Linux

Linux for Beginners

Basic Commands

cd

- Usage
  - To change directories
- Examples

```
cd temp # browses to the directory called temp
cd /usr/bin # browses to /usr/bin folder
cd - # browse to previous directory
cd .. # browse to the parent direcotry
cd ../../ #browse 2 parent directories behind
```

pwd

- Usage
  - To print current working directory
- Examples

```
pwd # simply lists the current folder you are in
```

ls

- Usage
  - To list all the files and directories
- Examples

```
ls # lists current directory
ls -l # list current directory with details
```

```
ls -a # list all the hidden files
ls -R # list recursively
```

Please note, you can combine a number of the switches in one query rather than running them one by one. e.g. `ls -lah`

mkdir

- Usage
  - to create a new directory
- Examples

```
mkdir directory #create a folder with the name directory
mkidr -p directory/data #create a folder with a subfolder called data
```

cp

- Usage
  - Copy a file from one place to another
- Examples

```
cp hash /temp #copies file into the specified folder
cp hash hash1 #copies the content of a file into a different one
cp -r folder1/ folder2/ #copies the folder with it's content into a folder called
folder2
```

mv

- Usage
  - To move a file from one place to another, also can be used to rename a file.
- Examples

```
mv file1 ../ #moves the file from the current folder to the parent folder
mv file1 file2 #renames file1 to file2
```

touch

- Usage
  - To create an empty file.
- Examples

```
touch file1 #create file 1
touch hash && echo "some_text" > hash # create a file cat hash and write in the file the words "some_text"
```

cat/tac

- Usage
  - To read the content of a file.
- Examples

```
cat hash # read the content of the file called hash
tac hash # same as above
```

rm

- Usage
  To remove a file or folder.
- Examples

```
rm file1 # remove file1
rm -rf folder1 # recursively remove the content of folder1 and discard any files in use.
```

grep

- Usage
  Search for a line or specific text in a file.

- Examples

```
grep "text" file_with_text.txt # Print all lines that have the word "text"
grep -c "text" file_with_text.txt # Count the number of occurences of the word "text"
grep -i "text" file_with_text #Print all lines that contain the case insensitive word "text"
```

> The grep function has a ton of use cases, it can do regex matches, show lines before or after specific text and a lot more. I would advise to read the manual pages for more use cases. See the below URL for more information on the command: GNU Grep 3.7

head

- Usage
  - Read the first n lines of a file.
- Examples

```
head file.txt # Default behaviour is to output the first 10 lines of a file
head -n 3 # Output the first 3 lines of a file
```

tail

- Usage
  - Read the last n lines of a file
- Examples

```
tail file.txt # Default behaviour, output the last 10 lines of a file
tail -n 3 # Output the last 3 lines of a file
```

chmod

- Usage
  - To change the permissions of a file or folder

- Examples

```
chmod +x file1 # make a specific file executable
chmod 754 file1 # grant the file read, write, execute to user, read and execute to group and read to other
```

echo

- Usage
  - To write something in the console or file.
- Examples

```
echo "Some really interesting text" #write the test between brackets
```

clear

- Usage
  - To clear the terminal windows of text.
- Examples

```
clear # clears the terminal window
```

sudo

- Usage
  - To escalate your privileges to super user.
- Examples

```
sudo ./script.sh # execute the script as root
sudo apt install memes # escalate your privileges to install the application called memes
```

Komodo

> https://komo.do - For in-depth information utilise the following

Introduction

Komodo is a lightweight but powerful build and deployment system that cuts through the usual headaches of managing Docker-based projects. At its core, it automates what so many teams end up duct-taping together: auto-versioned Docker image builds directly from your git repos, triggered on push. No fiddling with clunky CI pipelines just to get an image out the door.

Deployment is just as smooth. You can push containers or entire docker-compose setups, check uptime, and view logs across all your servers from one place. Instead of juggling multiple dashboards or SSH sessions, Komodo centralizes the experience.

What I love is that it’s written in Rust, which means the API and agent are ridiculously fast and memory-safe, exactly the kind of foundation you want for something that touches every part of your deployment stack. It feels clean, sturdy, and built to last.

In short: Komodo is a tool that makes deployment less of a chore and more of a background hum - always running, always reliable. It takes care of the boring stuff so you can focus on building.

Basic installation

> DO NOT USE THIS FOR PRODUCTION. Make sure to update the required fields before rolling out in your estate.

1. To set up and get you going it’s as simple as:

```
wget -P komodo
<https://raw.githubusercontent.com/moghtech/komodo/main/compose/mongo.compose.yaml> && \
wget -P komodo <https://raw.githubusercontent.com/moghtech/komodo/main/compose/compose.env>
```

1. Followed by:

```
docker compose -p komodo -f komodo/mongo.compose.yaml --env-file komodo/compose.env up -d
```

Updating

```
docker compose -p komodo -f komodo/mongo.compose.yaml --env-file komodo/compose.env pull
docker compose -p komodo -f komodo/mongo.compose.yaml --env-file komodo/compose.env down
docker compose -p komodo -f komodo/mongo.compose.yaml --env-file komodo/compose.env up -d
```

Usage

I utilise Komodo to manage all my docker containers that I utilise for various things, such as personal life, website, this knowledge base, security projects or quickly testing new applications to see if I like them or not.