ZERO TO HERO
DEVOPS & CLOUD ENGINEERING

GITLAB ACCOUNT & GIT SETUP MANUAL

Practical Student Guide

Training Environment: VS Code + Git Bash

All Git and GitLab command-line activities in this module are performed from the Git Bash terminal integrated into VS Code.

Module Objective

By the end of this module, you will have:

- Created a GitLab account
- Verified your email address
- Secured your account
- Installed and verified Git
- Configured Git with your identity
- Created an SSH key
- Connected your computer to GitLab using SSH
- Created your first GitLab repository
- Cloned the repository into VS Code
- Made your first Git commit
- Pushed your code to GitLab

IMPORTANT: You do not need to open a separate Git Bash window. Open Git Bash from the integrated terminal inside VS Code.

1. What Is Git?

Git is a version control system. It allows you to keep track of changes made to your code.

Why do DevOps Engineers need Git? Almost everything in modern DevOps involves code, including application source code, Terraform, Kubernetes YAML, Dockerfiles, CI/CD pipelines, shell scripts, Ansible and configuration files.

2. What Is GitLab?

GitLab is a DevOps platform that uses Git repositories to help teams manage software development and delivery.

Git and GitLab are not the same thing. Git is the version control system; GitLab is a platform that provides Git repositories plus collaboration, CI/CD, security and other DevOps capabilities.

3. Our Development Environment

For this course, we use Windows → VS Code → Git Bash. Git Bash provides a Unix-style command-line environment on Windows and lets us work with Git and other DevOps tools from one integrated terminal.

```
VS Code
  ↓
Git Bash Terminal
  ↓
Git / GitLab / AWS CLI / Terraform / Docker / other DevOps tools
```

4. Open Git Bash in VS Code

Open Visual Studio Code. From the top menu select Terminal → New Terminal. At the terminal window, select the terminal profile dropdown and choose Git Bash.

You should see a prompt containing something similar to MINGW64. You are now ready to work.

5. Verify Git

In the VS Code Git Bash terminal, run:

```
git --version
```

You should receive a response similar to:

```
git version 2.x.x
```

If Git is not recognised, resolve the Git installation and PATH configuration before continuing.

6. Create Your GitLab Account

Open your browser and go to GitLab: https://gitlab.com/

Click Register and create your account using your email address.

7. Choose Your GitLab Username

Choose a professional username, such as johnsmith, johnsmith-devops, johncloud or john.devops.

Your username can appear in your GitLab URLs and may eventually form part of your professional portfolio.

8. Verify Your Email

Check your email for the GitLab verification message. If you cannot find it, check Spam/Junk and request another verification email if necessary.

9. Log Into GitLab

Return to https://gitlab.com/ and click Sign in. Enter your credentials and confirm that you can access your GitLab dashboard.

10. Secure Your GitLab Account

As a DevOps Engineer, security is extremely important. Your GitLab account may eventually have access to source code, CI/CD pipelines, cloud infrastructure and production systems.

Enable Two-Factor Authentication from your GitLab account settings under Password and authentication. Use an authenticator application where appropriate.

11. Configure Git

Return to VS Code and make sure the integrated terminal is Git Bash.

```
git config --global user.name "Your Name"

git config --global user.email "your@email.com"
```

Use the email address associated with your GitLab account.

12. Verify Your Git Configuration

```
git config --global --list
```

You should see user.name and user.email entries. You can also check them individually:

```
git config --global user.name
git config --global user.email
```

13. What Is SSH?

SSH stands for Secure Shell. It provides a secure way for your computer to communicate with another computer or service. We will use SSH to securely connect your computer to GitLab.

```
YOUR COMPUTER
      |
      | SSH
      ▼
   GITLAB
```

14. Create an SSH Key

In the VS Code Git Bash terminal, run:

```
ssh-keygen -t ed25519 -C "your@email.com"
```

Replace the email with your GitLab email. When asked where to save the key, press ENTER to accept the default location. You may also be asked for a passphrase.

15. Understand Your SSH Keys

```
~/.ssh/

id_ed25519       ← PRIVATE KEY
id_ed25519.pub   ← PUBLIC KEY
```

The private key must never be shared. The public key is the key you add to GitLab.

SECURITY RULE: Never send your private key to your instructor, another student, GitLab, WhatsApp, email or anyone else.

16. Copy Your Public SSH Key

```
cat ~/.ssh/id_ed25519.pub
```

Copy the entire line displayed, beginning with ssh-ed25519.

17. Add the SSH Key to GitLab

In GitLab, go to your profile settings and navigate to Access → SSH Keys. Click Add new key. Paste your public key, give it a meaningful title such as VS Code Git Bash - My PC, and click Add key.

18. Test the GitLab SSH Connection

```
ssh -T git@gitlab.com
```

The first time you connect, you may be asked whether you want to continue connecting. Type yes. If configured correctly, GitLab should respond with a welcome message.

SUCCESS: Your computer is now authenticated with GitLab using SSH.

19. Create Your First GitLab Project

In GitLab select New project → Create blank project.

Use the project name: zero-to-hero-devops. For this first training project, select Private, then create the project.

20. Clone Your Project into VS Code

GitLab will provide an SSH repository address similar to git@gitlab.com:yourusername/zero-to-hero-devops.git. Copy it.

In the VS Code Git Bash terminal, navigate to where you want your projects stored. For example:

```
cd ~/Documents
```

Then clone the repository:

```
git clone git@gitlab.com:yourusername/zero-to-hero-devops.git
```

21. Enter the Project

```
cd zero-to-hero-devops
pwd
ls
```

You are now working inside your Git repository.

22. Open the Project in VS Code

```
code .
```

This opens the current project in VS Code.

23. Create Your First File

In VS Code, create a file called README.md and add:

```
# Zero to Hero DevOps

My journey into DevOps and Cloud Engineering.

## Technologies

- Linux
- Git
- GitLab
- AWS
- Terraform
- Docker
- Kubernetes
```

Save the file.

24. Check Git Status

```
git status
```

Git should show README.md as an untracked file. This means Git knows the file exists but is not yet tracking it.

25. Stage the File

```
git add README.md
```

Or stage everything in the current directory:

```
git add .
```

Check again:

```
git status
```

26. Commit Your Changes

```
git commit -m "Add DevOps README"
```

A commit records a snapshot of your changes in Git history.

27. Push Your Code to GitLab

```
git push
```

Return to your GitLab project in your browser and refresh the page. You should now see README.md.

Congratulations — you have successfully pushed your first code to GitLab.

28. The Basic Git Workflow

```
CREATE / MODIFY FILE
        ↓
   git status
        ↓
    git add
        ↓
   git commit
        ↓
    git push
        ↓
     GITLAB
```

You will use this workflow constantly throughout your DevOps career.

29. Essential Git Commands

git --version — Check Git installation
git config — Configure Git
git status — See what has changed
git add — Stage changes
git commit — Record changes
git push — Upload changes to GitLab
git pull — Download changes from GitLab
git clone — Copy a repository
git log — View commit history
git branch — View/manage branches
git switch — Switch branches

30. What Have We Built?

```
                GITLAB
                  ▲
                  |
                SSH
```

```
        │
        ▼
VS CODE + GIT BASH
        │
        ▼
YOUR PROJECT
```

You now have a professional foundation for working with Git and GitLab.

Practical Assignment

Create a GitLab repository called:

```
devops-training
```

Clone it into VS Code and create this structure:

```
devops-training/
│
├── README.md
│
└── notes/
    └── git-basics.md
```

README.md

```
# DevOps Training

Welcome to my Zero to Hero DevOps and Cloud Engineering journey.

## Tools I am learning

- Git
- GitLab
- AWS
- Terraform
- Docker
- Kubernetes
```

notes/git-basics.md

```
# Git Basics

Git is a distributed version control system.

GitLab is a platform for hosting Git repositories and managing DevOps workflows.

## Commands I have learned

git status
git add
git commit
git push
git pull
git clone
```

Then run:

```
git status
git add .
git commit -m "Add Git basics training notes"
git push
```

Finally, open GitLab and confirm that both files are visible.

Module Completion Checklist

☐ I have a GitLab account.

☐ My email is verified.

☐ I have secured my account with 2FA.

☐ Git is installed.

☐ I can use Git from the VS Code Git Bash terminal.

☐ My Git username is configured.

☐ My Git email is configured.

☐ I understand public and private SSH keys.

☐ My SSH key is connected to GitLab.

☐ I can successfully run ssh -T git@gitlab.com.

☐ I can create a GitLab repository.

☐ I can clone a repository.

☐ I can create files in VS Code.

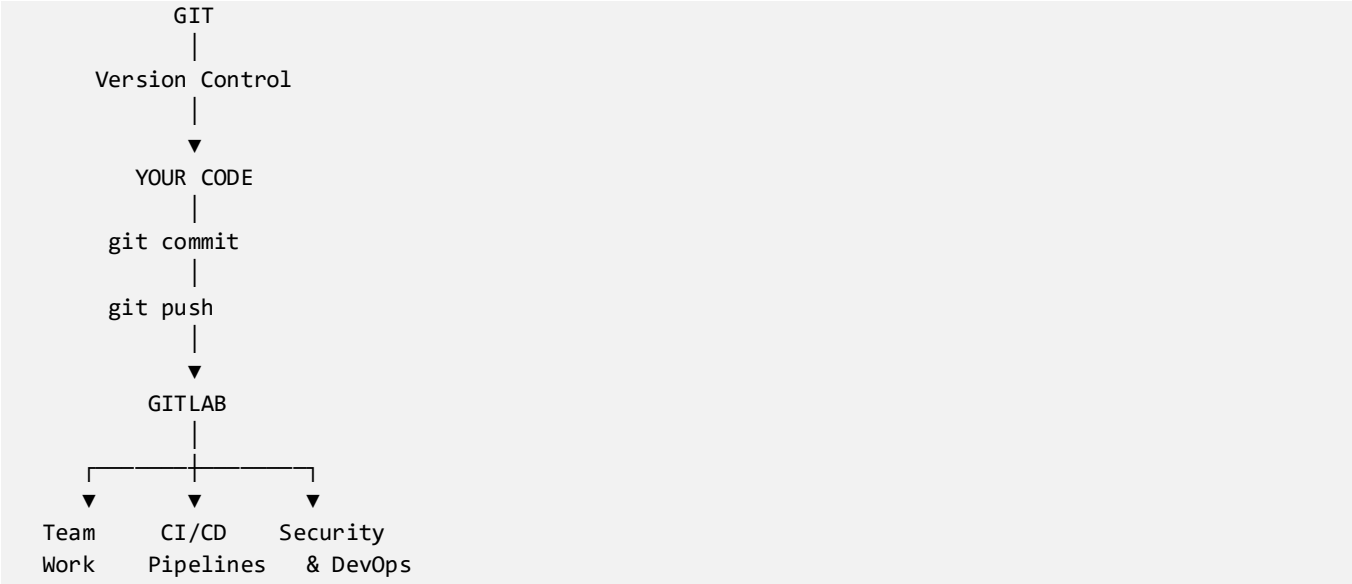
☐ I understand git add.

☐ I understand git commit.

☐ I understand git push.

☐ I can see my pushed files in GitLab.

The Big Picture



Git is one of the foundations of modern DevOps. Master this workflow, because almost everything we build later in this course will eventually connect back to Git and GitLab.