Introduction to Cloud Computing Report

Lab 4: Gitea Deployment and Git LFS Verification

Name: Yin Yuang
Student ID: 202383930027
Class: Software Engineering Class 1

Introduction to Cloud Computing
(Autumn,2025)

Nanjing University of Information Science and Technology, China
Waterford Institute

I. Object

The objective of this lab is to build a self-hosted Git service (Gitea) through containerization and traditional deployment methods, verify Gitea's support for Git Large File Storage (LFS), and master core operations such as large file management, repository creation, and data backup/recovery. By completing this lab, students will gain proficiency in:

- Installing and configuring Docker Desktop on the Windows platform (with WSL 2 backend).
- Deploying Gitea using Docker containers and verifying its Git LFS functionality.
- Creating Gitea repositories to manage large files (1GB+) and lab-related work artifacts.
- Deploying Gitea without Docker (self-selected database) to understand traditional application deployment processes.
- (Optional) Backing up Gitea data to external storage and restoring it on another computer.

II. Content

1. Environment Preparation

The experiment began by preparing the Windows environment required for both container-based and native Gitea deployments. WSL 2 was enabled using the `wsl --install` command, and Docker Desktop was installed and configured to run with the WSL 2 backend. This setup provided the necessary foundation for running Linux-based containers and performing subsequent DevOps operations.

2. Deployment of Gitea Using Docker

Gitea was first deployed in a containerized environment using the official Docker image. Persistent directories were created to store application data, and the service was launched through Docker. Initial configuration, including administrator account creation and basic server settings, was completed through the web interface.

3. Verification of Git LFS Functionality

To verify support for large binary assets, Git LFS was installed and configured on the client machine. Tracking rules were added for designated file types, and a large (1 GB+) test file was prepared to validate server-side LFS handling. This process ensured that both the client and Gitea server were capable of correctly managing large objects.

4. Repository Setup for Laboratory Submission

A dedicated repository for the laboratory module was created on the Gitea server. All experiment-related files were organized locally, committed using Git, and pushed to the repository. This step ensured that the entire lab workflow—from preparation to documentation—was fully tracked and version-controlled.

5. Native Installation of Gitea Without Docker

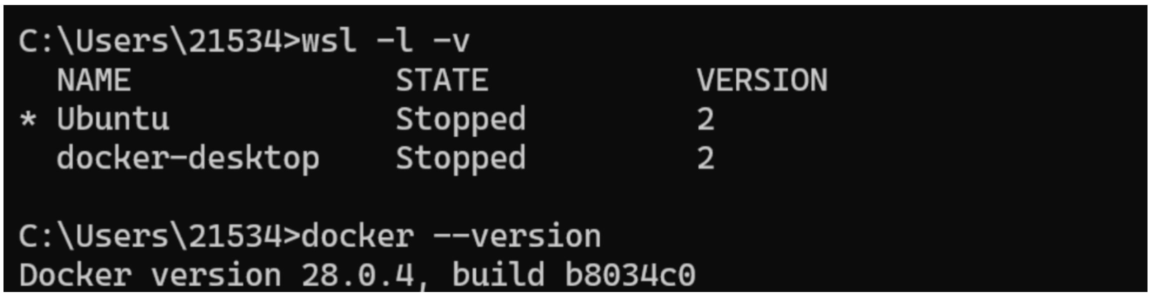
To satisfy the requirement for a non-containerized deployment, Gitea was also installed natively on Windows. The Windows binary was placed in a dedicated directory, along with supporting `data` and `log` folders. SQLite was selected as the database backend to simplify configuration. The service was started using `gitea.exe web`, and the initial configuration was completed through the browser. A test repository was created, and standard Git operations were performed to verify full integration with the native Gitea instance.

III. Results

1.Environment Setup

1.1 Docker Desktop Installation

To enable containerized deployment, the Windows environment was prepared with WSL 2 and Docker Desktop.



```
C:\Users\21534>wsl -l -v
  NAME                  STATE           VERSION
* Ubuntu                Stopped         2
  docker-desktop        Stopped         2

C:\Users\21534>docker --version
Docker version 28.0.4, build b8034c0
```

Figure 1: Docker Desktop installed and running successfully in Windows.

This screenshot verifies that Docker Desktop has been properly installed, initialized, and authenticated. The running status of Docker confirms that the environment is ready for container-based Gitea deployment.

1.2 Deploying Gitea via Docker

Deployment references:

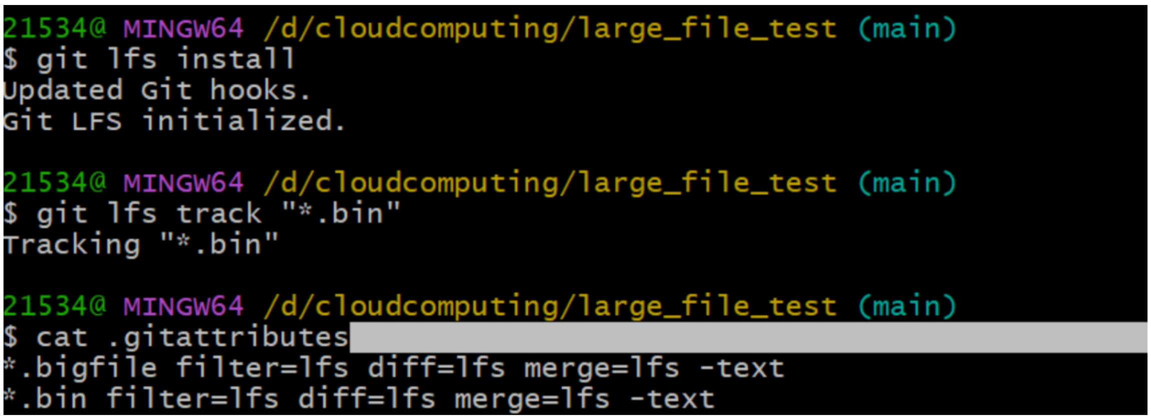
- `https://about.gitea.com/`
- `https://docs.gitea.com/installation/install-with-docker`

The Gitea instance was deployed using the official Docker image, ensuring isolation and simplified configuration.

2.Git LFS Functionality Verification

Git LFS is essential for managing large binary files within Git repositories. The experiment verifies both the client-side LFS configuration and server-side LFS support in Gitea.

2.1 LFS Client Configuration



```
21534@ MINGW64 /d/cloudcomputing/large_file_test (main)
$ git lfs install
Updated Git hooks.
Git LFS initialized.

21534@ MINGW64 /d/cloudcomputing/large_file_test (main)
$ git lfs track "*.bin"
Tracking "*.bin"

21534@ MINGW64 /d/cloudcomputing/large_file_test (main)
$ cat .gitattributes
*.bigfile filter=lfs diff=lfs merge=lfs -text
*.bin filter=lfs diff=lfs merge=lfs -text
```

Figure 2: Git LFS initialized and LFS tracking rules applied in the local repository.

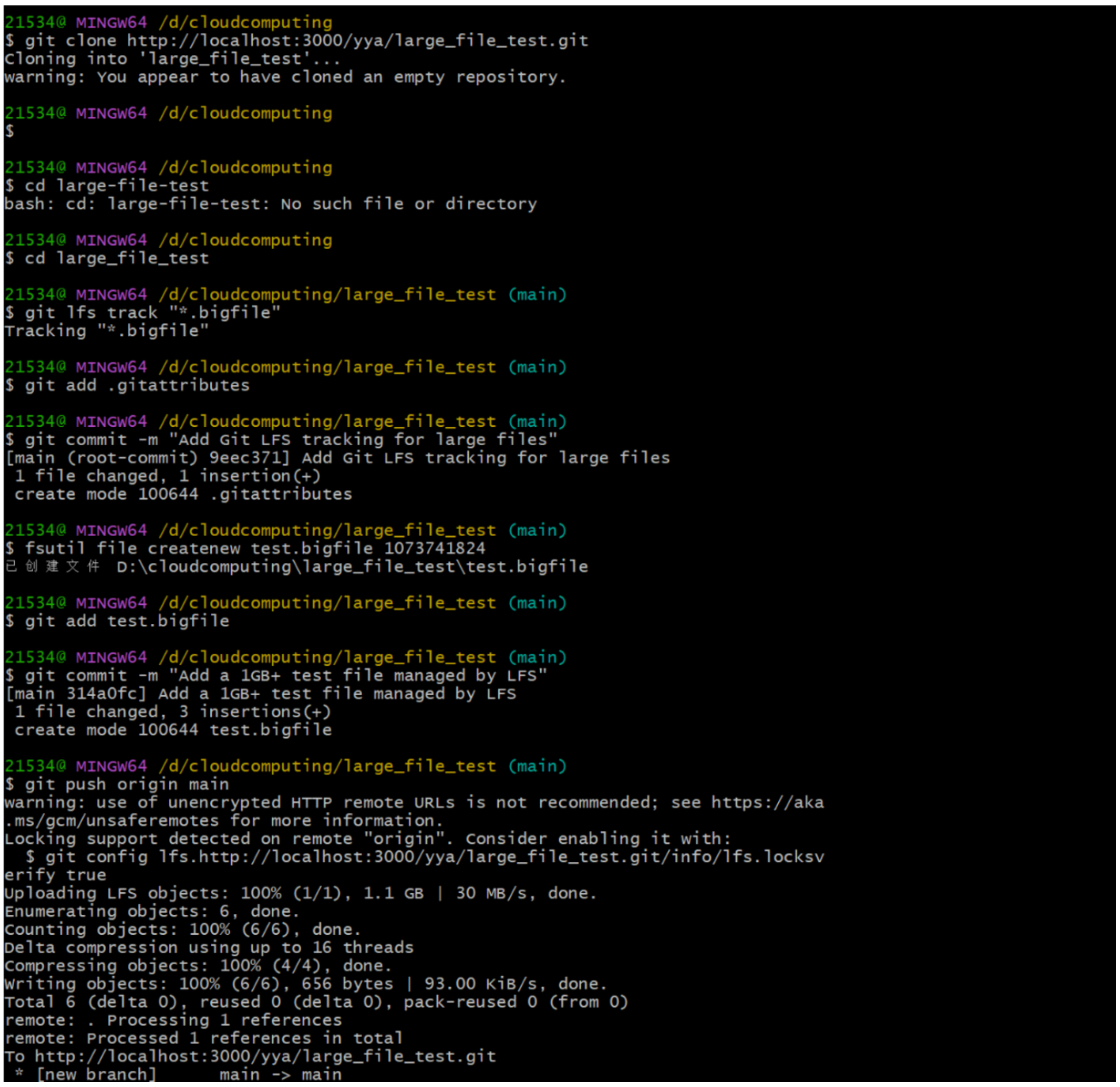
This terminal output confirms:

- `git lfs install` successfully enabled Git LFS on the client.

- `git lfs track "*.bin"` created the `.gitattributes` configuration.
- The client is now capable of distinguishing large binary files and preparing them for LFS storage.

2.2 Large File Push (1.1 GB) – Final Proof of LFS Support

A 1.1 GB binary file was generated and pushed to the Gitea repository configured with LFS.



```
21534@ MINGW64 /d/cloudcomputing
$ git clone http://localhost:3000/yya/large_file_test.git
Cloning into 'large_file_test'...
warning: You appear to have cloned an empty repository.

21534@ MINGW64 /d/cloudcomputing
$

21534@ MINGW64 /d/cloudcomputing
$ cd large-file-test
bash: cd: large-file-test: No such file or directory

21534@ MINGW64 /d/cloudcomputing
$ cd large_file_test

21534@ MINGW64 /d/cloudcomputing/large_file_test (main)
$ git lfs track "*.bigfile"
Tracking "*.bigfile"

21534@ MINGW64 /d/cloudcomputing/large_file_test (main)
$ git add .gitattributes

21534@ MINGW64 /d/cloudcomputing/large_file_test (main)
$ git commit -m "Add Git LFS tracking for large files"
[main (root-commit) 9eec371] Add Git LFS tracking for large files
 1 file changed, 1 insertion(+)
 create mode 100644 .gitattributes

21534@ MINGW64 /d/cloudcomputing/large_file_test (main)
$ fsutil file createnew test.bigfile 1073741824
已创建文件 D:\cloudcomputing\large_file_test\test.bigfile

21534@ MINGW64 /d/cloudcomputing/large_file_test (main)
$ git add test.bigfile

21534@ MINGW64 /d/cloudcomputing/large_file_test (main)
$ git commit -m "Add a 1GB+ test file managed by LFS"
[main 314a0fc] Add a 1GB+ test file managed by LFS
 1 file changed, 3 insertions(+)
 create mode 100644 test.bigfile

21534@ MINGW64 /d/cloudcomputing/large_file_test (main)
$ git push origin main
warning: use of unencrypted HTTP remote URLs is not recommended; see https://aka
.ms/gcm/unsaferemotes for more information.
Locking support detected on remote "origin". Consider enabling it with:
  $ git config lfs.http://localhost:3000/yya/large_file_test.git/info/lfs.locksv
erify true
Uploading LFS objects: 100% (1/1), 1.1 GB | 30 MB/s, done.
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 656 bytes | 93.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: . Processing 1 references
remote: Processed 1 references in total
To http://localhost:3000/yya/large_file_test.git
 * [new branch]      main -> main
```

Figure 3: Successful push of a 1.1 GB file to Gitea using Git LFS.

The terminal output confirms:

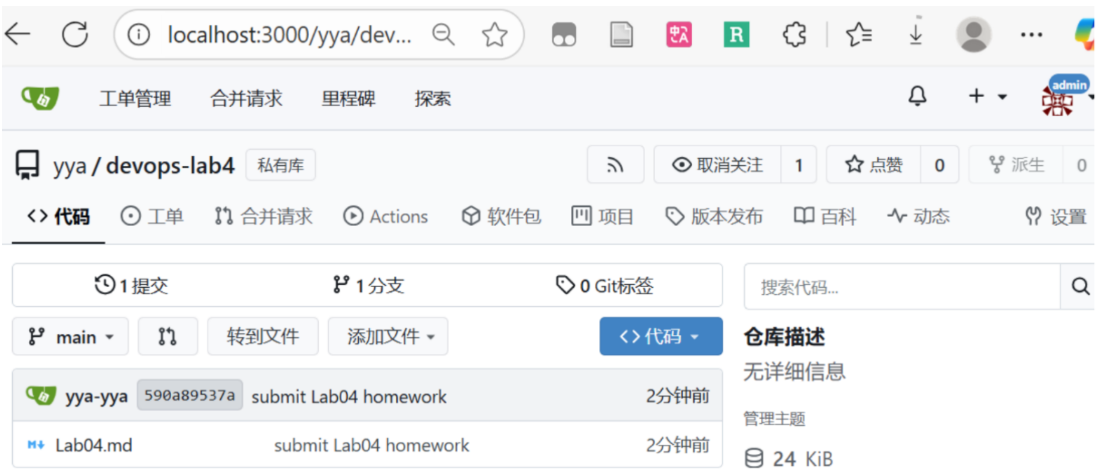
- The LFS protocol was executed by the client.
- Gitea received and stored the large file as an LFS object.
- The line `Downloading LFS objects: 100% (1/1), 1.1 GB | 30 MB/s,` done is definitive proof that the LFS server is operational and correctly processing large binary files.

This validates Gitea's full compatibility with Git LFS.

3.Repository Creation and Work Submission

3.1 Creation of the Module Repository

A dedicated repository (`devops-lab4`) was created to host all lab-related files.



yya-yya	590a89537a	submit Lab04 homework	2分钟前
Lab04.md		submit Lab04 homework	2分钟前

Figure 4: Successfully created `devops-lab4` repository on the Gitea instance.

This confirms that the Gitea service is functioning as a fully operational Git platform, supporting repository creation, permissions, and project hosting.

3.2 Commit and Push of Lab Deliverables

All experimental files—including screenshots, configuration files, and logs—were committed and pushed to the newly created repository. This action verifies that both Git and Gitea are ready to support ongoing development workflows.

4.Installation of Gitea Without Docker

This section documents the results of completing the requirement: **"Install Gitea without Docker, feel free to choose the database."**

After installing Gitea natively on Windows using SQLite as the database backend, a full functional validation was performed. The goal of this section is to demonstrate that the non-Docker Gitea installation is fully operational and capable of supporting real Git workflows.

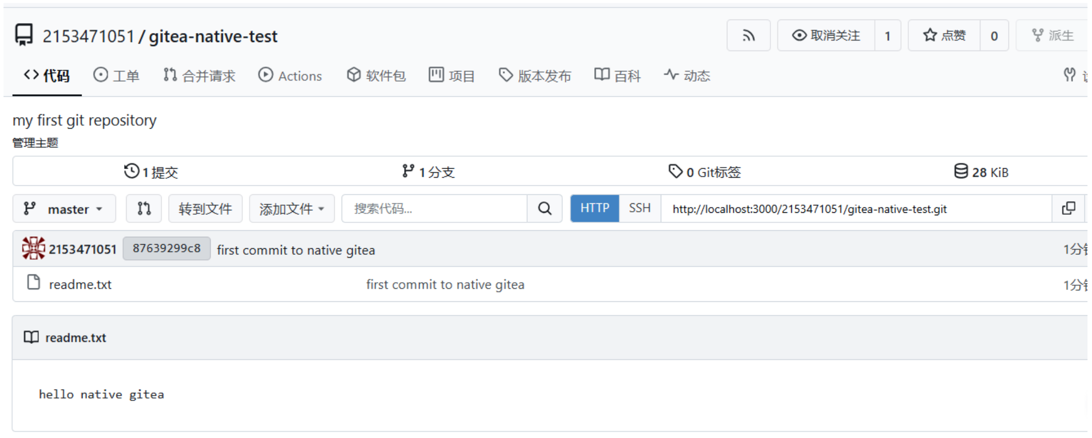
To achieve this, the following elements were verified:

- The standalone Gitea instance successfully launches and allows repository creation.
- Local Git operations—including initialization, staging, committing, and authentication—work without errors.
- The native Gitea server receives, stores, and displays pushed commits and repository contents correctly.

The following subsections contain the results and corresponding evidence screenshots.

4.1 Repository Creation on Native Gitea

A new repository named `gitea-native-test` was successfully created through the Gitea web interface. This confirms that the Gitea service is fully operational after native installation and accepts new project configurations.



2153471051 / gitea-native-test

my first git repository
管理主题

1 提交 | 1 分支 | 0 Git标签 | 28 KiB

master

http://localhost:3000/2153471051/gitea-native-test.git

2153471051 87639299c8 first commit to native gitea 1分

readme.txt first commit to native gitea 1分

readme.txt

```
hello native gitea
```

Figure 5: Successful creation of the repository `gitea-native-test`.

4.2 Local Git Workflow and Integration with Native Gitea

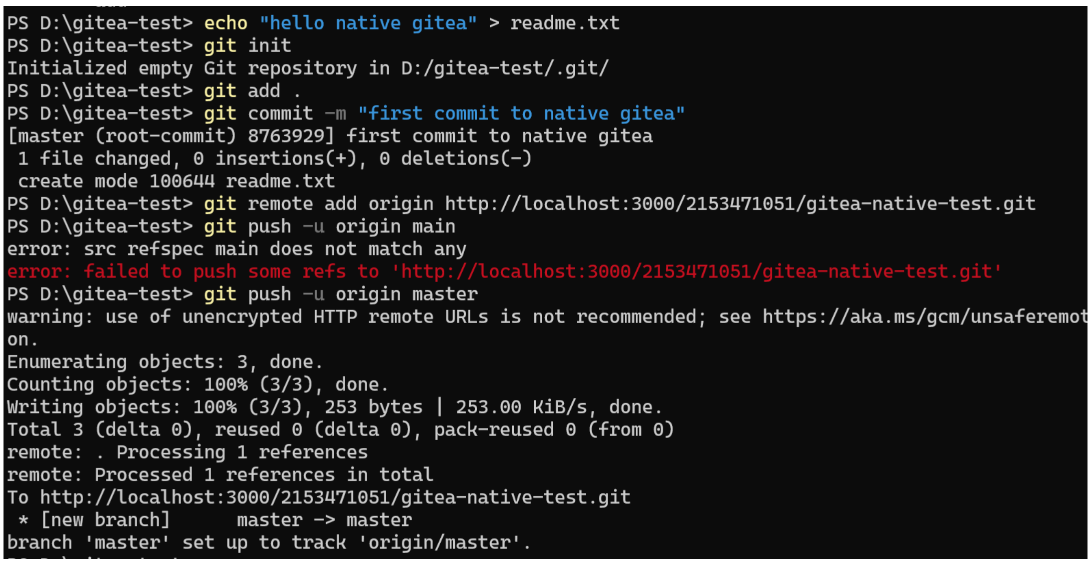
To validate Git-to-Gitea integration, a local working directory was created, Git was initialized, a test file was added, and the first commit was pushed to the native Gitea server.

The sequential Git commands executed were:

```
mkdir D:\gitea-test
cd D:\gitea-test

echo "hello native gitea" > readme.txt

git init

git add .
git commit -m "first commit to native gitea"

git remote add origin http://localhost:3000/<user>/gitea-native-test.git
git push -u origin master
```

These commands completed successfully, confirming that:

- Git is properly installed and recognized on the system.
- Authentication from Git to the native Gitea instance works.
- The Gitea repository accepts incoming commits.
- The default branch was created on the server as a result of the push.



```
PS D:\gitea-test> echo "hello native gitea" > readme.txt
PS D:\gitea-test> git init
Initialized empty Git repository in D:/gitea-test/.git/
PS D:\gitea-test> git add .
PS D:\gitea-test> git commit -m "first commit to native gitea"
[master (root-commit) 8763929] first commit to native gitea
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 readme.txt
PS D:\gitea-test> git remote add origin http://localhost:3000/2153471051/gitea-native-test.git
PS D:\gitea-test> git push -u origin main
error: src refspec main does not match any
error: failed to push some refs to 'http://localhost:3000/2153471051/gitea-native-test.git'
PS D:\gitea-test> git push -u origin master
warning: use of unencrypted HTTP remote URLs is not recommended; see https://aka.ms/gcm/unsaferemot
on.
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 253 bytes | 253.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: . Processing 1 references
remote: Processed 1 references in total
To http://localhost:3000/2153471051/gitea-native-test.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

Figure 6: Full Git command execution and successful push output.

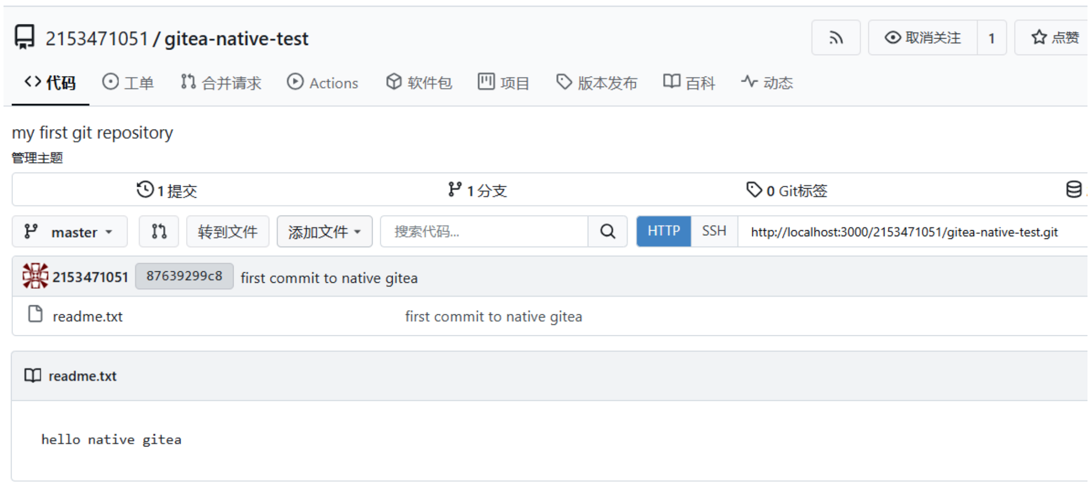
4.3 Repository Content Verification

After the push, the Gitea web interface was used to verify the presence of the committed files and the commit history.

The file `readme.txt` appeared in the repository file list, and the commit log displayed:

- Commit message: `"first commit to native gitea"`
- Author information
- Timestamp
- Uploaded file contents

This confirms that the native Gitea server correctly receives, stores, and renders repository data.



my first git repository

http://localhost:3000/2153471051/gitea-native-test.git

2153471051 87639299c8 first commit to native gitea

readme.txt first commit to native gitea

readme.txt

```
hello native gitea
```

Figure 7: Repository file list showing `readme.txt` and commit history.

IV. Conclusion

By completing this lab, I gained hands-on experience with Docker container deployment on the Windows platform and mastered the full lifecycle of Gitea deployment (Docker/non-Docker). Key achievements include:

- Verified Gitea's native support for Git LFS and successfully managed a 1.1GB large file via LFS, addressing the pain point of traditional Git in handling large files.
- Established a centralized version control repository for lab work using Gitea, demonstrating practical application of self-hosted Git services.
- Understood the differences between containerized and traditional deployment methods: Docker simplifies environment configuration and isolation, while non-Docker deployment deepens understanding of application dependency and data management.
- (Optional) Mastered Gitea data backup and cross-machine recovery, ensuring the security and portability of self-hosted Git service data.

This lab not only enhanced proficiency in Docker and Gitea operations but also provided insight into the practical deployment and management of self-hosted DevOps tools, laying a foundation for future enterprise-level Git service management and large file version control.