CompTIA Linux+: Scripting, Containers, and Automation

Implementing Version Control Using Git

Andrew Mallett
Linux Author and Trainer

@theurbanpenguin www.theurbanpenguin.com

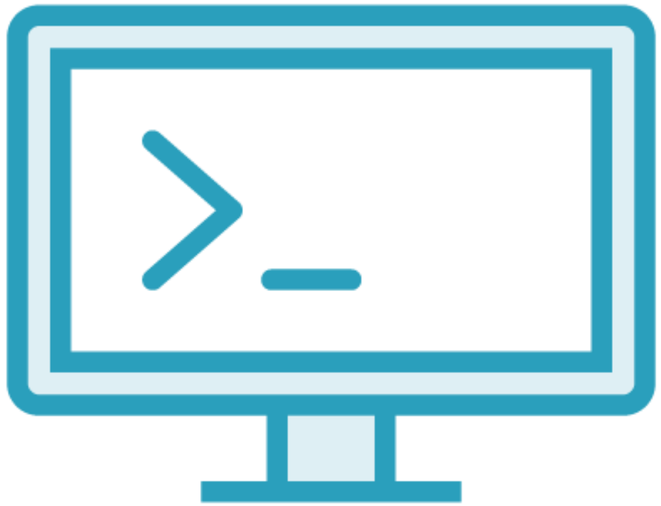
Overview

Course Overview

- Git
- Script automation
- Configuration management and IaC
- Ansible
- Puppet
- SaltStack
- Chef
- Terraform
- Docker
- Kubernetes

CompTIA Linux+ is a distribution agnostic Linux certification helping you to get started in Linux administration and DevOps. It is very much the DevOps tools that we look at in this course. Starting with Git

Distribution Agnostic Lab Systems



CompTIA recommend using a mix of distributions, we include

- Alma Linux 8.5 (RHEL Rebuild)
- Ubuntu 20.04 LTS
- openSUSE Leap 15.2

Additionally, as we are looking at DevOps tooling, we will show you the install of Vagrant and VirtualBox and access the Vagrantfile via Git.

```
$ sudo apt update

$ sudo apt install -y vagrant virtualbox vim git

$ git clone https://github.com/theurbanpenguin/comptia-automation

$ cd comptia-automation/vagrant

$ vagrant up ubuntu
```

Deploying Vagrant

For ease of access to the same virtual machines as used on the labs we recommend using Vagrant and VirtualBox. These are available free of change on Linux, MacOS and Windows. Once installed the Vagrantfile configures the virtual machines and is available from the git repository

Demo

Deploying the lab systems

- We demonstrate on Ubuntu 20.04 how to install Vagrant, VirtualBox and Git
- It is the Ubuntu virtual machine that we use mainly
- If you are using another virtualization system then that too is fine, you may also have Vagrant already installed in which vase you can make use of the supplied Vagrantfile

Working with

GIT

CVS

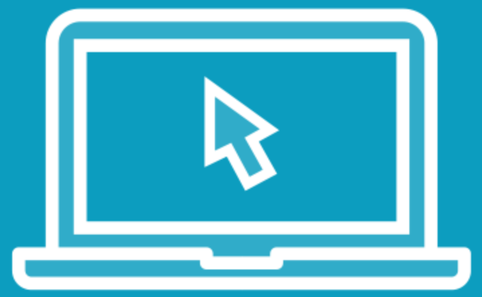
```
$ cd ~/comptia-automation
$ git status
$ git branch
$ cat ~/comptia-automation/shellscripting/hello.sh
$ git branch -a
$ git checkout scriptarg
$ cat ~/comptia-automation/shellscripting/hello.sh
$ git checkout main
```

Working with Branches

We have already used the git download to retrieve the Vagrantfile, allowing us consistency in the VM builds

We will work in creating out own branches, but we have 3 branches in this repo which we can use to see different versions of the same bash script

Demo



Switch Between Different Branches

- View local branches
- View remote branches
- Switch between branches

```
$ git config --list
$ git config --global --list
$ git config --global user.email "tux@example.com"
$ git config --global user.name " Tux Penguin"
$ git config --global --list
$ git config --list
```

Git Configuration

There is a database (.git/) that stores version control information for the git repository. There is also global configuration in the user's home directory and .gitconfig file. To commit changes, even locally we need the user information to be available

Demo

Git Configuration

- Locating git configuration
- Creating global configuration values

Create Local Git Repo

```
$ mkdir ~/myproject ; cd ~/myproject

$ git init ; git status

$ echo "My first line" > myfile ; git status

$ git add .

$ git commit -m "First commit"

$ git show
```

Even Locally: Version Control Can be Useful

Even if you are not going to collaborate on your projects a local git repo will llow ycou to revert changes or maintain different versions of the same file

```
$ echo "Hello world!" > myfile
$ git add . ; git commit -m "Added text"
$ git log --oneline
$ git revert --no-commit -m1 e778753..HEAD
$ git status
$ cat myfile
$ echo "Hello world!" >> myfile
$ git add . ; git commit -m "Initial text corrected"
```

Using Git: Reverting Changes

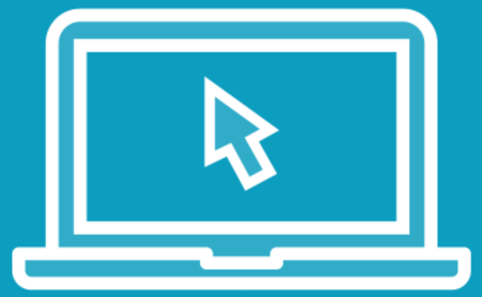
As a version control system, git allows us to revert changes to previously committed (saved) versions. Ideally, we would create a development branch for any new code. But even if we haven't, we can still revert. Choose the correct commit identifier to rollback to your previous changes

Demo

Creating Git Repo

- Create new project directory
- Initialize repo
- Add and commit a new file

Demo



Using a Git Repo

- Make changes and commit them
- Discover there has been a mistake
- Revert changes

```
$ cd ~/myproject
$ mkdir python
$ vim python/my.py
#!/usr/bin/env python3
import datetime
print (datetime.datetime.now())
$ chmod a+x my.py
$ git status ; git add . ; git commit -m "added python script"
$ vim .gitignore
__pycache__
$ python3 -m compileall python/
$ ls /python ; git status
```

Implementing .gitignore

Using git add . is a convenient way to add all modified and new files to the staging area to await committing. Some files should never be added to the git repo such as cache files that are created during execution

Demo

Implementing .gitignore

- Add Python code
- Commit files from subdirectory
- Create .gitignore
- Compile Python code

```
$ mkdir -p ~/git/myproject.git ; cd ~/git/myproject.git
$ git init --bare
$ cd ~/myproject ; git config --list
$ git remote add origin ~/git/myproject.git
$ git remote -v
$ git push --set-upstream origin master
$ git config --list
```

Working with Remote Master

Ideally, we work on a local copy of a repository that is hosted online. We use git push to push changes we have manage to the remote server repo. This allows others to share our code and collaborate. We can demonstrate using another directory structure as the remote repository

Demo

Implement Remote Repository

- Create bare repository
- Configure local repo with remote
- Push changes
- Re-clone to another location

```
$ cd ~/myproject

$ git checkout -b dev

$ vim python/my.py
...

$ git add .

$ git commit -m "Added more date info"

$ git push origin dev
```

Developing New Code Using Branches

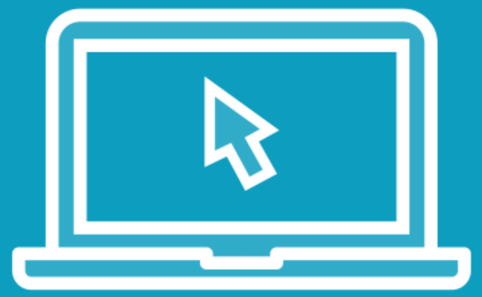
When developing new code, we can implement branches to isolate development code from production code. When development is complete and signed-off we can merge the code with our master or main branch

```
$ rm -rf ~/play/project1
$ git clone -b dev ~/git/myproject.git
$ cd ~/play/project ; git status
$ ./python/my.sh
$ git checkout master
$ git merge dev
$ git push
```

Verify Dev Branch and Merge

Ideally, another developer will check the code and sign it off. They can clone the required branch, test the code; if that is ok, they will then checkout the master branch and merge the dev branch to it

Demo



Developing Code

- Create new branch
- Merge branches

Summary

We have introduced the Linux+ Automation course to you and git:

- Using git as a version control system
- Checkout branches
- Created new local-only repository
  - git init
  - git status
  - git config
  - git add
  - git commit
- Server repository
  - git checkout -b
  - git merge

Implementing Linux Automation Using Shell Scripts