

**DEVOPS**

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE & ENGINEERING**

Name of student

Adnan Rasool

Submitted to

**Dr. Prateek Raj Gautam**

**SCHOOL OF COMPUTER SCIENCE**

**UNIVERSITY OF PETROLEUM & ENERGY STUDIES**

# Assignment 1: Version Control

# Subversion (SVN)

Apache Subversion (often referred to as SVN) is a version control system that allows you to manage and track changes to your files and directories over time. Below is a step-by-step guide to install and use Subversion.

## Install Apache Subversion

# Linux

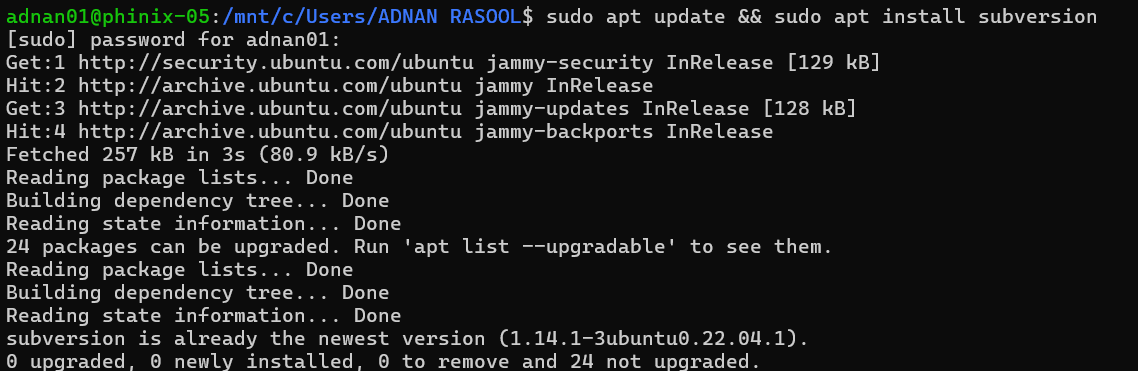
## 1. Setting Up SVN Server with svnserve

**Step 1: Install Subversion**

On a Linux system, install Subversion using your package manager:

sudo apt update

sudo apt install subversion

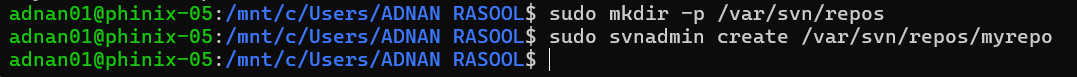


**Step 2: Create a Repository**

Create a directory for your repositories and initialize a new repository:

sudo mkdir -p /var/svn/repos

sudo svnadmin create /var/svn/repos/myrepo



**Step 3: Configure svnserve**

Edit the svnserve.conf file to configure access:

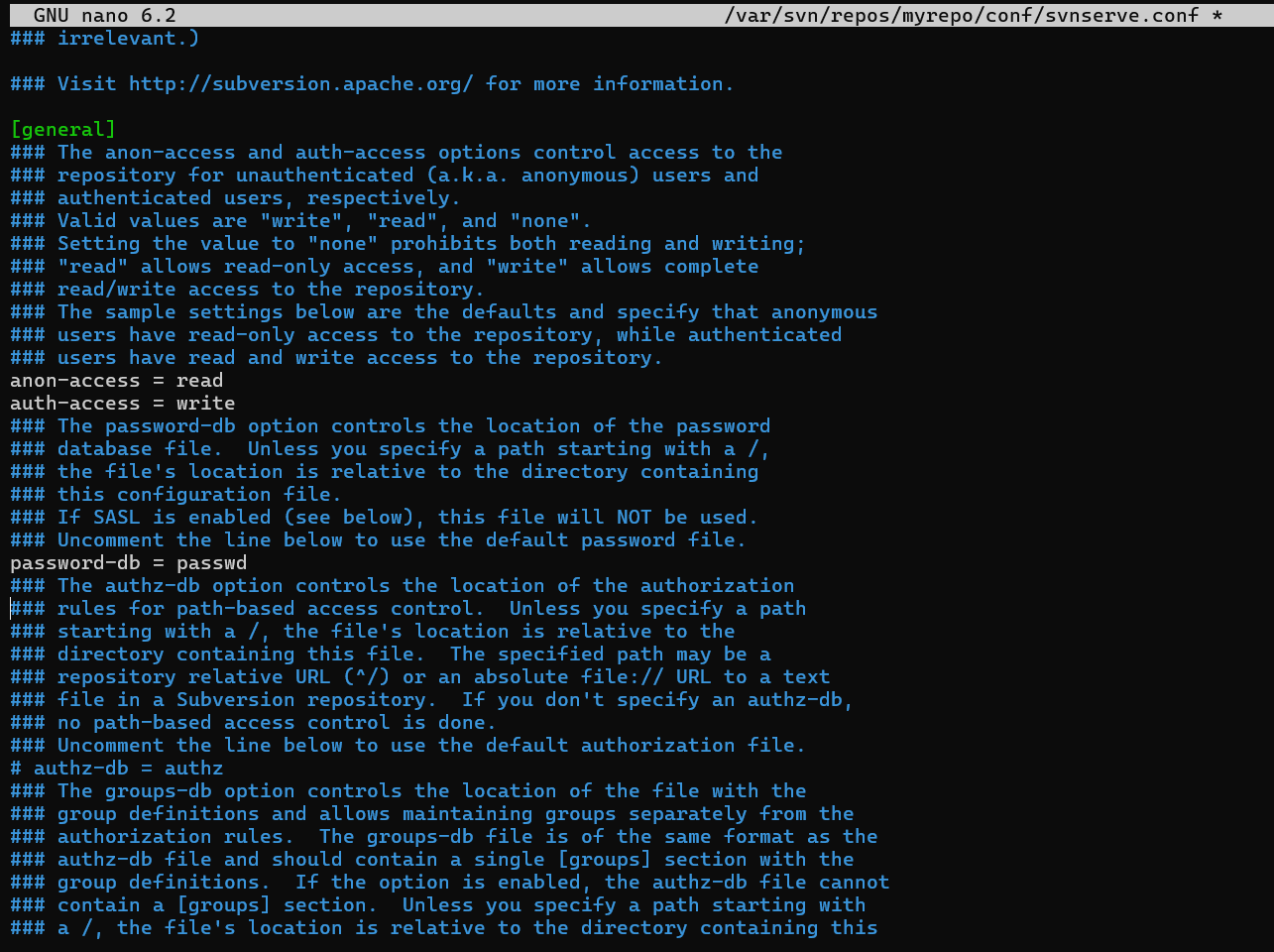
sudo nano /var/svn/repos/myrepo/conf/svnserve.conf

Make the following changes:

password-db = passwd # Use the passwd file for authentication

auth-access = write # Allow authenticated users to write

anon-access = none # Disable anonymous access

****

**Step 4: Set Up Users**

Edit the passwd file to add users:

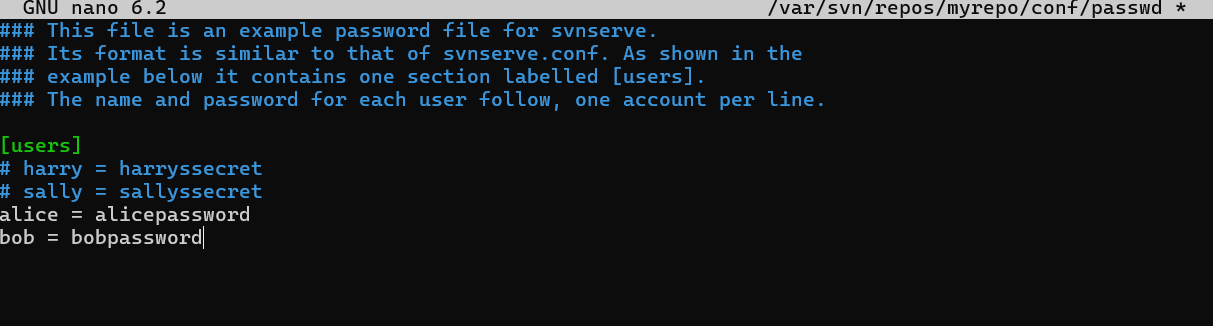
sudo nano /var/svn/repos/myrepo/conf/passwd

Add a user:

alice = alicepassword

bob = bobpassword

bob = bobpassword



**Step 5: Start svnserve**

Start the svnserve daemon:

sudo svnserve -d -r /var/svn/repos

The -d flag runs it in daemon mode, and -r specifies the root directory for repositories.

## 2. Common SVN Commands

**Task 1: Check Out a Repository**

To check out the repository to your local machine:

svn checkout svn://localhost/myrepo --username alice

You'll be prompted for the password.

**Task 2: Add Files to the Repository55**

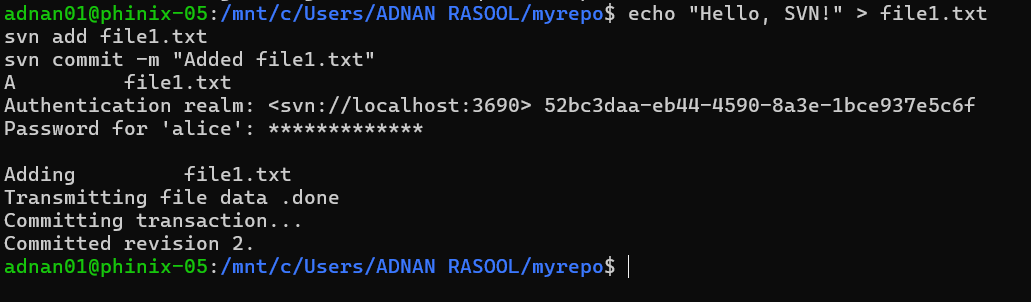
Create a new file and add it to the repository:

cd myrepo

echo "Hello, SVN!" > file.txt

svn add file.txt

svn commit -m "Added file.txt"



**Task 3: Update Your Working Copy**

To update your working copy with the latest changes:

svn update

**Task 4: View Repository Status**

Check the status of your working copy:

svn status

**Task 5: View Change History**

View the commit history of the repository:

svn log

****

**Task 6: Create the trunk Directory:**

Use the command svn mkdir to create trunk the directory:

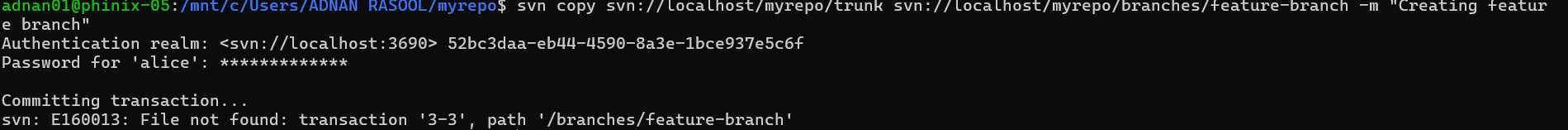
svn mkdir svn://localhost/myrepo/trunk -m "Creating trunk directory"



**Task 7: Create a Branch**

Create a branch for new development:

svn copy svn://localhost/myrepo/trunk svn://localhost/myrepo/branches/feature-branch -m "Creating feature branch"

****

**Task 8: Switch to a Branch**

Switch your working copy to the new branch:

svn switch svn://localhost/myrepo/branches/feature-branch

**Task 9: Merge Changes**

Merge changes from the branch back to the trunk:

svn merge svn://localhost/myrepo/branches/feature-branch

svn commit -m "Merged feature-branch into trunk"

**Task 10: Resolve Conflicts**

If there are conflicts during an update or merge, resolve them:

svn resolve --accept working file.txt

**Task 11: Delete a File**

Delete a file and commit the change:

svn delete file.txt

svn commit -m "Deleted file.txt"

# Mercurial

Mercurial is a **distributed version control system (DVCS)** designed for efficient handling of projects of all sizes. It is similar to Git in functionality but emphasizes simplicity and ease of use. Mercurial is written in Python and is known for its **intuitive commands**, **robust performance**, and **cross-platform compatibility**. It is widely used in both open-source and enterprise projects.

**Key Features of Mercurial:**

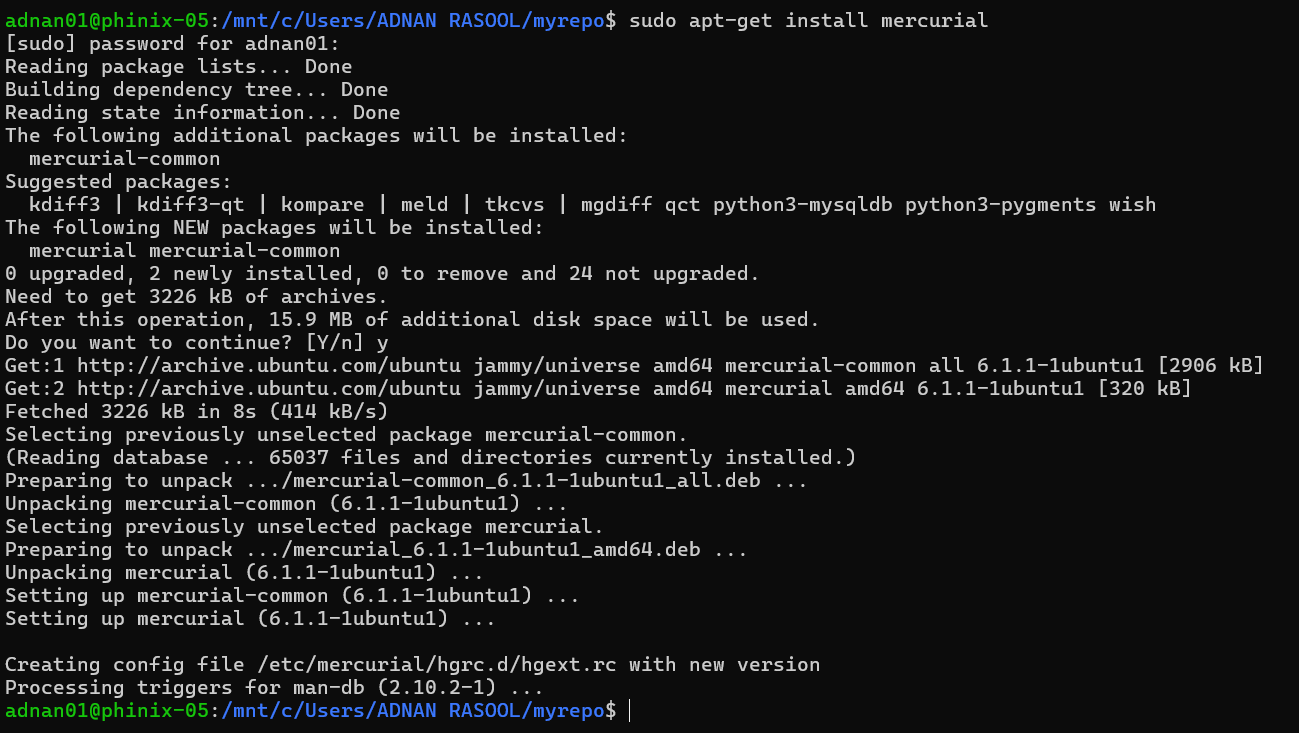
1. Distributed Version Control: Every developer has a full copy of the repository, enabling offline work and independent branching.
2. Lightweight and Fast: Efficient handling of large projects and binary files.
3. Simple and Intuitive Commands: Commands are easy to learn and use, with a consistent syntax.
4. Cross-Platform: Works on Windows, macOS, and Linux. 5. Extensible: Supports plugins for additional functionality.
5. Secure: Uses SHA-1 hashing for data integrity and supports signed commits.

## How to Use Mercurial

# Linux

**1.Installation**

sudo apt-get install mercurial



**2.Initialize a Repository:**

cd my\_project

hg init my\_project

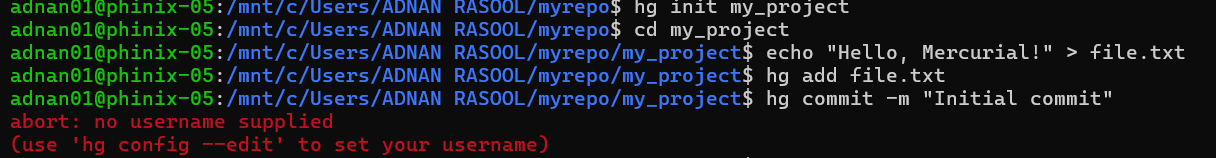
**3.Add Files:**

hg add file.txt

echo "Hello, Mercurial!" > file.txt

**4.Commit Changes:**

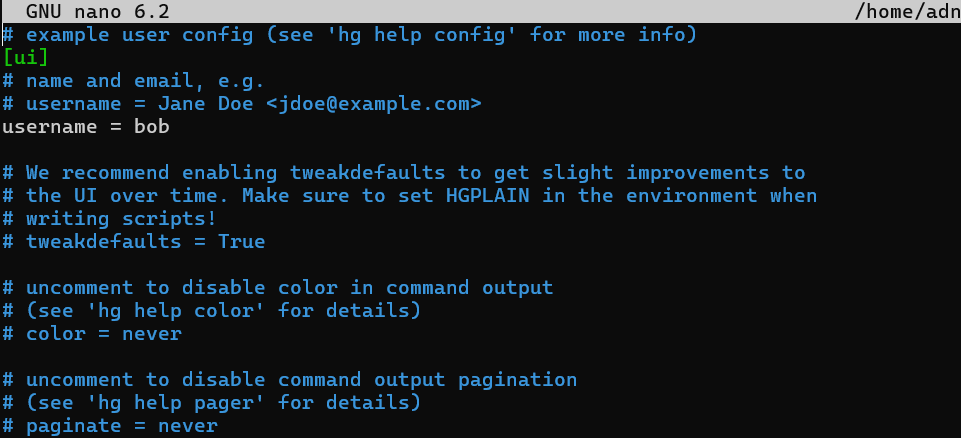
hg commit -m "Initial commit"



If the above error occurs than there is no username supplied.

Use hg config - -edit to give a username (choose 1 here):

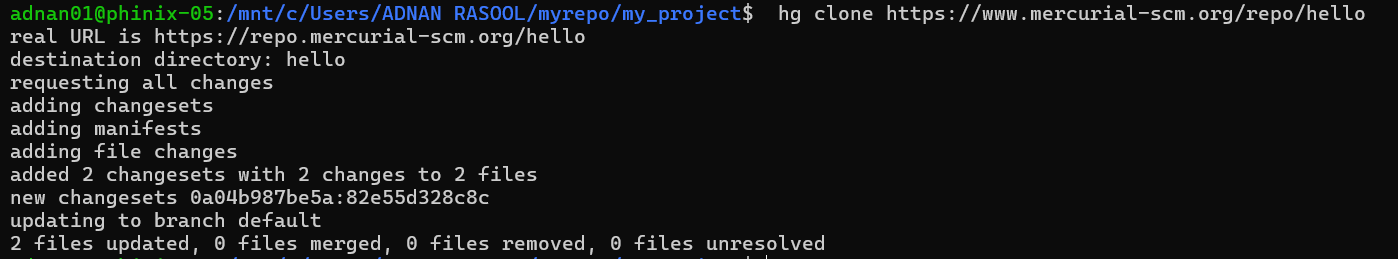
hg config --edit





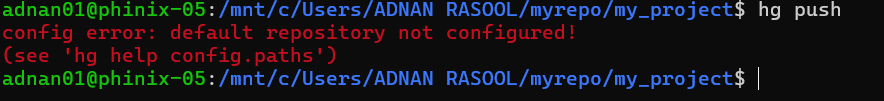
**5.Clone a Remote Repository:**

hg clone https://www.mercurial-scm.org/repo/hello



**6.Push Changes to Remote:**

hg push



If you encounter this error that means that there is no remote server into which we can push our changes

If you want to push your changes than you can connect to any free remote servers available on internet.