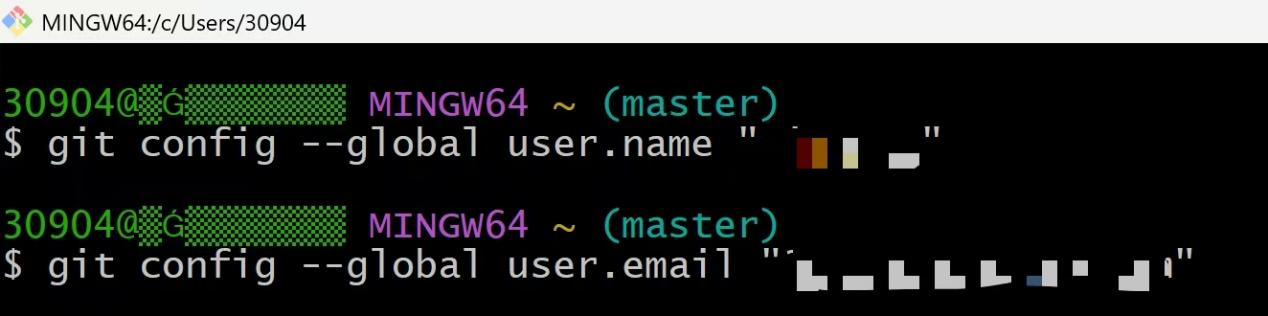
Git划分为三大区域，分别是工作区、暂存区和版本库。工作区，也称为工作目录，是开发者直接进行文件编辑和修改的地方，能够看见的目录。在这个区域中，文件处于未被版本控制或已被修改但尚未添加到暂存区的状态。暂存区，通常存放在.git/index文件下，是一个缓冲区域，用于临时存储准备提交到本地仓库的文件变更。当执行git add命令时，工作区中的文件修改会被添加到暂存区，这意味着这些修改已经被准备好纳入下一次的提交。版本库，分为本地版本库和远程版本库。本地版本库存储在本地计算机的.git文件夹中，保存了项目的所有版本历史记录，包括每次提交的文件快照、提交信息等。远程版本库则位于远程服务器上，是团队共享的代码仓库，开发者可以通过git push命令将本地仓库的代码推送到远程仓库，也可以通过git pull命令从远程仓库获取最新的代码并合并到本地仓库。

Git还有三个对象模型，分别是Blob对象、Tree对象和Commit对象。以及灵活的分支管理和清晰的工作流程。这些原理共同构成了Git强大而高效的版本控制体系，为开发者提供了便捷的代码管理、协作开发和版本控制功能，使得软件开发过程更加高效、可靠和可维护。

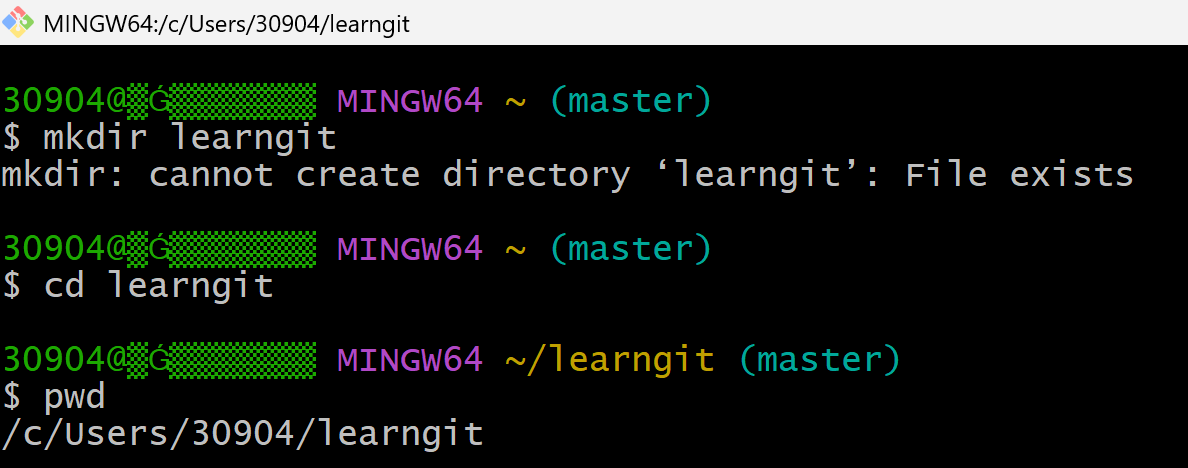
2.2 实验步骤

1.创建版本库

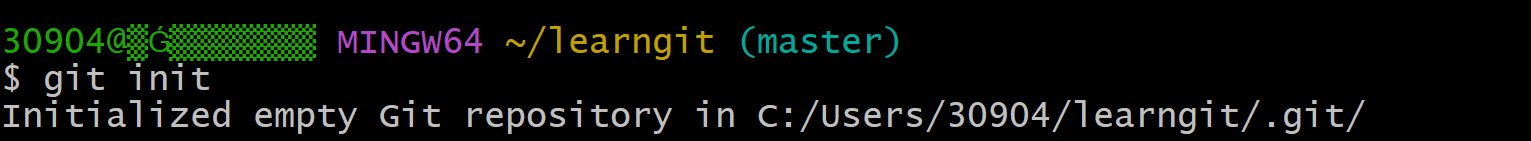
第一步，安装完成后的基础操作。



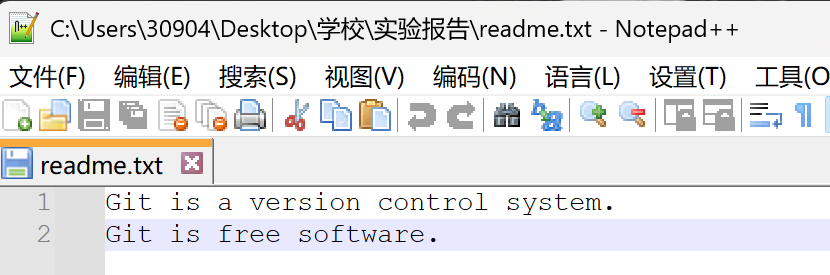
第二步，选择一个合适的地方创建一个空目录。



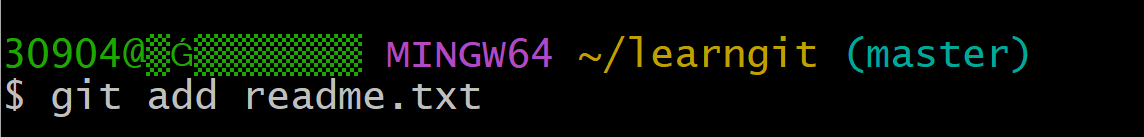
第三步，通过Git init命令把这个目录变成Git可以管理的仓库。



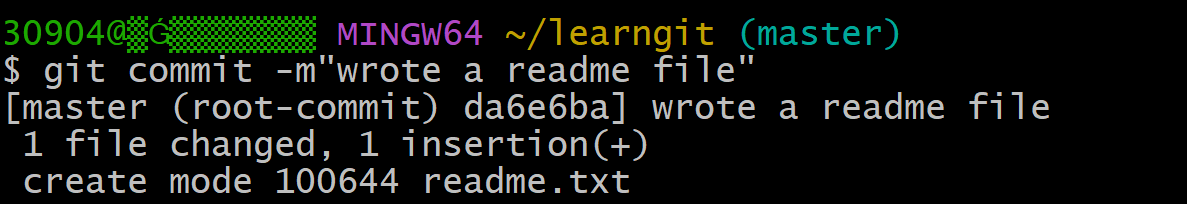
第四步，编写一个readme.txt文件，内容如下：



用命令git add告诉Git,把文件添加到仓库：

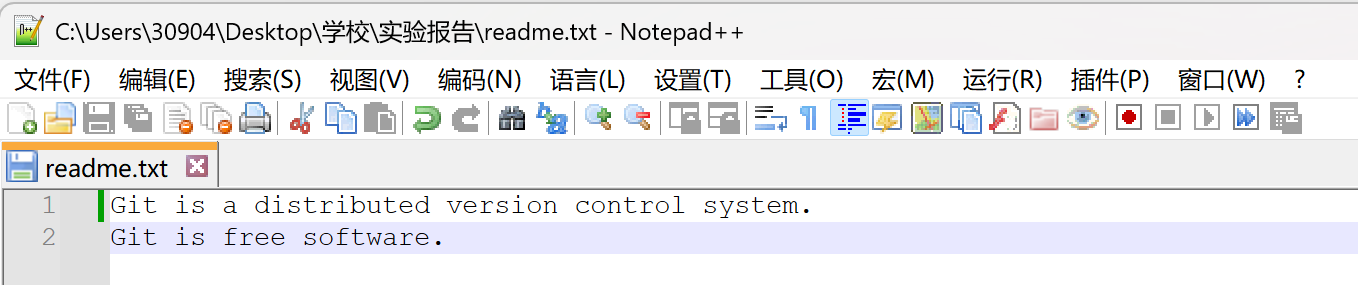


用命令git commit告诉Git，把文件提交到仓库：

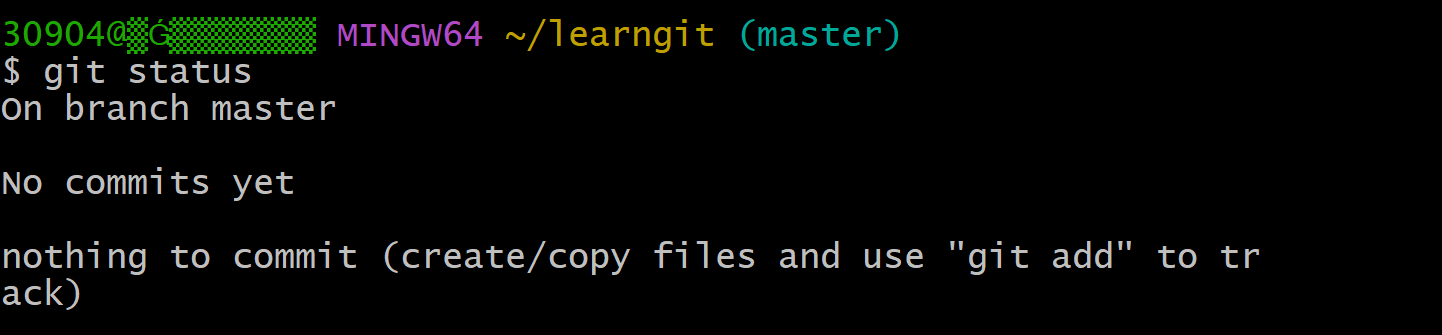


2.时光机穿梭

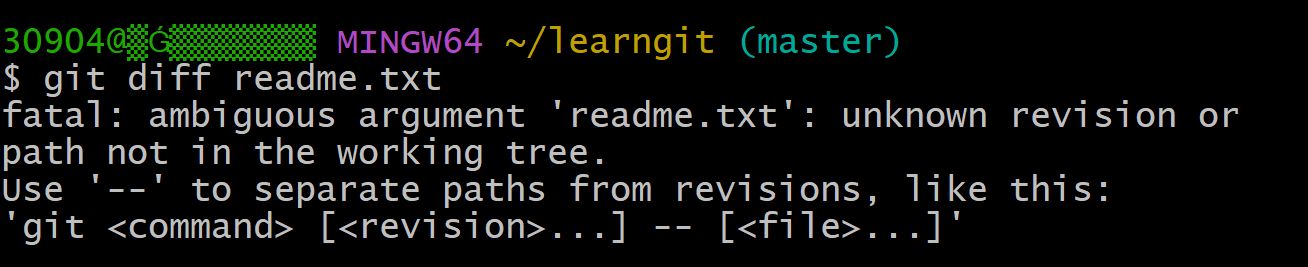
修改readme.txt文件：



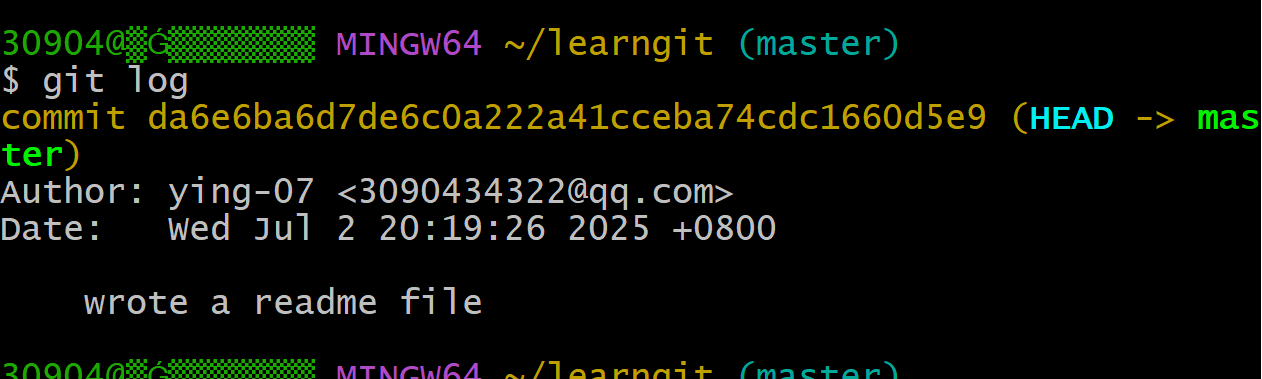
运行git status命令看结果：



用git diff命令查看历史修改：

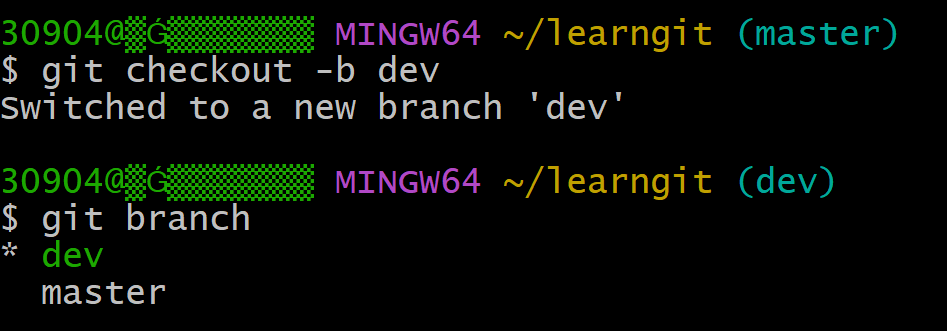


使用git log 命令，查看从最近到最早的提交日志：

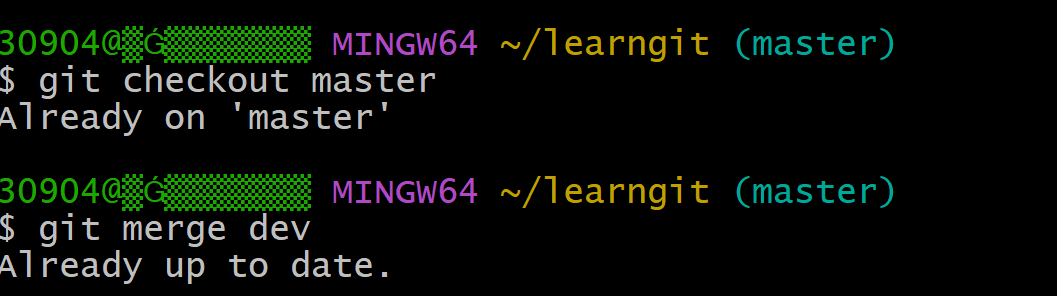


3.分支管理

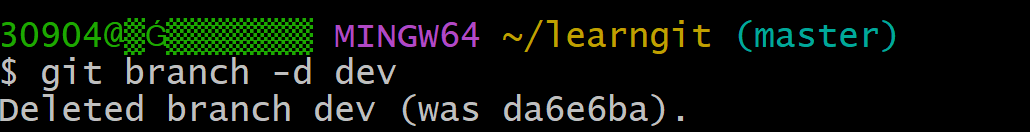
首先，创建dev分支，然后切换到dev分支，并用git branch命令查看当前分支：



然后，切换到主分支上，然后把dev 分支的内容合并到master 分支上：

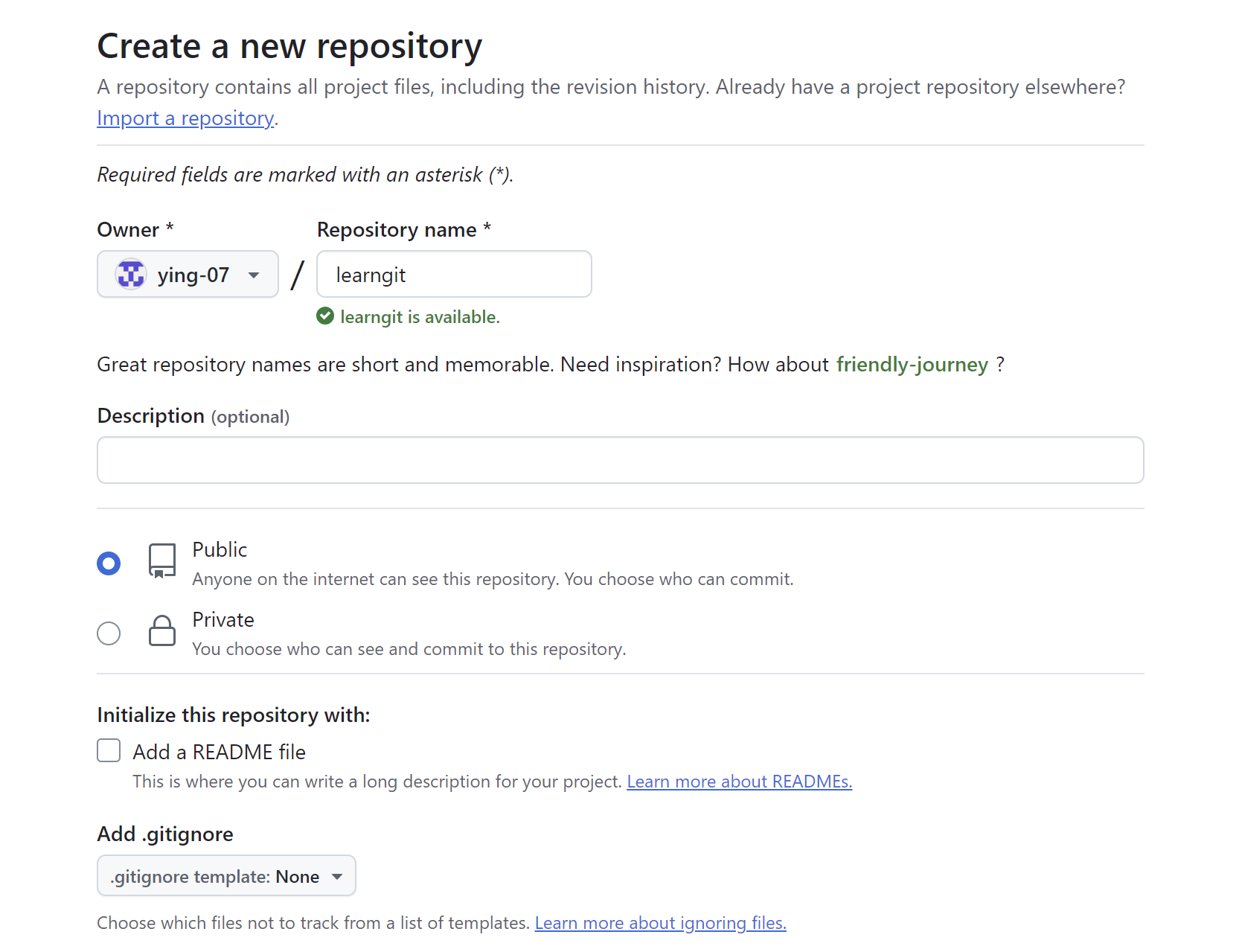


合并完以后删除branch，此时就只剩下master 分支

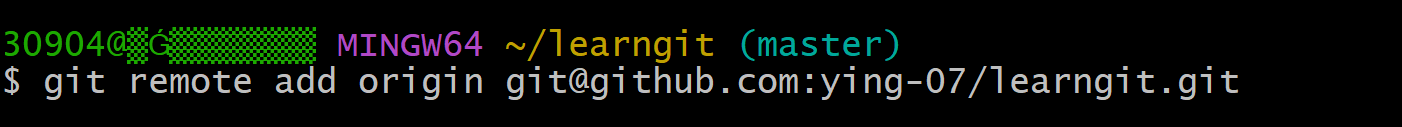


4.远程协作

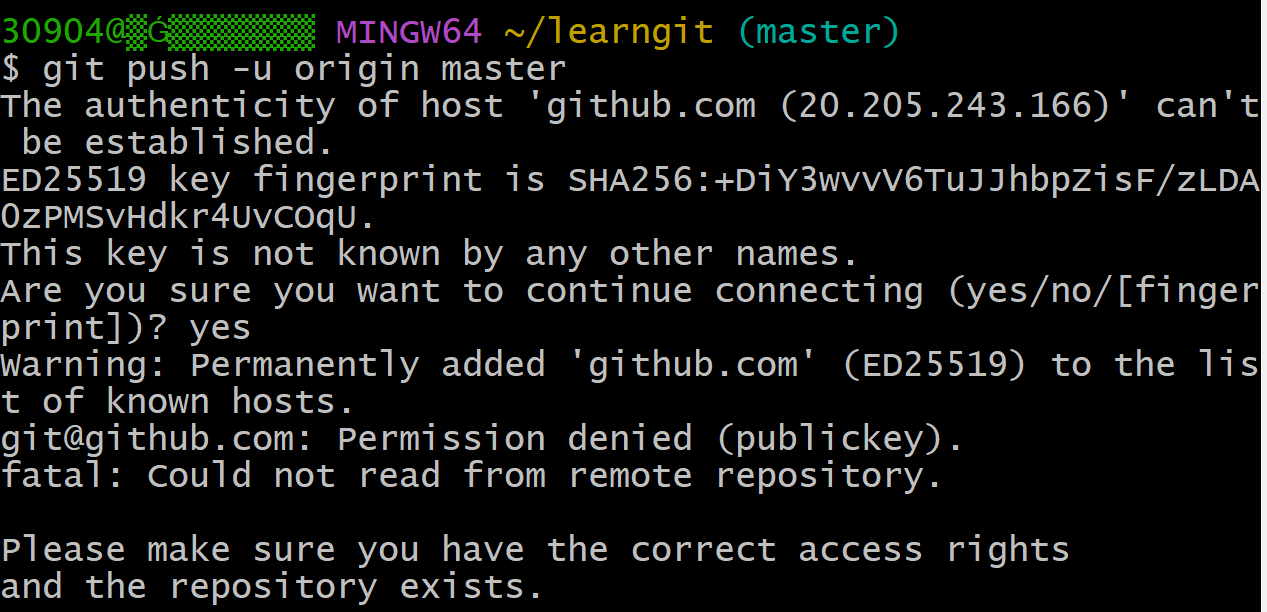
首先创建一个新的git仓库：



然后把本地的git仓库关联到远程仓库：



把本地库的内容推送到远程库上：



2.3实验结果与分析

理解了工作目录、暂存区和本地仓库这三个核心区域的功能及交互关系。通过git add命令可将工作目录中的修改添加到暂存区，git commit -m "commit message"命令则能将暂存区的内容提交到本地仓库。通过实操，观察文件在这三个区域之间的状态转换，例如新建文件后查看其未跟踪状态，修改文件后确认已修改状态，使用git add将其加入暂存区，再通过git commit提交到本地仓库，并用git log查看提交历史。理解当执行git add时，暂存区的目录树如何更新，文件内容如何写入对象库；执行git commit时，暂存区的目录树如何写入版本库，master 分支如何更新。

掌握了与远程仓库的交互。git push命令用于将本地版本库的提交推送到远程仓库，如git push origin main；git pull或git fetch命令用于从远程仓库获取更新，git pull是git fetch和git merge的组合操作。配置远程仓库，通过git remote add <remote\_name> <remote\_url>添加远程仓库，使用git remote -v查看远程仓库信息，并完成从本地推送到远程、从远程拉取更新的完整流程，理解origin作为远程仓库默认别名的作用，以及 HTTPS 和 SSH 两种远程 URL 的区别。

掌握了分支管理。使用git checkout -b <branchname>创建并切换到新分支，通过git branch查看本地分支，git branch -r查看远程分支，git branch -a查看所有分支。在不同分支中进行开发操作，体验分支切换时工作目录内容的变化，理解分支如何实现并行开发而不影响主代码库。合并分支是分支管理的关键操作，学生将学习如何将其他分支合并到当前分支，如git merge <branchname>，并掌握解决合并冲突的方法：当合并出现冲突时，手动修改冲突文件，使用git add <conflict-file>标记冲突解决，再执行git commit。此外，还需掌握删除分支的操作，包括删除本地分支git branch -d <branchname>、强制删除未合并分支git branch -D <branchname>以及删除远程分支git push origin --delete <branchname>。

**第3章Docker**

3.1基本原理

Docker 是一种开源的应用容器引擎，它基于容器化技术，允许开发者将应用程序及其依赖打包到一个独立的容器中，从而实现快速部署、隔离运行和跨环境一致性。其核心概念包括以下几点：

（1）镜像（Image）

Docker 镜像是容器运行的基础，它是一个只读模板，包含了运行应用程序所需的代码、运行时、库、环境变量和配置文件。镜像是分层的，每一层都代表了镜像构建过程中的一个操作（如安装软件包、复制文件等）。这种分层结构使得镜像可以高效地存储和共享。

（2）容器（Container）

容器是镜像的运行实例。通过从镜像创建容器，Docker 可以在隔离的环境中运行应用程序。容器具有独立的文件系统、网络接口和进程空间，但它们共享宿主机的操作系统内核，因此启动速度非常快，资源占用也比虚拟机低得多。

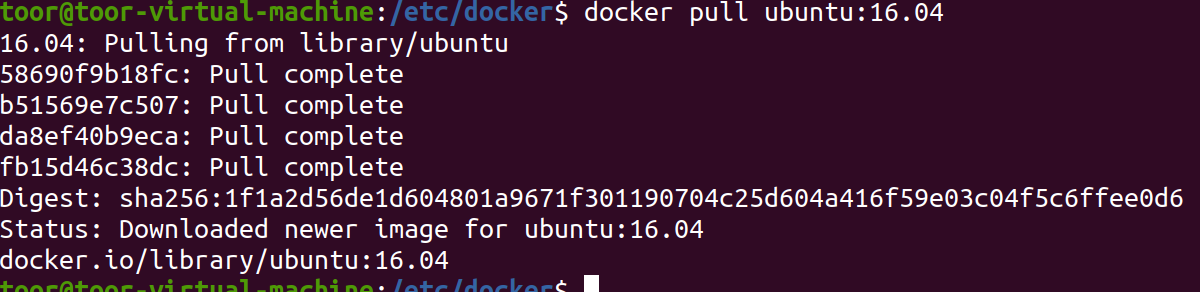
（3）仓库（Repository)

仓库是用于存储和分发Docker镜像的服务，包括公共仓库和私有仓库。仓库可以储存多个版本的镜像，并支持镜像的推送和拉取。通过标签，开发者可以标识不同的镜像版本，方便管理和回滚。而私有仓库支持访问控制，确保只有授权用户可以访问和操控镜像。

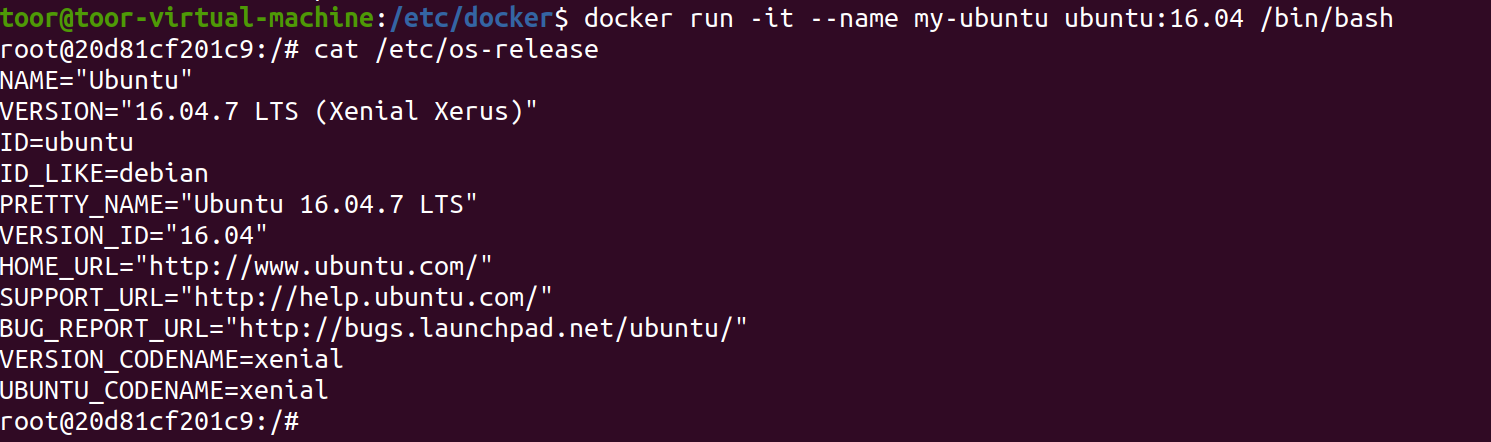
综上，Docker 的基本原理是通过容器化技术，利用 Linux 内核特性实现隔离和资源管理，通过镜像和容器的机制，实现应用程序的快速打包、部署和运行。它极大地简化了开发和运维流程，提高了开发效率和系统的可移植性。

3.2实验步骤

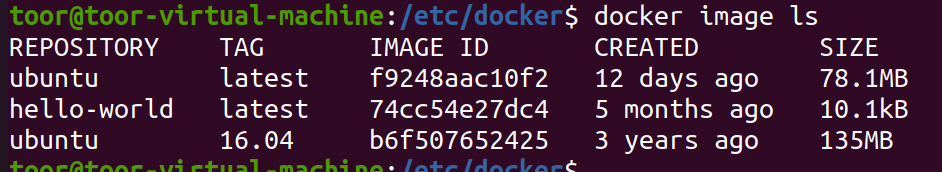
1.拉取一个ubuntu的镜像：

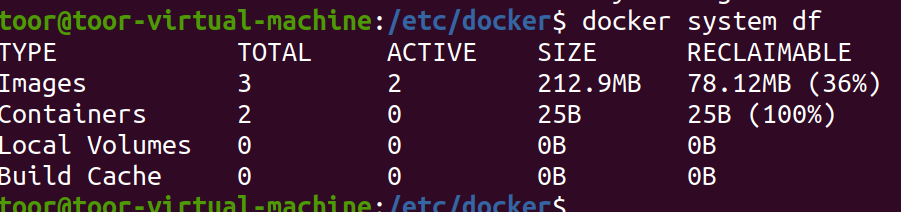


2.以镜像为基础启动一个容器来运行（启动里面的bash并进行交互式操作）：

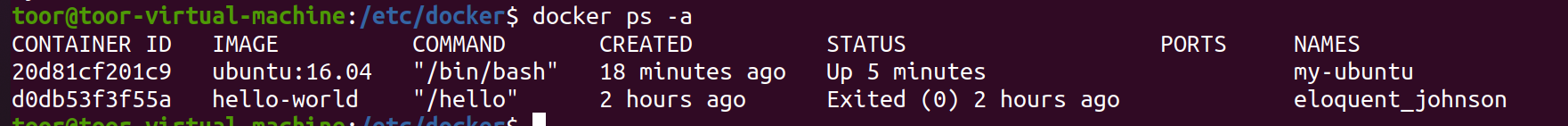


3.列出已经下载好的镜像（使用docker image ls命令）：

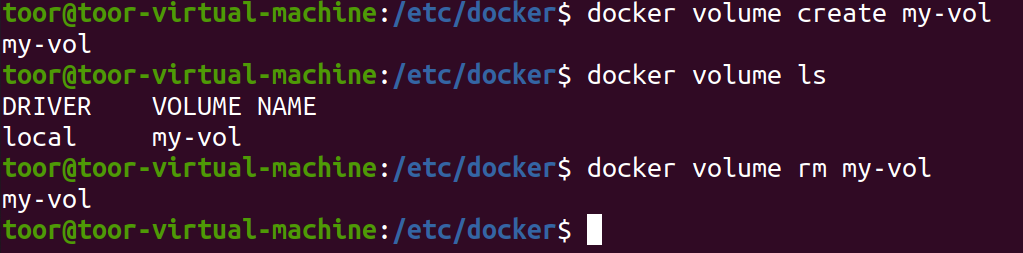
4.查看镜像、容器、数据卷所占用的空间：



5.查看所有容器：



6.创建一个数据卷，后查看所有数据卷，最后删除数据卷：



3.3实验结果与分析

系统掌握 Docker 的基本原理和常用操作,理解镜像、容器和仓库的功能及交互关系，包括镜像构建、容器管理、数据管理基础等核心功能。通过实操，掌握如何从仓库拉取镜像、如何基于镜像创建并运行容器，以及如何将自己创建的镜像推送到仓库。掌握基本的镜像操作。学习如何使用docker pull命令从 DockerHub拉取镜像，通过 docker images命令查看本地已有的镜像列表，了解镜像的仓库名、标签、ID等信息。掌握基本的容器操作。学习如何使用docker run命令创建并启动容器、学习停止和启动容器以及重新启动已停止容器。掌握数据管理基础。学习 Docker 的数据卷和绑定挂载机制。数据卷是 Docker 管理的目录，用于持久化存储容器数据，学习使用docker volume create my-volume创建数据卷，实现数据的持久化和共享。

**第4章Wireshark**

4.1基本原理

Wireshark 是一款常用的网络协议分析工具，其基本原理涉及**数据捕获**、**数据解析**、**数据显示**这几个关键环节：

### 数据捕获**：**计算机的网卡在默认情况下只接收目的地址为本机的数据包。而 Wireshark 在进行数据捕获时，会将网卡设置为混杂模式，使其能接收流经该网卡的所有数据包，不管这些数据包的目的地址是不是本机。这样就能获取到网络中传输的所有数据流量。在不同的操作系统中，Wireshark 依靠不同的抓包驱动来实现数据捕获。

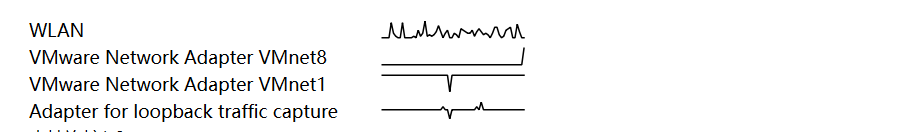
### （2）数据解析：Wireshark 内置了大量的协议解析器，涵盖了 TCP/IP 协议族（如 IP、TCP、UDP 等）、应用层协议（如 HTTP、FTP、SMTP 等 ），以及其他众多网络协议。当捕获到原始数据包后，Wireshark 会根据数据包的头部特征，按照从链路层到应用层的顺序，依次将数据包与各个协议解析器进行匹配。依据网络分层模型，Wireshark 对数据包进行层次化解析。先解析链路层，接着解析网络层，再解析传输层，最后解析应用层协议，提取出具体的应用数据。

### （3）数据显示：Wireshark 提供了一个直观的图形用户界面，将解析后的数据以表格形式呈现。每一行代表一个数据包，每一列对应数据包的不同属性，如编号、时间戳、源地址、目的地址、协议类型、长度等。同时，为了便于用户快速识别不同类型的数据包，Wireshark 支持颜色编码功能。

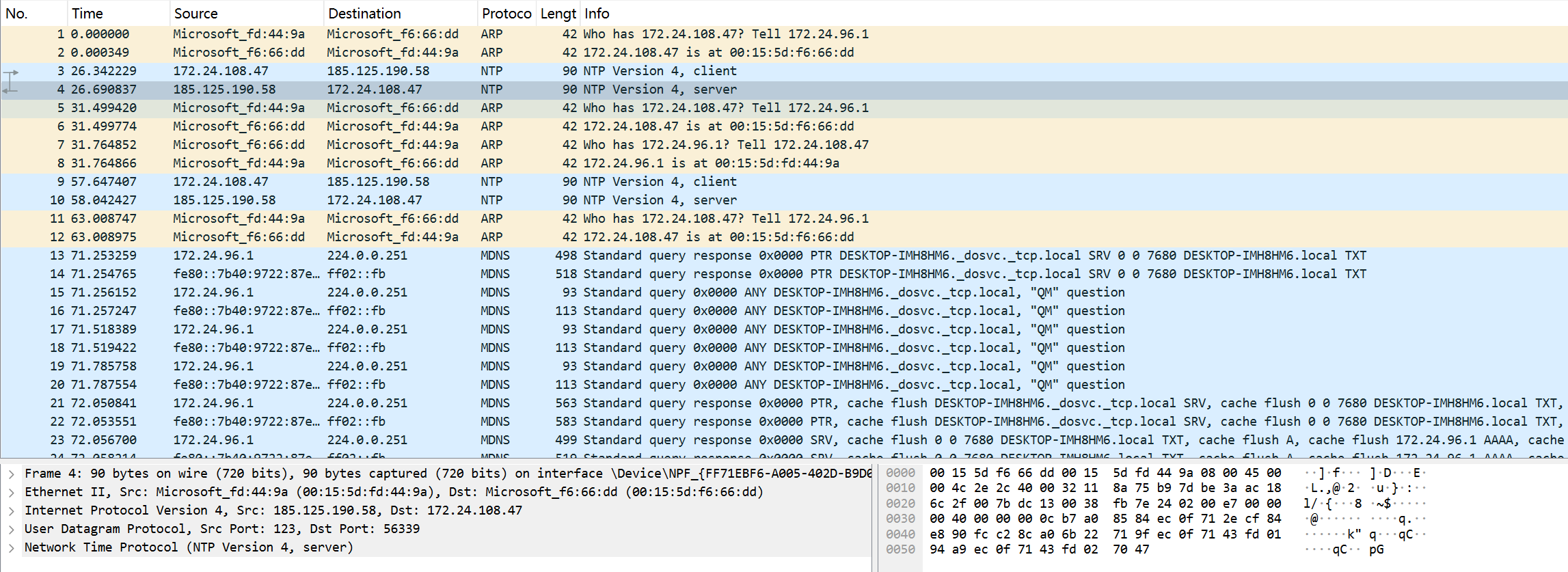
4.2实验步骤

(1)基本操作

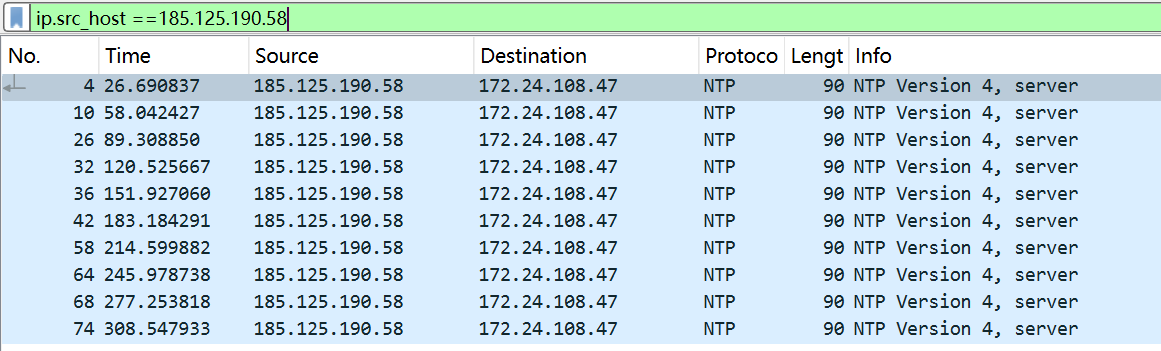
在Wireshark的首页面点击合适的网络接口进行捕获数据包的实验



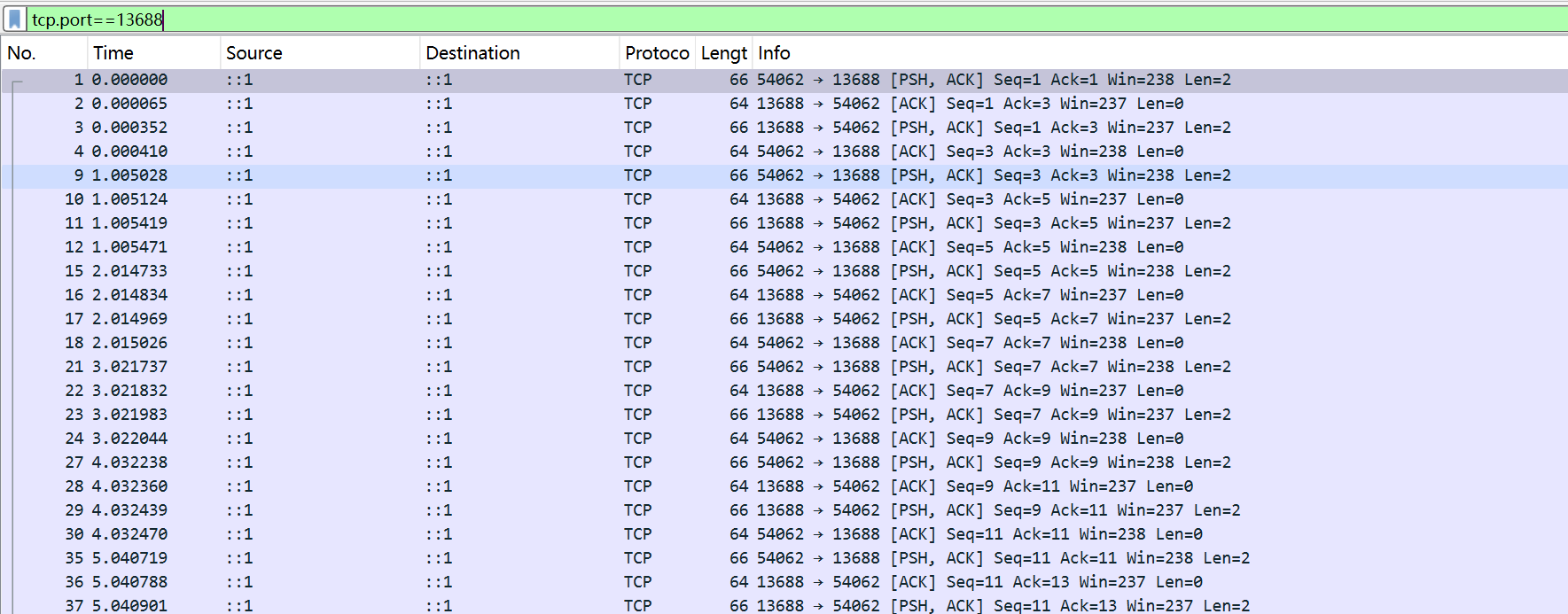
点击蓝色按钮进行捕获，红色停止捕获



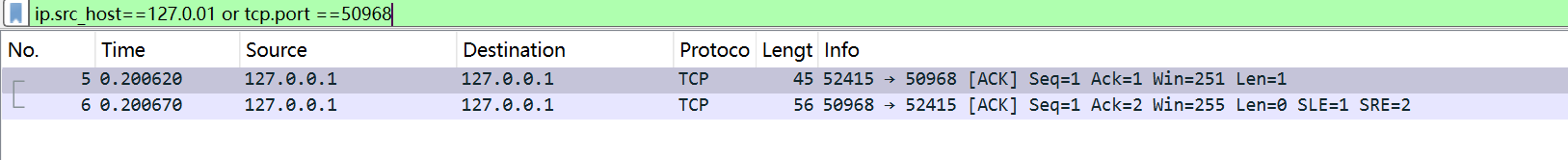
要想获得特定的源IP地址，例如地址为172.24.108.47的数据包，可以通过使用ip.src\_host，



tcp.port过滤tcp端口为13688的数据包

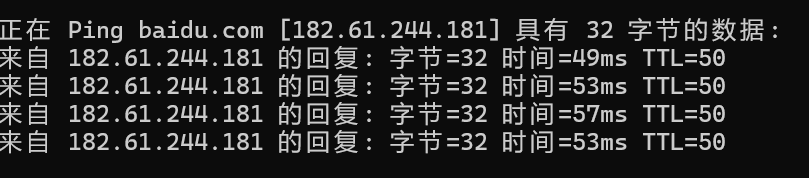


Ip.src\_host==or tcp.port==可以同时实现上述两种方法的过滤，而and是同时要符合两种特性的才会被保留

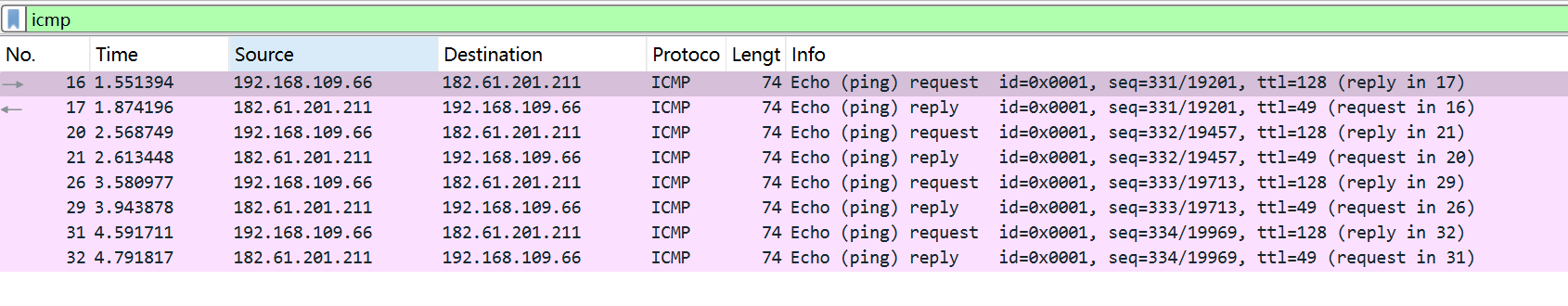


（2）在数据包分析方面

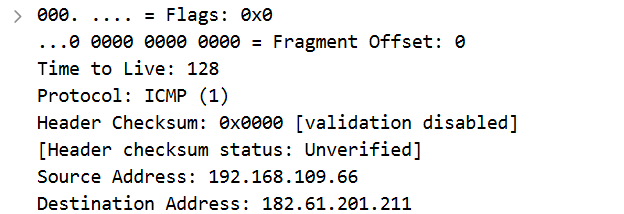
网络流量的生成



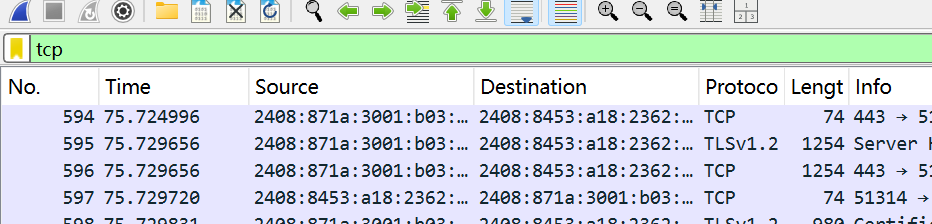
在过滤器中输入icmp查找产生的ICMP包

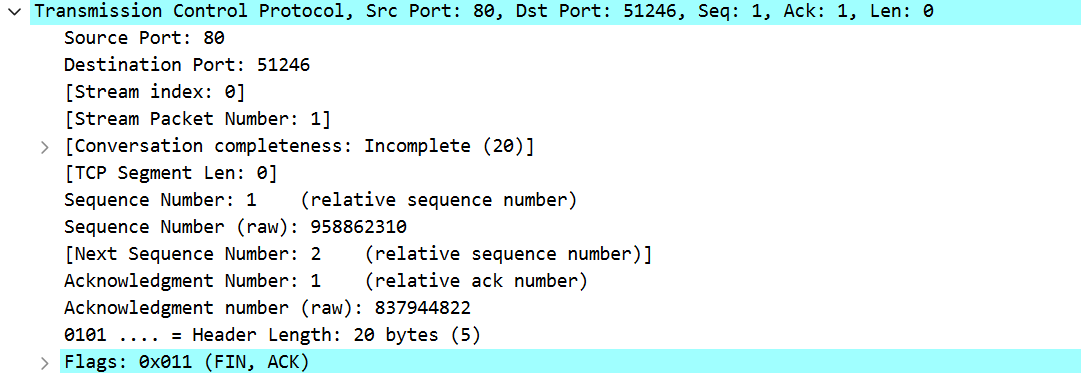


在下面的白色窗口中，我们可以看到我们过滤的数据包后信息，其中有IP地址，目标IP地址，TTL值和服务类型。

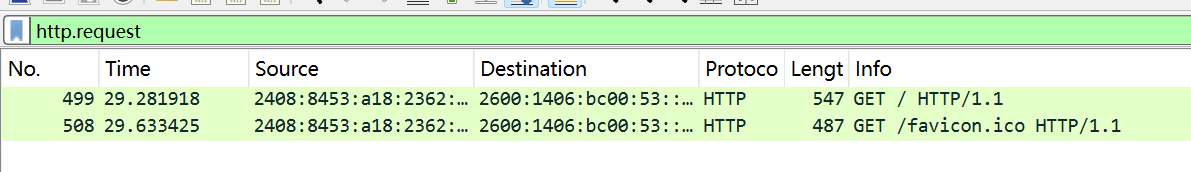


在TCP包中查看端口号（SYN），序列号(ACK )，标志位(FIN)



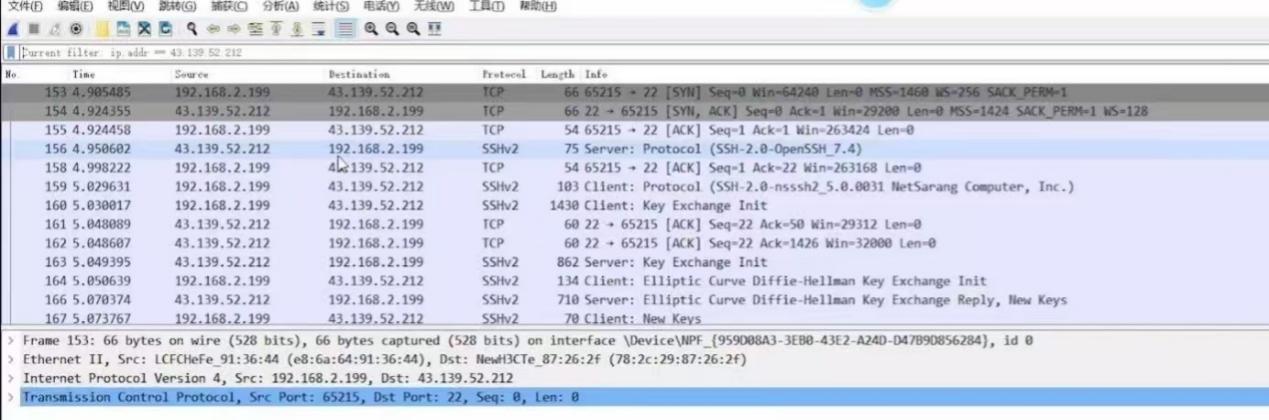


在HTTP中可以查看到GET的请求方法

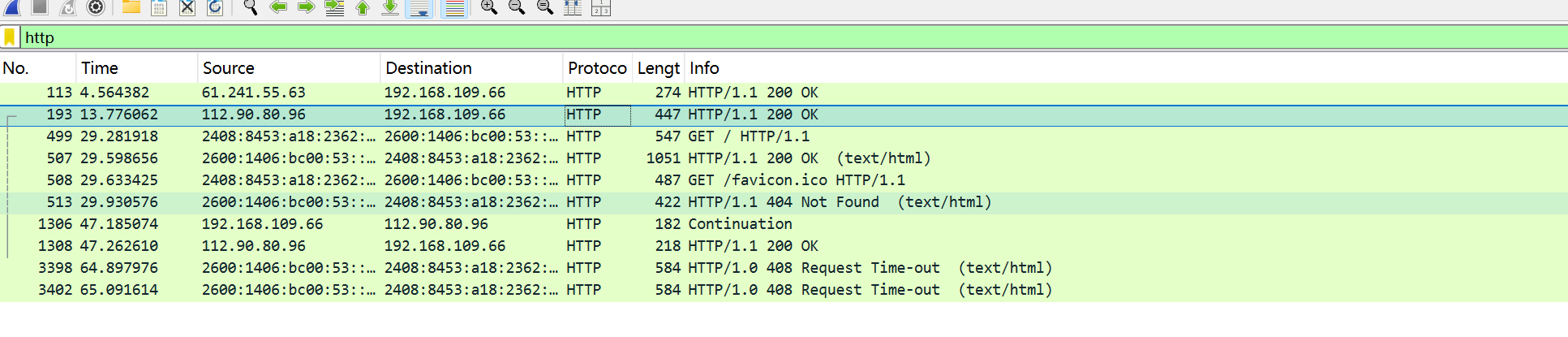


（3）协议分析

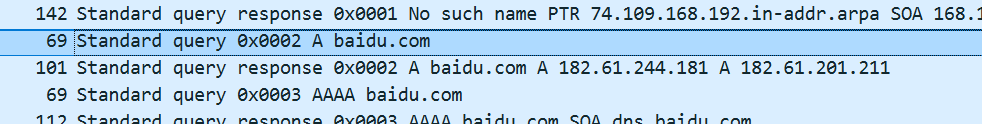
在TCP协议中，可以明白到是如何进行三次握手的，在下面的图片，前三列展示了三次握手的过程



以下是HTTP请求和响应



在DSN协议中，展示了查询请求和响应



4.3实验结果与分析

掌握软件的启动与界面组成，包括主窗口的数据包列表、详情面板和十六进制数据视图。选择合适的网络接口进行数据包捕获，使用start按钮开始捕获，通过stop按钮停止捕获。理解捕获过滤器的使用，在数据包分析方面，学习如何解读捕获到的数据包。理解数据包列表中各列的含义，如时间戳、源地址、目的地址、协议、长度和信息摘要。深入分析数据包详情，逐层查看各协议层的信息，从物理层的帧头到应用层的数据。

**第5章 总结**

本次计算机通用工具实训，围绕 Wireshark 网络分析、Docker 容器管理、git等工具展开实践。本次实训，是理论向实践转化的重要桥梁，让我对计算机通用工具的应用有了深刻认知，我不仅掌握了 Git、Docker 和 Wireshark 这三种工具的使用方法，还学会了如何将它们综合应用于实际项目中。在软件开发过程中，我可以使用 Git 来管理代码版本，使用 Docker 来构建和部署应用程序，同时使用 Wireshark 来分析网络通信情况，确保应用程序的正常运行和安全性。这种综合应用的能力让我在解决实际问题时更加得心应手， 同时，也让我意识到工具间协同（如网络工具与容器工具结合排查容器网络问题 ）的潜力，拓宽了技术应用视野。展望未来，我将继续深入学习这些工具的高级功能和最佳实践。例如，在 Git 方面，我将学习更复杂的分支策略和代码审查流程；在 Docker 方面，我将探索容器编排技术如 Kubernetes 的使用；在 Wireshark 方面，我将学习更多高级的网络协议分析技巧和安全分析方法。我相信，通过不断学习和实践，我能够更好地将这些工具应用于实际工作中，为软件开发和网络安全领域做出更大的贡献，在计算机技术实践之路上稳步前行。