Team Alton

Alton Matovu
Department of Computer Science
Pacific Lutheran University
Tacoma, United States
alton.matovu@plu.edu

Linux User Account Automation

Abstract
This project aims to automate Linux user account provisioning using Bash scripting and CSV data. The automation removes the need for manual account creation, group assignment, directory setup, and SSH login configuration. The system reads a CSV list of users and creates Linux user accounts, assigns proper groups, generates /home/<user>/www web directories, and produces secure key based authentication using SSH. The script ensures efficiency and prevents mistakes that occur when users are configured individually. This solution is useful for IT environments, cybersecurity labs, and organizations needing onboarding automation to manage multiple users. The project also demonstrates Linux system administration skills and Bash file automation that focuses on simplicity and practical use.

Keywords
Linux automation, Bash, SSH, CSV, Apache2, user provisioning

I. INTRODUCTION

Managing user accounts and permissions in Linux environments can be difficult, especially when working with multiple users. When accounts are set up manually, mistakes happen easily. System administrators may forget to set permissions, misassign groups, or fail to secure accounts. These mistakes can lead to users losing access, systems being exposed to security risks, and administrators wasting time going back to fix something simple.

The goal of this project is to remove this problem by providing a Bash automation script that processes user information from a CSV file. The script reads usernames and roles, creates the Linux group if it does not already exist, and then creates the user. After user creation, the tool assigns the correct group ownership and builds a personal Apache2 web directory located at /home/<user>/www. This directory allows the user to host website files separately from others. The script also generates SSH key pairs automatically to allow secure login without typing passwords. This reduces credential risks and prevents attacks like brute force guessing.

The overall design reflects tasks often done in professional IT or cybersecurity environments where there are many machines and many people. Automation saves time and keeps rules consistent. By combining Bash scripting, Linux utilities, Apache2, and SSH key systems, the tool provides a real demonstration of what onboarding systems can look like in real organizations.

II. BACKGROUND

The project builds on four core technologies: Bash scripting, Apache2 web servers, SSH authentication, and CSV data input. Each tool brings a specific function into the automation pipeline.

A. Bash

Bash is a default shell used in Linux systems. It allows command automation and provides access to utilities like useradd, groupadd, chmod, mkdir, and file permissions. Bash scripting allows the system to perform the same commands every time without human error. The goal is to use condition checks, loops, and branching logic to manage users.

For example, the script checks if a group exists using getent group. If it does not exist, the tool runs sudo groupadd <groupName>. For users, it checks using id <username>. If the system returns an error, the script will create the account. By structuring commands this way, Bash prevents duplicate groups, avoids creating users multiple times, and manages folders automatically. Bash also works directly with Linux file paths and can set permission values using chmod 755 or ownership using chown user:group. These small moves are foundational to every Linux deployment.

One important feature of the script is error handling. The process logs whether a user already exists, whether a group was already created, and whether directories need to be made. This is similar to how real systems work. Bash is useful because it is not heavy or complicated. It does not require frameworks or additional packages. It relies on built in Linux tools that nearly every system has.

B. Apache2

Apache2 is a web server that allows public directory hosting. The project uses the Apache2 UserDir module to give each account a personal website. The script creates directories using the format /home/<user>/www. Permissions are applied so only the user has full access to edit content.

Apache2 hosting allows each person to maintain a website under their own username. The format looks like:

http://<server>/~username

This gives students, employees, or members a way to share project files or custom pages without needing root access. The tool replicates a common academic environment where universities host public pages for computer science classes or student projects.

In order to function correctly, Apache2 requires ownership and group permissions to be set exactly. A mistake like chmod 777 can expose a directory to anyone. The project maintains strict folder and SSH key restrictions so that users do not accidentally break or misuse the system.

C. SSH

SSH is secure remote login software. Instead of using passwords, this project uses key pair authentication. The Bash script runs ssh-keygen to create two keys. The private key remains with the user while the public key is placed into ~/.ssh/authorized_keys.

This makes login faster and much safer because nobody can guess a password. Intrusions become harder because an attacker would need access to the private key file itself. The script assigns directory permissions as well, using values such as:

.ssh → 700

id_rsa → 600

id_rsa.pub → 644

These permissions prevent others from reading private keys. SSH keys are widely used in DevOps, cybersecurity labs, and professional infrastructure.

D. CSV Input

The CSV spreadsheet is the main source of user data. Basic columns include usernames and group names. The script processes the file line by line so that a list of users can be provisioned in a single action. CSV files are easy to edit and easy to share among teams.

The script uses IFS=, to split each entry into variables. The automation builds accounts without needing manual checks. This makes onboarding new employees or students much simpler.

III. RELATED WORKS

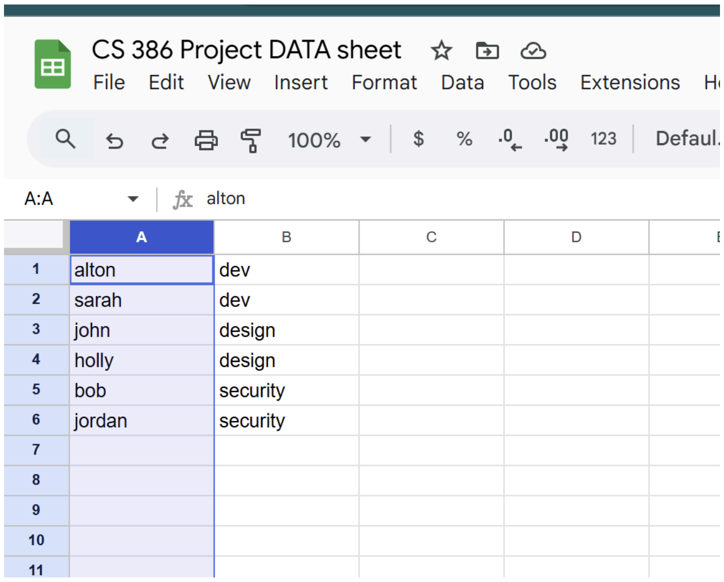
There are many tools that automate Linux system administration. Some of the most common tools include Ansible, Puppet, and Chef. These tools let administrators manage hundreds of machines by writing configuration files in YAML or JSON. They allow powerful automation but require large environments and extra setup. They are useful for enterprise groups but too heavy for smaller student projects or small companies.

The difference with this project is that everything is handled directly through Bash. That gives the user complete control of Linux commands. Many online tutorials show how to create users with CSV files, but they stop at account creation. They do not configure Apache2 personal directories or generate SSH keys. This project combines these steps to produce a simple one script provisioning tool, which is practical and accessible to beginners.

IV. EXPERIMENT AND DEMO

This section demonstrates the entire workflow using screenshots. Each screenshot should be inserted in the order shown.

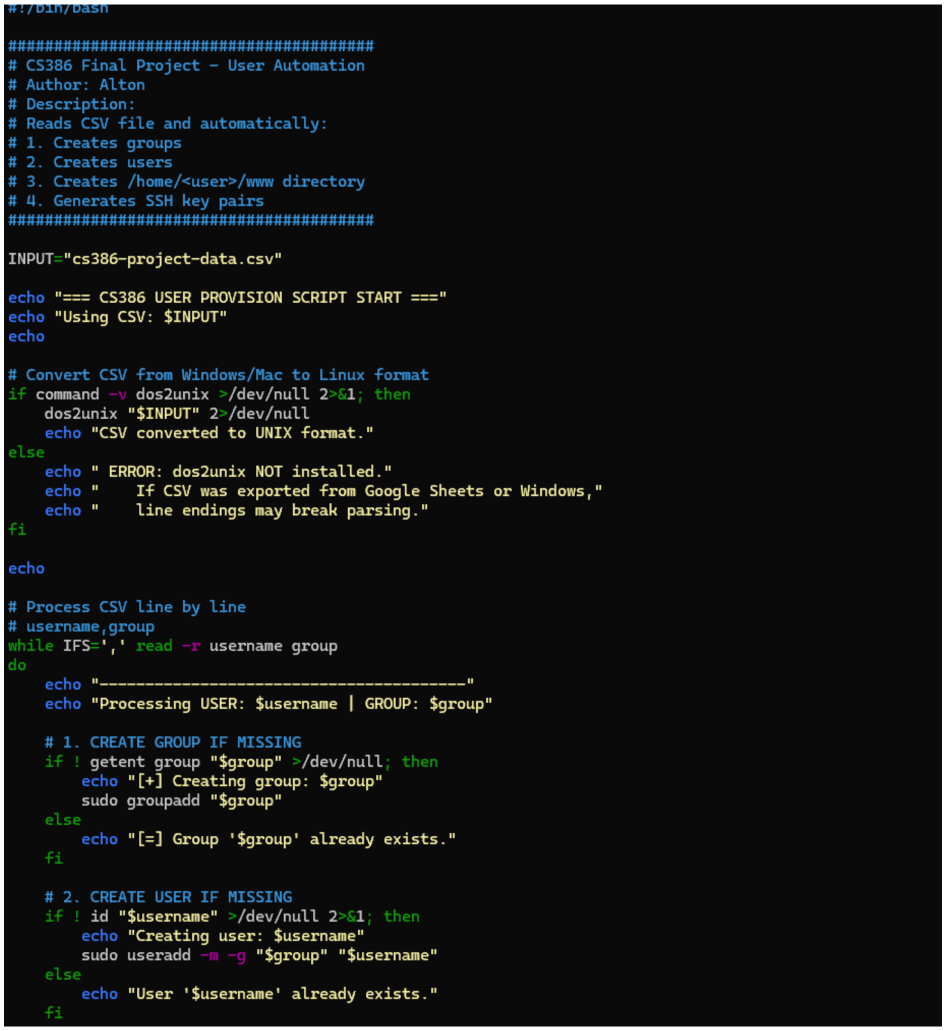
A. CSV Input Example


CS 386 Project DATA sheet
File Edit View Insert Format Data Tools Extensions
100%
Defaul
A:A
alton

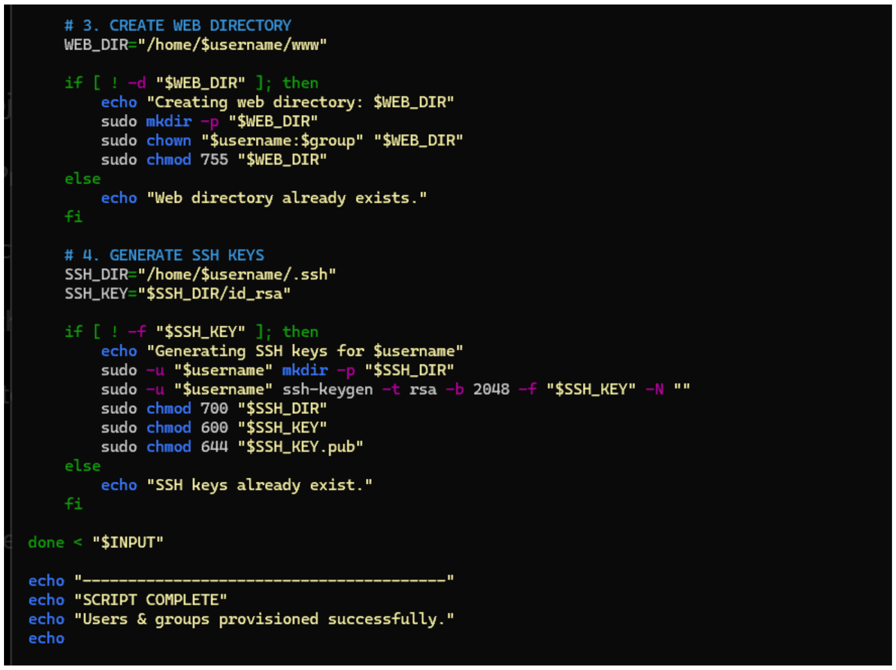
	A	B	C	D
1	alton	dev		
2	sarah	dev		
3	john	design		
4	holly	design		
5	bob	security		
6	jordan	security		
7				
8				
9				
10				
11				

The CSV file includes columns that list users and their group assignments. The script reads each row and prepares to assign the proper access and permissions. Example entries show alton → dev, sarah → dev, john → design, and bob → security. This allows multiple departments to be handled in a single execution.

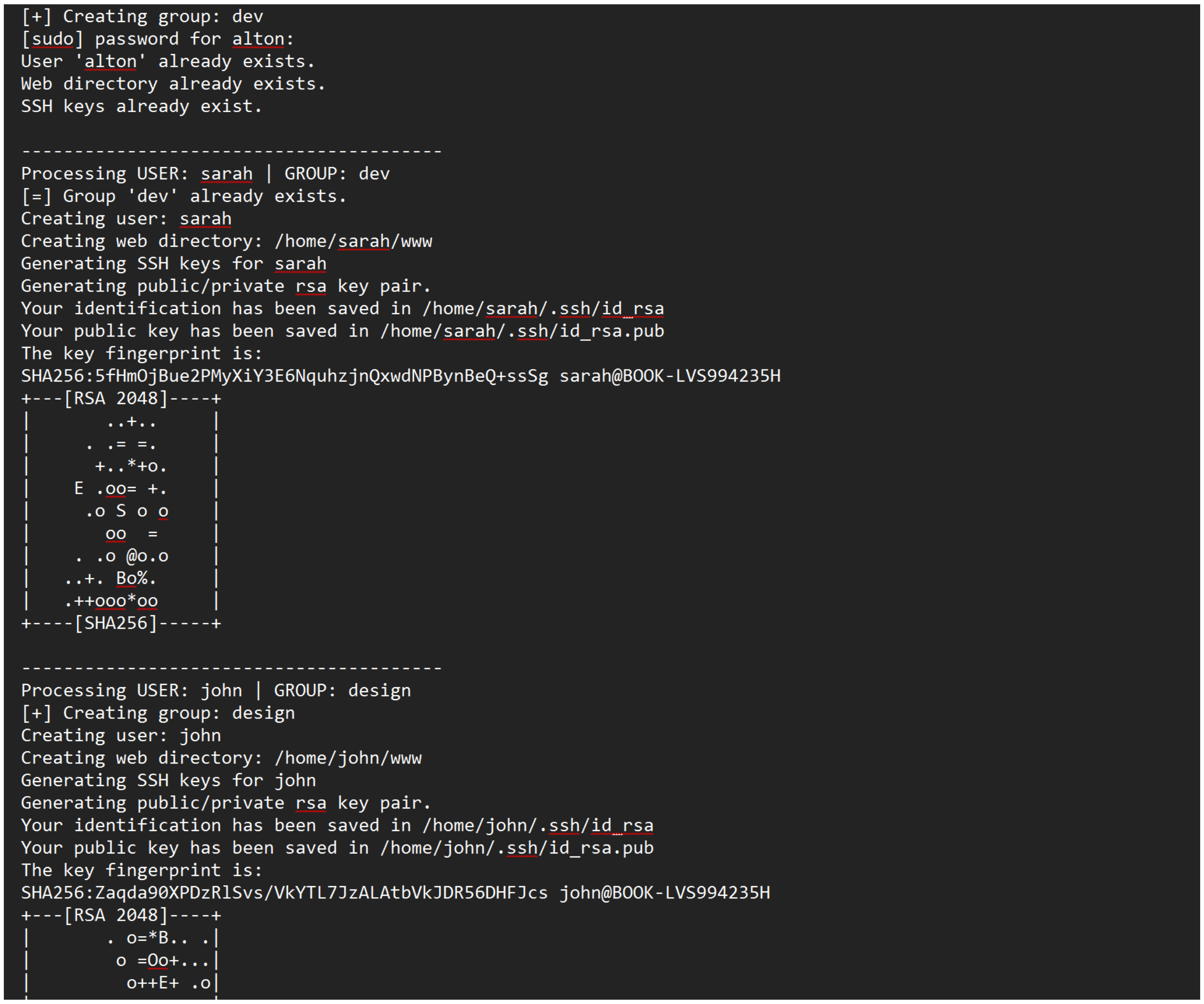
B. Script Execution



```
#!/bin/bash

########################################
# CS386 Final Project - User Automation
# Author: Alton
# Description:
# Reads CSV file and automatically:
# 1. Creates groups
# 2. Creates users
# 3. Creates /home/<user>/www directory
# 4. Generates SSH key pairs
########################################

INPUT="cs386-project-data.csv"

echo "=== CS386 USER PROVISION SCRIPT START ==="
echo "Using CSV: $INPUT"
echo

# Convert CSV from Windows/Mac to Linux format
if command -v dos2unix >/dev/null 2>&1; then
    dos2unix "$INPUT" 2>/dev/null
    echo "CSV converted to UNIX format."
else
    echo " ERROR: dos2unix NOT installed."
    echo "    If CSV was exported from Google Sheets or Windows,"
    echo "    line endings may break parsing."
fi

echo

# Process CSV line by line
# username,group
while IFS=',' read -r username group
do
    echo "----------------------------------------"
    echo "Processing USER: $username | GROUP: $group"

    # 1. CREATE GROUP IF MISSING
    if ! getent group "$group" >/dev/null; then
        echo "[+] Creating group: $group"
        sudo groupadd "$group"
    else
        echo "[=] Group '$group' already exists."
    fi

    # 2. CREATE USER IF MISSING
    if ! id "$username" >/dev/null 2>&1; then
        echo "Creating user: $username"
        sudo useradd -m -g "$group" "$username"
    else
        echo "User '$username' already exists."
    fi
```



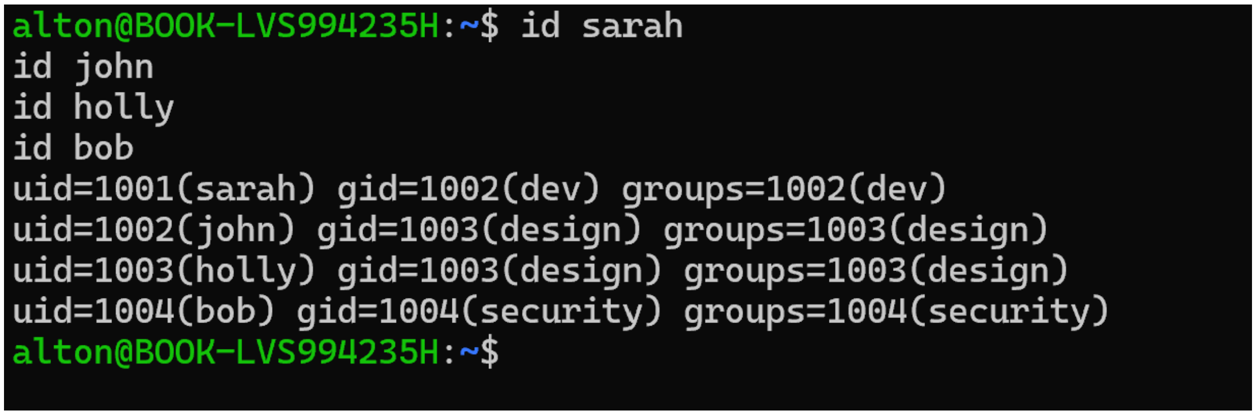
```
    # 3. CREATE WEB DIRECTORY
    WEB_DIR="/home/$username/www"

    if [ ! -d "$WEB_DIR" ]; then
        echo "Creating web directory: $WEB_DIR"
        sudo mkdir -p "$WEB_DIR"
        sudo chown "$username:$group" "$WEB_DIR"
        sudo chmod 755 "$WEB_DIR"
    else
        echo "Web directory already exists."
    fi

    # 4. GENERATE SSH KEYS
    SSH_DIR="/home/$username/.ssh"
    SSH_KEY="$SSH_DIR/id_rsa"

    if [ ! -f "$SSH_KEY" ]; then
        echo "Generating SSH keys for $username"
        sudo -u "$username" mkdir -p "$SSH_DIR"
        sudo -u "$username" ssh-keygen -t rsa -b 2048 -f "$SSH_KEY" -N ""
        sudo chmod 700 "$SSH_DIR"
        sudo chmod 600 "$SSH_KEY"
        sudo chmod 644 "$SSH_KEY.pub"
    else
        echo "SSH keys already exist."
    fi

done < "$INPUT"

echo "----------------------------------------"
echo "SCRIPT COMPLETE"
echo "Users & groups provisioned successfully."
echo
```



```
[+] Creating group: dev
[sudo] password for alton:
User 'alton' already exists.
Web directory already exists.
SSH keys already exist.

----------------------------------------
Processing USER: sarah | GROUP: dev
[=] Group 'dev' already exists.
Creating user: sarah
Creating web directory: /home/sarah/www
Generating SSH keys for sarah
Generating public/private rsa key pair.
Your identification has been saved in /home/sarah/.ssh/id_rsa
Your public key has been saved in /home/sarah/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:5fHmOjBue2PMyXiY3E6NquhzjnQxwdNPBynBeQ+ssSg sarah@BOOK-LVS994235H
+---[RSA 2048]----+
|       ..+..     |
|     . .= =.     |
|      +..*+o.    |
|    E .oo= +.    |
|     .o S o o    |
|       oo  =     |
|    . .o @o.o    |
|   ..+. Bo%.     |
|   .++ooo*oo     |
+----[SHA256]-----+

----------------------------------------
Processing USER: john | GROUP: design
[+] Creating group: design
Creating user: john
Creating web directory: /home/john/www
Generating SSH keys for john
Generating public/private rsa key pair.
Your identification has been saved in /home/john/.ssh/id_rsa
Your public key has been saved in /home/john/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Zaqda90XPDzRlSvs/VkYTL7JzALAtbVkJDR56DHFJcs john@BOOK-LVS994235H
+---[RSA 2048]----+
|       . o=*B.. .|
|        o =Oo+...|
|         o++E+ .o|
```

When the script runs, every user is processed. It checks whether the group exists. If not, it creates the group. Then it checks whether the user exists. If the user does not exist, it creates the account. Then Apache2 directories and SSH keys are generated. Each step prints a confirmation message. This gives feedback to the administrator.

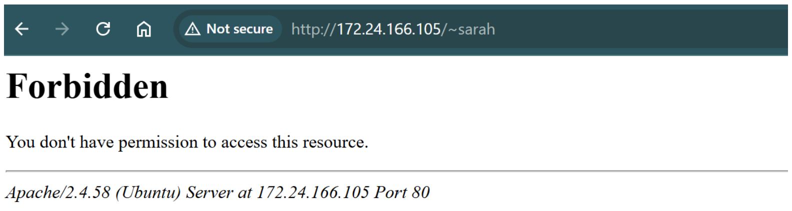
C. Account and Group Verification



```
alton@BOOK-LVS994235H:~$ id sarah
id john
id holly
id bob
uid=1001(sarah) gid=1002(dev) groups=1002(dev)
uid=1002(john) gid=1003(design) groups=1003(design)
uid=1003(holly) gid=1003(design) groups=1003(design)
uid=1004(bob) gid=1004(security) groups=1004(security)
alton@BOOK-LVS994235H:~$
```

Running the command id <user> reveals the Linux user. For example, id john shows the UID along with GID tied to the design group. This confirms the account was properly created. This step also verifies whether multiple users share the same group or have different organizational roles.

D. Web Directory Access



Forbidden

You don't have permission to access this resource.

Apache/2.4.58 (Ubuntu) Server at 172.24.166.105 Port 80

This screenshot shows users attempting to access their /www directory. Since permissions are strict, the server returns error 403 Forbidden. This is good because it proves that unauthorized visitors cannot modify directory files.

V. FUTURE IMPROVEMENTS

Future improvements include automatic HTML page template creation. Instead of an empty directory, each user would receive a starting sample website. This would help new users who do not understand Apache2. Another improvement would be a logging system. The script could write logs into /var/log noting which accounts were created and when. This would help keep track of auditing and mistakes. The last improvement would be the ability to delete accounts automatically when users leave a system, ensuring deprovisioning is safe and clean.

VI. CONCLUSION

This project shows what Bash automation can do in Linux. It automates group creation, user creation, SSH access, and Apache directories without requiring additional software. The result is a practical tool that prevents mistakes, increases security, and saves time. The structure can be adapted to classrooms, small offices, or beginner cybersecurity labs. It gives real world experience in Linux system administration tasks.

VII. REFERENCES

Apache Software Foundation. (n.d.). Apache HTTP Server Documentation. https://httpd.apache.org/docs/

GNU Project. (n.d.). Bash Reference Manual. Free Software Foundation. https://www.gnu.org/software/bash/manual/bash.html

OpenBSD Project. (n.d.). OpenSSH: Secure Shell Protocol. https://www.openssh.com/

Red Hat, Inc. (2023). Using Shell Scripts for System Administration. https://access.redhat.com/documentation/

GitHub. (2024). User Management Automation Script Example Repository. https://github.com/robertdebock/ansible-role-users