Is DevOps a good career?

Manjunath.R
#16/1, 8th Main Road, Shivanagar, Rajajinagar, Bangalore560010, Karnataka, India
*Corresponding Author Email: manjunath5496@gmail.com
*Website: http://www.myw3schools.com/

Abstract

DevOps (a set of software development practices that combines software development (Dev) and information technology operations (Ops) to shorten the systems development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives) is becoming the standard way of working for Enterprises. Among the few powerful trends we had experienced in the recent times, one is undoubtedly the adoption of DevOps practices – and adoption of DevOps within the organization is rising on a broader scale, and Enterprises are trending toward it. DevOps builds upon best practices to help drive enterprise performance in modernizing environments. It offers organizations a new way to move the business forward and turn technology into a strategic advantage. An increasing number of businesses recognize the power that DevOps can bring a natural extension for Agile and continuous delivery approaches.
Patrick Debois is best known as the founder of DevOpsDays and as a creator of the DevOps movement, which explains why some refer to him as the "Godfather of DevOps".

"At its essence, DevOps is a culture, a practice, a philosophy."

Introduction

DevOps expertise is in high demand. Job postings with "DevOps" in a title or keyword are sprouting up everywhere. DevOps is an enterprise software development phrase emerging from combination of IT teams, process and products to enable the continuous delivery of value to end users. It is a firm bond between development and operations that emphasizes a shift in mindset, better collaboration, and tighter integration and aims to create a culture and environment where building, testing, and releasing software can happen rapidly, often, and more reliably, so organizations can solve critical issues quickly, and better serve their customers and compete more effectively in the market.
What is DevOps?

"A software development method formed out of a fundamental need that stresses communication, collaboration and integration between software developers and IT professionals." DevOps could be explained simply as operations working together with engineers to get things done faster in an automated and repeatable way.

History of DevOps

At the 2008 Agile Toronto conference, Andrew Shafer and Patrick Debois introduced the term in their talk on "Agile Infrastructure". Since 2009, the DevOps term has been steadily promoted based on a simple philosophy — business works best when efforts being coordinated and collaborative — and brought into more mainstream usage through a series of "DevOpsDays", which started in Belgium and has now spread into Web-enabled sphere to resolve the conflict between the software developers and the operations teams when it comes to getting great work done quickly. In recent years, more tangential DevOps initiatives have also evolved, such as OpsDev, WinOps, and BizDevOps to encourage the communication between software developers and IT Operations to increase the speed at which applications being delivered.

Benefits of DevOps
The technical benefits include:

- Continuous software delivery
- Less complexity to manage
- Faster resolution of problems

The cultural benefits include:

- More productive teams
- Higher employee engagement
- Greater professional development opportunities

The business benefits include:

- Faster delivery of features
- More stable operating environments
- Improved communication and collaboration
- More time to innovate (and not fix / keep up)

Features of DevOps

- **Source control**: Software developers need to safely store their code and keep track of source-code history and versions. For this reason alone, source control is of critical importance.
- **Issue tracking system**: An issue tracking system allows everyone involved to track current issues, estimates, and deadlines.
- **Build system**: The build system supports continuous integration by building the software, running unit and integration tests, deploying to the integration environment, and performing any other automated checks defined for new versions of the software.
- **Monitoring system**: Monitoring systems continuously track all autonomous systems within the DevOps environment, notifying necessary maintenance staff if a system failure occurs.
- **Communications system**: The constant exchange of information is important so email, wiki, and a real-time chat system being enabled for effective communication and collaboration among all members of the project team.
• **Integration environment:** The integration environment hosts all the virtual machines that make up our DevOps environment

• **Code review system:** To make sure software quality, every line of code being reviewed by a experienced developer. The practice of reviewing code also accelerates career growth and learning.

• **Documentation system:** Regrettably, documentation often remains an afterthought in production software projects. To ensure that documentation being written throughout the project, an automated system being developed to allow developers to write documentation easily, along with source code.

![DevOps Cycle Diagram](image)

**DevOps Goals**

- Improved deployment frequency
- To make faster time to market
- Less failure rate to new releases
- Short lead time between fixes
- Improve mean time to recovery

**Is DevOps a good career?**

DevOps practitioners are among the highest paid IT professionals today, and the market demand for them is growing rapidly because organizations using DevOps practices are overwhelmingly high-functioning to
deliver IT services that offer value to the business. According to a study on the application economy and the role of DevOps, 88% of enterprise IT organizations and \textbf{LOB (line of business)} executives already have planned to adopt DevOps sometime within the next five years to accelerate delivery of apps and offer customers with higher-quality software. In the last two years, listings for DevOps jobs at \textbf{Indeed.com} increased 75 percent. On \textbf{LinkedIn.com}, mentions of DevOps as a skill increased 50 percent. In a recent survey by \textbf{Puppetlabs}, half of their 4,000-plus respondents (in more than 90 countries) said their companies consider DevOps skills when hiring.

\textbf{Basic MongoDB Commands:}

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>db.help()</code></td>
<td>get a list of commands</td>
</tr>
<tr>
<td><code>show dbs</code></td>
<td>print a list of all databases on the server</td>
</tr>
<tr>
<td><code>use myTestDB</code></td>
<td>create new database &quot;myTestDB&quot;</td>
</tr>
<tr>
<td><code>db</code></td>
<td>know your current selected database</td>
</tr>
<tr>
<td><code>db.dropDatabase()</code></td>
<td>drop the current selected database</td>
</tr>
<tr>
<td><code>db.createCollection(&quot;Employee&quot;)</code></td>
<td>create new collection &quot;Employee&quot;</td>
</tr>
<tr>
<td><code>show collections</code></td>
<td>print a list of all collections created</td>
</tr>
<tr>
<td><code>db.Employee.drop()</code></td>
<td>drop the collection &quot;Employee&quot;</td>
</tr>
<tr>
<td><code>db.Employee.insert({name: 'Raj', address: 'Bangalore'})</code></td>
<td>insert document in collection &quot;Employee&quot;</td>
</tr>
<tr>
<td><code>db.Employee.find()</code></td>
<td>list the documents in collection &quot;Employee&quot;</td>
</tr>
<tr>
<td><code>{ &quot;_id&quot; : ObjectId(&quot;60658a0dbe02cfa1d386ab52&quot;), &quot;name&quot; : &quot;Raj&quot;, &quot;address&quot; : &quot;Bangalore&quot; }</code></td>
<td></td>
</tr>
<tr>
<td><code>db.Employee.update({'name' : 'Raj'}, {$set: {'name' : 'Albert'}})</code></td>
<td>update the document in collection &quot;Employee&quot;</td>
</tr>
<tr>
<td><code>db.Employee.find()</code></td>
<td>list the documents in collection &quot;Employee&quot;</td>
</tr>
<tr>
<td>Document</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| `{ "_id" : ObjectId("60658a0dbe02cfa1d386ab52"), "name" : "Albert", "address" : "Bangalore" }` | save document in collection "Employee"
| `db.Employee.save({"_id": new ObjectId("60658a0dbe02cfa1d386ab53"), name: "Newton", address: "Delhi"});` | save document in collection "Employee"
| `db.Employee.find()` | list the documents in collection "Employee"
| `{ "_id" : ObjectId("60658a0dbe02cfa1d386ab52"), "name" : "Albert", "address" : "Bangalore" }`
| `{ "_id" : ObjectId("60658a0dbe02cfa1d386ab53"), "name" : "Newton", "address" : "Delhi" }` | list the documents in collection "Employee"
| `delete document in collection "Employee"` | delete document in collection "Employee"
| `db.Employee.remove({'name': 'Albert'})` | delete document in collection "Employee"
| `db.Employee.find()` | list the documents in collection "Employee"
| `{ "_id" : ObjectId("60658a0dbe02cfa1d386ab53"), "name" : "Newton", "address" : "Delhi" }` | list the documents in collection "Employee"
| `db.getUsers();` | list down all the users of current database
| `show roles` | list down all the roles
| `db.Employee.dataSize()` | get the size of the collection "Employee"
| `db.Employee.storageSize()` | get the total size of document stored in the collection "Employee"
| `db.Employee.totalSize()` | get the total size in bytes for both collection data and indexes
| `db.Employee.totalIndexSize()` | get the total size of all indexes in the collection "Employee"

**Docker Commands:**

![Docker Logo](docker.png)
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>docker --version</code></td>
<td>get the installed docker version</td>
</tr>
<tr>
<td><code>docker pull hello-world</code></td>
<td>download the image &quot;hello-world&quot; from the docker repository (hub.docker.com)</td>
</tr>
<tr>
<td><code>docker images</code></td>
<td>list all the images that are locally stored with the docker engine</td>
</tr>
<tr>
<td><code>docker run hello-world</code></td>
<td>create a container from the image &quot;hello-world&quot;</td>
</tr>
<tr>
<td><code>docker container ls -a</code></td>
<td>list all containers</td>
</tr>
<tr>
<td><code>docker container ls -a -s</code></td>
<td>list the size for all containers</td>
</tr>
<tr>
<td><code>docker rmi 515d5e66f68a</code></td>
<td>remove the docker image &quot;hello-seattle&quot; with image id &quot;515d5e66f68a&quot;</td>
</tr>
<tr>
<td><code>docker rm d9bf06498bb2</code></td>
<td>remove the docker container with container id &quot;d9bf06498bb2&quot;</td>
</tr>
<tr>
<td><code>docker history hello-world</code></td>
<td>display the history of the image &quot;hello-world&quot;</td>
</tr>
<tr>
<td><code>docker info</code></td>
<td>get detailed information about docker installed on the system including the kernel version, number of containers and images, etc.</td>
</tr>
<tr>
<td><code>docker volume create</code></td>
<td>create a volume which docker container will use to store data</td>
</tr>
<tr>
<td><code>docker volume ls</code></td>
<td>list all the volumes known to Docker</td>
</tr>
<tr>
<td><code>docker logs c70201336fd8</code></td>
<td>display the logs of the docker container with contained id &quot;c70201336fd8&quot;</td>
</tr>
<tr>
<td><code>docker search hadoop</code></td>
<td>search for docker image &quot;hadoop&quot; on dockerhub</td>
</tr>
<tr>
<td><code>docker network ls</code></td>
<td>list all docker networks</td>
</tr>
<tr>
<td><code>docker login</code></td>
<td>login into docker repository (hub.docker.com)</td>
</tr>
<tr>
<td><code>docker logout</code></td>
<td>logout from docker repository (hub.docker.com)</td>
</tr>
<tr>
<td><code>docker start c70201336fd8</code></td>
<td>start the docker container with container id &quot;c70201336fd8&quot;</td>
</tr>
<tr>
<td><code>docker stop c70201336fd8</code></td>
<td>stop the docker container with container id &quot;c70201336fd8&quot;</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>docker restart c70201336fd8</code></td>
<td>restart the docker container with container id &quot;c70201336fd8&quot;</td>
</tr>
<tr>
<td><code>docker inspect c70201336fd8</code></td>
<td>get detailed information about the docker container with container id &quot;c70201336fd8&quot;</td>
</tr>
<tr>
<td><code>docker stats c70201336fd8</code></td>
<td>get the statistics of the docker container with container id &quot;c70201336fd8&quot;</td>
</tr>
<tr>
<td><code>docker image ls</code></td>
<td>List all images that are locally stored with the docker engine.</td>
</tr>
</tbody>
</table>

"While Docker automatically captures logs for you, it does not also rotate them. In fact, currently none of the provided packages set up any log rotation. You’ll need to do that yourself in most cases. Rather frustratingly, Docker also does not respond to a signal to tell it to reopen logs. If you send it the standard HUP signal, it will instead restart all the containers, which is not what you want. The current best practice for rotation of Docker logs is to have logrotate use the copytruncate method to copy the logfile and then truncate it in place. There are open bugs against docker asking for a better solution."

— Karl Matthias

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>docker system prune</code></td>
<td>delete all unused containers, unused networks, and dangling images</td>
</tr>
<tr>
<td><code>systemctl status docker</code></td>
<td>check the Docker service</td>
</tr>
<tr>
<td><code>systemctl start docker</code></td>
<td>start the Docker service</td>
</tr>
<tr>
<td><code>docker image prune</code></td>
<td>remove unused images</td>
</tr>
<tr>
<td><code>docker save hello-world &gt; hello-world.tar</code></td>
<td>save the image &quot;hello-world&quot; to a tar archive</td>
</tr>
<tr>
<td><code>docker load &lt; hello-world.tar</code></td>
<td>load the image &quot;hello-world&quot; from the saved tar file</td>
</tr>
<tr>
<td><code>docker export a27999b71e62 &gt; hello-world.tar</code></td>
<td>export the docker container with container id &quot;a27999b71e62&quot; as a tar archive</td>
</tr>
<tr>
<td><code>docker import hello-world.tar</code></td>
<td>import the contents from hello-world.tar</td>
</tr>
</tbody>
</table>
The **command-line interface** is one of the nearly all well built trademarks of **Linux**. There exists an ocean of **Linux commands**, permitting you to do nearly everything you can be under the impression of doing on your Linux operating system. Although, this to the end of time creates a problem: by all of so copious commands accessible to manage, you don't comprehend where and at which point to fly learning them, especially when you are learner. If you are facing this problem, and are peering for a painless method to begin your command line journey in Linux, you've come to the right place, as in this, we will launch you to a hold of well liked and **helpful Linux commands**.

---

**Description:**

**Display system date and time.**

---

**Command:**

date

---

**Description:**

**Display calendar.**
Command: cal

Description:
Display date, time and calendar.

Command: date & cal

Description:
Display August month 2016 year calendar.

Command: cal 8 2016

Description:
Used to clear the terminal window.

Command:
clear

Description:
Exit from the terminal window.

Command:
exit

Description:
Display free and used system memory.
Description:

Display free and used system memory in bytes.

Command:

```
free -b
```

Description:

Display free and used system memory in kilobytes.

Command:

```
free -k
```

Description:

Display free and used system memory in megabytes.

Command:

```
free -m
```
Description:
Change user password.

Command:
```
passwd
```

Description:
Power-off the machine.

Command:
```
shutdown
```

Description:
Power-off the machine immediately.

Command:
```bash
shutdown -h now
```

**Description:**

*Power-off the machine after 10 minutes.*

```bash
shutdown -h +10
```

**Description:**

*Print current working directory.*

```bash
echo $PWD
```

**Description:**

*Print previous working directory.*
Command:

`echo $OLDPWD`

Description:

Executes the 11th command in command history.

Command:

`!11`

Description:

Reveals your command history.

Command:

`history`

Description:

Power off or reboot the Operating system.
Command:

sudo reboot

Description:

Display the IP address of the host.

Command:

ip address

Description:

List the size of files and directories.

Command:

ls -s

Description:
View mounted file systems.

Command:

mount

Description:

Display the information of disk usage of files and directories.

Command:

du

Description:

Tells you how long the system has been running.

Command:

uptime
Description:

Set current date as 02 Nov 1988.

Command:

date -- set 1998-11-02

Description:

Set current time as 12:11:02 IST.

Command:

date -- set 12:11:02

Description:

View and change the configuration of the network interfaces on the system.

Command:

ifconfig
Description:

Lists all files and directories in the present working directory.

Command:

ls

Description:

Report the process information.

Command:

ps

Description:

Display disk usage.
**Description:**

Display disk usage in gigabytes, megabytes, or kilobytes.

**Command:**

```
df -H
```

---

**Description:**

Delete every file and every directory.

**Command:**

```
rm -r *
```

---

**Description:**

Provides a quick overview of the currently running processes.
Command: top

Description: The system performs an immediate reboot.

Command: reboot

Description: Terminate processes without having to log out or reboot.

Command: kill

Description: Change the current working directory.
Command:
cd

Description:
Create a new session on the system.

Command:
login

Description:
List open files.

Command:
lsof

Description:
List USB devices.

Command:

lsusb

Description:

Check the status of the network services.

Command:

service network status

Description:

Start the network service.
Description:

Stop the network service.

Command:

```bash
service network stop
```

Description:

Restart the network service.

Command:

```bash
service network restart
```

Description:

Report information about the users currently on the machine and their processes.

Command:

```bash
w
```
Description:

Display the current directory.

Command:

pwd

Description:

Displays CPU architecture information (such as number of CPUs, threads, cores, sockets, and more).

Command:

lscpu

Description:

Displays the number of processing units available to the current process.
Command:
nproc

Description:
The system performs an immediate reboot.

Command:
init 6

Description:
Power-off the machine.

Command:
init 0

Description:
List files by date.
Command:

ls -lrt

Description:

Report information about storage devices such as hard disks, flash drives etc.

Command:

lsblk

Description:

Show exit status of previous command.

Command:

echo $?
Lists a few useful info commands.

**Command:**

info

**Description:**

Prints current year’s calendar.

**Command:**

cal -y

**Description:**

Check the status of all the services.

**Command:**

service --status-all
Description:

Display time in hh:mm:ss.

Command:

date +%T

Description:

Tells when the user last logged on and off and from where.

Command:

last -1 username

Description:

Sort files and directories by extension name.

Command:

ls -X
Description:

Display the manual for the `pwd` command.

Command:

```
man pwd
```

Description:

Displays information about running processes in the form of a tree.

Command:

```
pstree
```

Description:

Resets your terminal.

Command:
Description:

Displays What date is it this Friday.

Command:

date -d fri

Description:

Displays the size of each individual file.

Command:

du -a

Description:

Display information about the Advanced configuration and power Interface.
Command:

acpi

Description:

Takes you two folders back.

Command:

cd ../..

Description:

Takes you to the previous directory.

Command:

cd -

Description:

Displays a list of shell built-in commands.
Command:

help

Description:

Lists your last logins.

Command:

last yourusername

Description:

Create a new directory called myfiles.

Command:

mkdir myfiles

Description:
Remove the directory myfiles.

Command:

rmdir myfiles

Description:

Disable password for a specific user "root1".

Command:

passwd -d root1

Description:

Switch to user "root1".

Command:

sudo su root1
Description:

Exit from the terminal window.

Command:

logout

Description:

Creates a user "root1".

Command:

useradd "root1"

Description:

Assign password to user "root1".

Command:

passwd "root1"
Description:

Repeats the last command.

Command:

!!

Description:

Display Who you are logged in as.

Command:

whoami

Description:

Display the login name of the current user.
logname

Description:

Report the name of the kernel.

Command:

uname

Description:

Print the kernel version.

Command:

uname -v

Description:

Print the operating system.
Command:
uname -o

Description:
Report the machine hardware name.

Command:
uname -m

Description:
Print version information and exit.

Command:
uname --version

Description:
Print the kernel release.
Command:

```
uname  -r
```

Description:

Report the network node hostname.

Command:

```
uname  -n
```

Description:

Display all port connections (both TCP and UDP).

Command:

```
netstat  -a
```

Description:
Display only TCP (Transmission Control Protocol) port connections.

Command:

```
netstat -at
```

Description:

Display only UDP (User Datagram Protocol) port connections.

Command:

```
netstat -au
```

Description:

Display all active listening ports.

Command:

```
netstat -I
```
Description:

Display all active listening TCP ports.

Command:

netstat  -It

Description:

Display all active listening UDP ports.

Command:

netstat  -lu

Description:

Reveal all the information about the current user (user id, username, group id, group name etc.).

Command:
Description:

Reveal all the information about the user "root1" (user id, username, group id, group name etc.).

Command:

id root1

Description:

Print the machine's architecture.

Command:

arch

Description:

Display the list of available fonts.
Command:

fc-list

Description:

Create two directories (myfiles, files).

Command:

mkdir myfiles files

Description:

Install apache (CentOS).

Command:

yum install httpd

Description:
install apache (Ubuntu).

Command:

apt install httpd

Description:

upgrade apache (CentOS).

Command:

yum update httpd

Description:

upgrade apache (Ubuntu).

Command:

apt update httpd
**Description:**

uninstall apache (CentOS).

**Command:**

```
yum remove httpd
```

**Description:**

uninstall apache (Ubuntu).

**Command:**

```
apt remove httpd
```

**Description:**

Display usage summary for the command (date).

**Command:**

```
date --help
```
Description:

List active connections to/from system.

Command:

ss -tup

Description:

List internet services on a system.

Command:

ss -tup1

Description:

Display all active UNIX listening ports.

Command:
netstat -lx

Description:
Display all the active interfaces details.

Command:
ifconfig

Description:
Display information of all network interfaces.

Command:
ifconfig -a

Description:
Compare the contents of two files (1.txt, 2.txt).
Command:

diff 1.txt 2.txt

Description:

Tells you how many lines, words, and characters there are in a file (1.txt).

Command:

wc 1.txt

Description:

Compresses file (1.txt), so that it take up much less space.

Command:

gzip 1.txt

Description:

Uncompresses file (1.txt) compressed by gzip.
Command:

gunzip 1.txt

Description:

Examine the contents of the file (1.txt).

Command:

cat 1.txt

Description:

Display calendar.

Command:

ncal

Description:
Removes the file (1.txt).

Command:

```
rm 1.txt
```

Description:

Rename a file named 1.txt to 0.txt.

Command:

```
mv 1.txt 0.txt
```

Description:

Replace the contents of 0.txt with that of 1.txt.

Command:

```
cp 1.txt 0.txt
```
Description:
Create a empty file (test.txt).

Command:
touch test.txt

Description:
Print the last 10 lines of a file (1.txt).

Command:
tail 1.txt

Description:
Print N number of lines from the file (1.txt).

Command:
tail -n N 1.txt
Description:

Prints the number of words in a file (1.txt).

Command:

```
wc -w 1.txt
```
Description:

Print information about USB ports, graphics cards, network adapters etc.

Command:

lspci

Description:

View contents of a file (1.txt).

Command:

less 1.txt

Description:

Display calendar (last month, current month, and next month).
Command:

cal -3

Description:

Compare the contents of three files (1.txt, 2.txt, 3.txt) line by line.

Command:

diff3 1.txt 2.txt 3.txt

Description:

Compare two files (1.txt, 2.txt) line-by-line.

Command:

comm 1.txt 2.txt

Description:

Perform byte-by-byte comparison of two files (1.txt, 2.txt).
Command:

cmp 1.txt 2.txt

Description:

Prints the CRC checksum and byte count for the file "myfiles.txt".

Command:

csum myfiles.txt

Description:

Append contents of files (1.txt, 2.txt) into one file (0.txt).

Command:

cat 1.txt 2.txt > 0.txt

Description:
Append contents of files (1.txt, 2.txt, 3.txt) into one file (0.txt).

Command:

```
sed  r 1.txt 2.txt 3.txt > 0.txt
```

Description:

Append contents of files (1.txt, 2.txt, 3.txt) into one file (0.txt).

Command:

```
sed  h 1.txt 2.txt 3.txt > 0.txt
```

Description:

Append contents of files (1.txt, 2.txt, 3.txt) into one file (0.txt).

Command:

```
sed  -n  p 1.txt 2.txt 3.txt > 0.txt
```
**Shortcuts:**

<table>
<thead>
<tr>
<th>ctrl+c</th>
<th>Halts the current command</th>
</tr>
</thead>
<tbody>
<tr>
<td>ctrl+z</td>
<td>Stops the current command</td>
</tr>
<tr>
<td>ctrl+d</td>
<td>Logout the current session</td>
</tr>
<tr>
<td>ctrl+w</td>
<td>Erases one word in the current line</td>
</tr>
<tr>
<td>ctrl+u</td>
<td>Erases the whole line</td>
</tr>
<tr>
<td>ctrl+r</td>
<td>Type to bring up a recent command</td>
</tr>
</tbody>
</table>

**Description:**

*Writes contents of a file (0.txt) to output, and prepends each line with line number.*

**Command:**

`nl 0.txt`

**Description:**

*Create a empty file (test1.txt) inside a directory (test).*
Command:

```bash
mkdir test
cd test
pwd
touch test1.txt
```

Description:

Gather information about hardware components such as CPU, disks, memory, USB controllers etc.

Command:

```bash
sudo lshw
```

Description:

Gather information about file system partitions.

Command:

```bash
sudo fdisk -l
```
Description:

Displays the line (good morning) in which the string (good) is found in the file (1.txt).

Command:

grep good 1.txt

Description:

Append contents of files (1.txt, 2.txt, 3.txt) into one file (0.txt) using for loop.

Command:

for i in {1..3}; do cat "$i.txt" >> 0.txt; done

Description:

Search for files (test.txt, test1.txt, test2.txt, test.php, test.html) in a directory as well as its sub-directories.
Command:

find test*

Description:

Displays status related to a file (1.txt).

Command:

stat 1.txt

###

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>Open vi editor</td>
</tr>
<tr>
<td>i</td>
<td>Go to Insert mode</td>
</tr>
<tr>
<td>a =20; b =64;</td>
<td></td>
</tr>
<tr>
<td>print (a + b);</td>
<td></td>
</tr>
<tr>
<td>Hit Escape to return to Normal mode.</td>
<td></td>
</tr>
<tr>
<td>:w hello.py</td>
<td>Save text</td>
</tr>
<tr>
<td>:q</td>
<td>Quit</td>
</tr>
<tr>
<td>python hello.py</td>
<td>Print the output:84</td>
</tr>
</tbody>
</table>
Download the file (file.txt) from url "http://website.com/files/file.txt".

Command:

`wget http://website.com/files/file.txt`

Display host's numeric ID in hexadecimal format.

Command:

`hostid`

Display file type of the file (myfiles.txt).

Command:

`file myfiles.txt`
Description:
Create a file (myfiles.txt) containing a text (Hello World).

Command:
```bash
echo 'Hello World' > myfiles.txt
```

Description:
Create a file (myfiles.txt) containing a text (Hello World).

Command:
```bash
printf 'Hello World' > myfiles.txt
```

Description:
Display IP address of the hostname.

Command:
hostname -i

---

**Description:**

Add a new line of text to an existing file (1.txt).

**Command:**

```bash
echo "Hello world!" >> 1.txt
echo "this is 2nd line text" >> 1.txt
echo "last line!" >> 1.txt
```

---

**Description:**

Displays a single line description about a command (cal).

**Command:**

```bash
whatis cal
```

---

```text
###
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>---------------</td>
<td>-------------------</td>
</tr>
</tbody>
</table>
```

64
| vi                 | Open vi editor     |
| i                  | Go to Insert mode  |
| Type some text.    |                    |
| Hit Escape to return to Normal mode. |
| :w test.txt       | Save text          |
| :q                 | Quit               |
| :q!               | Quit without saving |

###

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| :-----------------|-------------:
| vi               | Open vi editor        |
| i                | Go to Insert mode     |
| $name = "Paul"; |                       |
| print "$name";  |                       |
| Hit Escape to return to Normal mode. |
| :w hello.pl      | Save text             |
| :q               | Quit                  |
| perl hello.pl    |Print the output: Paul |

###

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| :-------------------------|-------------:
| vi                        | Open vi editor         |
| i                         | Go to Insert mode     |
| echo "What is your name?" |                       |
| read PERSON               |                       |
| echo "Hello, $PERSON"     |                       |
| Hit Escape to return to Normal mode. |
| :w hello.sh               | Save text             |
| :q                         | Quit                  |
| sh hello.sh               | Output:              |
|                            | What is your name?    |
|                            | If you enter: Zara Ali |
|                            | Hello, Zara Ali       |
Description:

Check the network connectivity between host (your connection) and server (Google server).

Command:

```
ping google.com
```

Description:

Find the location of source/binary file of a command (cal).

Command:

```
whereis cal
```

Description:

List the files in the bin directory.
Command:

ls /bin

Description:

List the files in the bin directory and the etc directory.

Command:

ls /bin /etc

Description:

Moves the file test.txt to the folder newrepo.

Command:

mv test.txt ./newrepo

Description:

Deletes all the lines in the test.txt containing tue word.
Command:

```bash
sed -i "/tue/d" test.txt
```

```
import subprocess
subprocess.call('linux command')
```

```
import os
os.system('linux command')
```

```
import os
os.system('ls')
```

List all the files and directories in the current directory
DevOps isn't any single person's job. It's everyone's job.

Christophe Capel

Ideate, define, and describe features and capabilities of the application

Plan

Workflow

Collaboration

Application

Security and Compliance

Continuous improvement

Operate

Develop

Deliver

Maintain, monitor and troubleshoot application in production environment

Deploy application into production environment in a consistent and reliable way

Write, test, review and integrate code
Development (Software engineering) + Quality assurance + operations = DevOps

Development (Software engineering) + Quality assurance + operations + Security

DevSecOps

Dev

(Software releases and updates)

Ops

(Reliability, performance and scaling)

Sec

(Confidentiality, availability and integrity)

Agile

Focuses on processes
highlighting change while
accelerating delivery

CI/CD

Focuses on software-defined
life cycles highlighting tools
that emphasize automation

DevOps

Focuses on culture
highlighting roles that
emphasize responsiveness
<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Git</td>
<td>Version Control System tool</td>
</tr>
<tr>
<td>Jenkins</td>
<td>Continuous Integration tool</td>
</tr>
<tr>
<td>Selenium</td>
<td>Continuous Testing tool</td>
</tr>
<tr>
<td>Puppet, Chef, Ansible</td>
<td>Configuration Management and Deployment tools</td>
</tr>
<tr>
<td>Nagios</td>
<td>Continuous Monitoring tool</td>
</tr>
<tr>
<td>Docker</td>
<td>Containerization tool</td>
</tr>
</tbody>
</table>
Cloud Service:

On-premise

You manage

Applications
Data
Runtime
Middleware
OS
Virtualization
Servers
Storage
Networking

You manage

Platform as a service

Others manage

Application
Data
Runtime
Middleware
OS
Virtualization
Servers
Storage
Networking

You manage

Software as a service

Others manage

Application
Data
Runtime
Middleware
OS
Virtualization
Servers
Storage
Networking
Stages of a typical DevOps delivery pipeline

Understand your users and frame a hypothesis

Craft an experiment

Test your hypothesis with users

Learn, adopt and correct
# Basic Hello World program written in Groovy

class MyClass {
    static void main(String[] args) {

        println('Hello World');
    }
}

## 6 Build Phases in Maven:

Validate → Compile → Test → Package → Install → Deploy

### Command:

ifstat

### Description:

Prints network interface statistics
Processes

Parent processes

Processes that create other processes during runtime

Child processes

Processes that are created by other processes during run-time

Command:

```
pidof bash
``` 

Description:

Display the process IDs of a specific running program (Bash)

```
# Display the process ID of the current shell

echo $$

# Display the parent process ID of the current shell

echo $PPID
```
**Command:**

```
ps aux | awk '{print $6/1024 " MBtt" $11}' | sort -n
```

**Description:**

Display a list of most memory consuming processes

<table>
<thead>
<tr>
<th>Github Fetch</th>
<th>Github Pull</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetches the required information only to local repository</td>
<td>Fetches the required information not only to local repository but also to the workspace that you are currently working in</td>
</tr>
</tbody>
</table>

Github Pull = Github fetch + Merge

Combination of fetch and merge the content

Fetch the content

<table>
<thead>
<tr>
<th>Docker registry</th>
<th>Docker repository</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service for hosting and distributing docker images</td>
<td>Collection of related Docker images</td>
</tr>
</tbody>
</table>
Selenium supports 2 types of testing:

- **Regression Testing** → retesting a product around an area where a bug was fixed.
- **Functional Testing** → testing of software features (functional points) individually.

**Command:**

```
ps aux | awk '{print $6/1024 " MBtt" $11}' | sort -n
```

**Description:**

Display a list of most memory consuming processes

**Command:**

```
ps aux
```

**Description:**

Display all processes and their status and resource usage
**Command:**

last reboot

**Description:**

Show system reboot history

**Command:**

dmesg

**Description:**

Displays the messages from the kernel ring buffer (a data structure that records messages related to the operation of the kernel)

**Command:**

`cat /proc/cpuinfo`

**Description:**

Display CPU information
Command:

```
cat /proc/meminfo
```

**Description:**

Display memory information

Command:

```
lspci -tv
```

**Description:**

Display PCI (Peripheral Component Interconnect) devices

Command:

```
lsusb -tv
```

**Description:**

Display USB devices
Command:

```bash
free -h
```

Description:

**Display free and used memory** (-h for human readable, -m for MB, -g for GB)

Command:

```bash
mpstat 1
```

Description:

**Display processor related statistics**

Command:

```bash
vmstat 1
```
Display virtual memory statistics

Command:

iostat 1

Description:
Display Input / Output statistics

Command:

watch df -h

Description:
Execute "df -h" command, showing periodic updates

Command:

ps -ef

Description:
Display all the currently running processes on the system

**Command:**

```
ip a
```

**Description:**

Display all network interfaces and IP address

**Command:**

```
dig wikipedia.org
```

**Description:**

Display DNS information for domain (wikipedia.org)

**Command:**

```
host wikipedia.org
```
Display the IP address details of the specified domain (wikipedia.org)

**Command:**

netstat -nutlp

**Description:**

Display listening Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP) ports and corresponding programs

**Command:**

rpm -qa

**Description:**

List all installed packages

**Command:**

yum list installed

**Description:**

List all installed packages (CentOS)

**Command:**
**yum info httpd**

**Description:**

Display description and summary information about package "httpd" (CentOS)

**Command:**

`du -ah`

**Description:**

Display disk usage for all files and directories in human readable format

**Command:**

`du -sh`

**Description:**

Display total disk usage off the current directory

**Command:**
cd /etc

**Description:**

Change to the /etc directory

**Command:**

ps -A

**Description:**

List the status of all the processes along with process id and PID

**Command:**

```
#include <stdio.h>
int main()
{
    printf("Hello world\n");
    return 0;
}
```

```
gcc Hello.c
```
**Description:**

Compile the C program saved in Hello.c file

**Command:**

```cpp
#include <iostream>
int main()
{
  std::cout << "Hello world!";
  return 0;
}
```

`g++ Hello.cpp`

**Description:**

Compile the C++ program saved in Hello.cpp file

**Command:**

`tty`

**Description:**

Displays the file name of the terminal connected to standard input
**Command:**

```
public class MyClass {
public static void main(String [] args) {
    System.out.println("Hello, World!");
}
}
```

`javac MyClass.java`

**Description:**

Compile the Java program saved in `MyClass.java` file using javac compiler

**Command:**

```
od -b myfiles.txt
```

**Description:**

Displays the contents of `myfiles.txt` file in octal format

**Command:**

```
od -c myfiles.txt
```
Description:

Displays the contents of myfiles.txt file in character format

Command:

```
od -An -c myfiles.txt
```

Description:

Displays the contents of myfiles.txt file in character format but with no offset information

Command:

```
csplit myfiles.txt 13 62 101
```

Description:

If the file myfiles.txt has 123 lines, the csplit command would create four files: the xx00 file would contain lines 1–12, the xx01 file would contain lines 13–61, the xx02 file would contain lines 62–100, the xx03 file would contain lines 101–123

Command:
md5sum myfiles.txt

**Description:**
Prints a 32-character (128-bit) checksum of myfiles.txt file using the MD5 algorithm

**Command:**

```
more myfiles.txt
```

**Description:**
Displays the content of myfiles.txt file

**Command:**

```
sha1sum myfiles.txt
```

**Description:**
Prints SHA1 (160-bit) checksum of myfiles.txt file

```
SHA 1 → Secure Hash Algorithm 1
```

**Command:**
shred myfiles.txt

**Description:**

Overwrites the myfiles.txt file repeatedly – in order to make it harder for even very expensive hardware probing to recover the data

**Command:**

```bash
cat myfile.txt
01. Einstein
02. Newton
03. Maxwell
04. Tesla
05. Edison

tac myfile.txt
05. Edison
04. Tesla
03. Maxwell
02. Newton
01. Einstein
```

**Description:**

Print the lines of myfile.txt in reverse (from last line to first)
Command:

chkconfig --list

Description:

Displays a list of system services and whether they are started (on) or stopped (off) in run levels 0–6

Command:

chkconfig --list

Description:

Displays a list of system services and whether they are started (on) or stopped (off) in run levels 0–6

Command:

halt -p

Description:

Power-off the system
Command:

lastlog

Description:

Prints the details of the last login (login-name, port and last login time)

Command:

lastlog -t 1

Description:

Displays the login information (1 day ago)

Command:

lastlog -u manju

Description:

Display lastlog information for a particular user (manju)
**Command:**
cat /etc/passwd

more /etc/passwd

less /etc/passwd

getent passwd

**Description:**
List all users on Linux

**Command:**
tail -5 /etc/passwd

head -5 /etc/passwd
**Description:**

List last 5 users on Linux

List first 5 users on Linux

**Command:**

```
wall "The system will be shutdown in 10 minutes."
```

**Description:**

The message *(The system will be shutdown in 10 minutes.)* **will be broadcasted to** all users that are currently logged in

**Command:**

```
chage -l manju
```

**Description:**

List the password and its related details for a user *(manju)*
Command:

cage -M 10 manju

Description:

Set Password Expiry Date for an user (manju)

Command:

cage -E "2020-07-30" manju

Description:

Set the Account Expiry Date for an User (manju)

Command:

cage -I 10 manju

Description:

Force the user (manju) account to be locked after 10 inactivity days
Command:

ftp 192.168.42.77

Description:

Connect to an FTP server at remote server IP address "192.168.42.77"

Command:

arp –a

Description:

Lists all the peers connected at various interfaces along with their MAC Addresses and IP addresses

Command:

dnsdomainname

Description:
Display the system's DNS domain name

**Command:**

domainname

**Description:**
Display the name of the domain your machine belongs to

**Command:**

echo 'Hello World!' | base64

Output: SGVsbG8gV29ybGQhCg==

**Description:**
Encode text (Hello World!) to base64

**Command:**

echo 'SGVsbG8gV29ybGQhCg==' | base64 -d

Output: Hello World!

**Description:**
**Decode (SGVsbG8gV29ybGQh)** to **text** (Hello World!)

**Command:**

```
fccache -f -v
```

**Description:**

Build font information cache files

**Command:**

```
cat 1.txt
Einstein
Newton
Albert
fmt 1.txt
Einstein Newton Albert
```

**Description:**

Formats text in a single line
Albert Einstein was a German-born theoretical physicist, widely acknowledged to be one of the greatest physicists of all time. Einstein is known for developing the theory of relativity, but he also made important contributions to the development of the theory of quantum mechanics.
Albert Einstein was a German-born theoretical physicist, widely acknowledged to be one of the greatest physicists of all time. Einstein is known for developing the theory of relativity, but he also made important contributions to the development of the theory of quantum mechanics.
a German-born theoretical physicist, widely acknowledged to be one of the greatest physicists of all time. Einstein is known for developing the theory of relativity, but he also made important contributions to the development of the theory of quantum mechanics.

Command:

traceroute google.com

Description:

Prints the route that a packet takes to reach the Google (172.217.26.206) host from the local machine

Command:

cat 1.txt
Einstein
Newton
Albert

gzip 1.txt

zcat 1.txt.gz

Einstein
Newton
Albert

Description:
View the contents of zipped file

Command:

diff 1.txt.gz 2.txt.gz

Description:
Compare the contents of two zipped files (1.txt.gz, 2.txt.gz)
Command:

ss | less

Description:
List all connections

Command:

ss -aA tcp

Description:
Filter out TCP (Transmission Control Protocol) connections

Command:

ss -aA udp

Description:
Filter out UDP (User Datagram Protocol) connections
Command:

ss -lnt

Description:
Display only listening sockets

Command:

ss -ltp

Description:
Print process name and PID

Command:

ss -s

Description:
Print summary statistics
**Command:**

```
ss -t16
```

**Description:**

Display only IPv6 connections

---

**Command:**

```
ss -t1 -f inet
```

**Description:**

Display only IPv4 socket connections

---

**Command:**

```
ss -t4 state established
```

**Description:**

Display all IPv4 TCP sockets that are in connected state
Command:

pmap 3244

Description:

View the memory map of a process with Process ID (3244)

Command:

apropos -r 'remove file'

Description:

Find command that removes file

Command:

apropos editor

Description:

Display information about the editing programs that are available on a system
Command:

apropos pstree

Description:

Provide information about the pstree command (which displays the names of the processes currently on the system in the form of a tree diagram)

The apropos command is useful when you know what you want to do, but you have no idea what command you should be using to do it. If you were wondering how to locate files, for example, the commands

```
apropos find
```

and

```
apropos locate
```

would have a lot of suggestions to offer.

basename /etc/passwd

Output: passwd

basename /usr/local/apache2/conf/httpd.conf
Output: httpd.conf

```
echo a b c d e f | xargs
```

Output: a b c d e f

```
echo a b c d e f | xargs -n 3
```

Output: display only 3 items per line

a b c
d e f

**Command:**

```
env
```

**Description:**

Print out a list of all environment variables

**Command:**
printenv HOME

**Description:**

Print HOME variable value

cat score.txt

Albert-30
John-50
William-80
Stephen-20
Justin-40

cut -d- -f2 score.txt

30
50
80
20
40

cut -d- -f1 score.txt

Albert
John
William
Stephen
Justin
Command:

```
rev 1.txt
```

Description:

Reverse lines of a file (1.txt)

```
cat 3.txt
22
33
11
77
55

sort 3.txt
11
22
33
55
77
```

sorts numeric values in 3.txt file and displays sorted output
cat 1.txt
Hello World
cat 1.txt | tr "[a-z]" "[A-Z]"  \{ convert from lower case to upper case \}
HELLO WORLD

cat 5.txt
zz
zz
yy
yy
yy
xx
uniq 5.txt  \{ removes duplicate lines and displays unique lines \}
zz
yy
xx
Command:

ls -l *.txt

Description:

Lists the files with .txt extension

The thing with Linux is that the developers themselves are actually customers too: that has always been an important part of Linux.

Linus Torvalds
### Linux vs. Unix

<table>
<thead>
<tr>
<th>Linux</th>
<th>Unix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free to use <em>(open source)</em></td>
<td>Licensed Operating System <em>(closed source)</em></td>
</tr>
<tr>
<td>Linux is just the kernel</td>
<td>Unix is a complete package of Operating System</td>
</tr>
<tr>
<td>Bash <em>(Bourne Again SHell)</em> is default shell for Linux</td>
<td>Bourne Shell is default shell for Unix</td>
</tr>
<tr>
<td>Portable and is booted from a USB Stick</td>
<td>Unportable</td>
</tr>
<tr>
<td>Source code is accessible to the general public</td>
<td>Source code is not accessible to anyone</td>
</tr>
<tr>
<td>Uses Graphical User Interface with an optional Command Line Interface</td>
<td>Uses Command Line Interface</td>
</tr>
</tbody>
</table>

**Command:**

```bash
echo $SHELL
```

**Description:**

Print the Default shell of user

**Command:**

```bash
echo $0
```

**Description:**
Display the name of the currently running process ($0 is the name of the running process). If you use it inside of a shell then it will return the name of the shell. If you use it inside of a script, it will return the name of the script

**Command:**

```bash
echo *
```

**Description:**

Print all files and folders – similar to `ls` command

**Command:**

```bash
ps -p $$
```

**Output:**

```
PID TTY TIME CMD
3352 pts/0 00:00:00 bash
```

**Description:**

Print the process ID of the current shell ($$ is the process ID of the current shell)

**Command:**
**Description:**

List shells

**Command:**

`last`

**Description:**

List last logins of users and what happened such as "shutdown" or "crash" etc.

**Command:**

`last`
**Description:**

Compresses but does not deletes the original file

```
phy.txt → phy.txt.bz2
```

**Command:**

```
bzip2 -k phy.txt
```

**Description:**

Decompresses the compressed file (phy.txt.bz2)

```
phy.txt.bz2 → phy.txt
```

**Command:**

```
bzcat phy.txt.bz2
```
**Description:**

Display the contents of compressed file (phy.txt.bz2)

**Command:**

```
bunzip2 phy.txt.bz2
```

**Description:**

Decompresses the compressed file (phy.txt.bz2)

**Command:**

```
crontab -l
```

**Description:**

Display current logged-in user's crontab entries

**Command:**

```
cat /dev/null > phy.txt
```
cp /dev/null phy.txt

echo "" > phy.txt

Description:

Empty the content of a file (phy.txt)

Command:

nohup ping google.com &

Description:

Ping google.com and send the process to the background

Command:
**Description:**

Display the contents of `/home`

**Command:**

```
sudo shutdown 2
```

**Description:**

Power-off the machine after 2 minutes

**Command:**

```
shutdown -c
```

**Description:**

Cancel the shutdown process

**Command:**

```
pr 36.txt
```
Description:

Display the contents of the file (36.txt) one page after the other

Command:

```
stty -a
```

Description:

Display all current terminal settings

Command:

```
ls -l
```

Description:

List files one per line

Command:
yes John

Description:

Outputs a string (John) repeatedly until killed

Command:

dir

Description:

List files and directories in the current directory (one per line) with details

Command:

who -b

Description:

Print when the system was booted

# Open phy.txt with nano
nano phy.txt

# Open phy.txt with vim

vim phy.txt

```
User Request
  ↓
  Shell  Application
  ↓
  Linux Kernel
  ↓
  Computer Hardware
```

w -ip-addr
# Displays information regarding the users currently on the machine, login time, IDLE time, TTY and CPU time

Output:

11:12:10 up 1:29, 2 users, load average: 0.02, 0.04, 0.10
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
manju :0 :0 02:43 ?xdm? 3:30 0.65s gdm-session-worker [pa
manju pts/0 :0 11:01 2.00s 0.10s 0.01s w --ip-addr

w -short
# Omits CPU time and login information
Output:

11:11:46 up 1:28, 2 users, load average: 0.02, 0.04, 0.11
USER TTY FROM IDLE WHAT
manju :0 :0 ?xdm? gdm-session-worker [pam/gdm-password
manju pts/0 :0 2.00s w --short

**Command:**

```
findmnt
```

**Description:**

Display a list of currently mounted file systems
ip addr show

**Description:**
List IP addresses and network interfaces

**Command:**
netstat -pnltu

**Description:**
List active (listening) ports

**Command:**
journalctl

**Description:**
Display systemd, kernel and journal logs

**Command:**
sudo systemctl status network

**Description:**

Display the status of network service

**Command:**

sudo systemctl start network

**Description:**

Start the network service

**Command:**

sudo systemctl stop network

**Description:**

Stop the network service

**Command:**
**Description:**

Display the current state of Booleans

**Command:**

`getenforce`

**Description:**

Reports whether SELinux is enforcing, permissive or disabled

**Security-Enhanced Linux (SELinux)** is a security architecture for Linux systems that allows administrators to have more control over who can access the system

**Description:**

`setenforce 0`
getenforce

Output:
Permissive

setenforce 1

getenforce

Output:
Enforcing

- **Enforcing** - SELinux security policy is enforced.
- **Permissive** - SELinux prints warnings instead of enforcing.
- **Disabled** - No SELinux policy is loaded.
Command:

sestatus

Description:
Display the current status of the SELinux that is running on your system

Command:

ps -aef

Description:
Display full listing of processes on your system

Command:

sar

Description:
Display System Activity Report

Command:

`ulimit`

Description:

Report the resource limit of the current user

Output:

```
Unlimited
```

The current user can consume all the resources the current system supports

2 types of resource limitation:

- **Hard resource limit**: The physical limit that the user can reach.
- **Soft resource limit**: The limit that is manageable by the user *(its value can go up to the hard limit)*

Command:
ulimit -a

**Description:**

Report all the resource limits for the current user

**Command:**

ulimit -s

**Description:**

Check the maximum stack size of the current user

**Command:**

ulimit -e

**Description:**

Check out the max scheduling priority of the current user
Command:

ulimit -u

Description:
Display the maximum number of user processes

Command:

ulimit -v

Description:
Check out the size of virtual memory

Command:

ulimit -n

Description:
Check out how many file descriptors a process can have
**Command:**

```
man limits.conf
```

**Description:**

Display the in-depth information on the `limits.conf` configuration file

---

**Command:**

```
sar -V
```

**Description:**

Display the sar version

---

**Command:**

```
sar -u 2 5
```

**Description:**

Report CPU details total 5 times with the interval of 2 seconds
**Command:**

```
sar -n DEV 1 3 | egrep -v lo
```

**Description:**

Report about network interface, network speed, IPV4, TCPV4, ICMPV4 network traffic and errors

**Command:**

```
sar -v 1 3
```

**Description:**

Report details about the process, kernel thread, i-node, and the file tables

**Command:**

```
sar -S 1 3
```

**Description:**

Report statistics about swapping
Command:

```
sar -b 1 3
```

Description:

Report details about I/O operations like transaction per second, read per second, write per second

Command:

```
sudo systemctl status firewalld
```

Description:

Display the status of the `firewalld`

Command:

```
sudo systemctl start firewalld
```

Description:

Start the `firewalld` service
Command:

**firewall-config**

Description:

Start the graphical firewall configuration tool

---

Command:

**firewall-cmd**

**firewall-cmd --list-all-zones**

Description:

List all zones

---

**firewalld** is a firewall management tool for Linux operating systems
**Description:**

`firewall-cmd --get-default-zone`

Check the currently set default zone

**Command:**

`firewall-cmd --list-services`

**Description:**

Display currently allowed service on your system

**Command:**

`firewall-cmd --list-ports`

**Description:**

List the ports that are open on your system

**Command:**
firewall-cmd --zone=work --list-services

**Description:**

List services that are allowed for the public zone

**Command:**

mtr --report google.com

**Description:**

Provides information about the route that Internet traffic takes between the local system and a remote host (google.com)

**Command:**

sudo yum install samba

**Description:**

install Samba (CentOS)
Samba is client/server technology that implements network resource sharing across operating systems. With Samba, files and printers can be shared across Windows, Mac and Linux/UNIX clients.

**Command:**

```
sudo firewall-cmd --add-service samba --permanent
```

**Description:**

Add Samba service to firewall

**Command:**

```
zip q.zip q.txt
```

**Description:**

Create a zip file (q.zip)

**Command:**
unzip q.zip

**Description:**

Unzip a zip file (q.zip)

zipcloak q.zip

# zipcloak prompts you for a password, and then ask you to confirm it:

   Enter password:
   Verify password:

...if the passwords match, it encrypts q.zip file

unzip q.zip

# When you try to unzip the q.zip file, it prompts you for the password before allowing you to extract the file (q.txt) it contains
**Command:**

    zgrep -l "Einstein" *

**Description:**

Display the names of the files with the word (Einstein) present in it

**Command:**

    zipsplit -n 1048576 q.zip

**Description:**

Split q.zip file to create a sequence of zipfiles (q1.zip, q2.zip.....) – each no larger than 1048576 bytes (one megabyte)

You could concatenate (q1.zip, q2.zip.....) into a new file, w.zip, with the command:

    cat q*.zip > w.zip
Git Commands

Description:

*Display information about previous commits.*

Command:

git log

Description:

*Display information about previous commits (detailed).*

Command:
Description:

Display information about previous commits (briefly).

Command:

git log --summary

Description:

Obtain the repository "Git-Commands" from the URL "https://github.com/manjunath5496/Git-Commands.git".

Command:

git clone https://github.com/manjunath5496/Git-Commands.git

Description:

Display most commonly used git commands.
Command:

```bash
git help
```

**Description:**

*Display git version.*

Command:

```bash
git version
```

**Description:**

*Set the basic configurations on github (your name and email).*

Command:

```bash
git config --global user.name "myw3schools"
git config --global user.email myw3schools@gmail.com
```
Description:

*Check status.*

Command:

git status

Description:

*List all branches (local and remote).*

Command:

git branch -a

Description:

*Display Git configurations.*

Command:
Description:

Add an empty file "test.txt" to an existing repo "colors".

Command:

touch test.txt
git init
git add test.txt
git commit -m "first commit"
git remote add origin git@github.com:myw3schools/colors.git
git push -u origin main

My first official teaching job was at GIT, which was fantastic because I wanted to pay the rent and I got to stay in the building, which is an inspiring place to be - the vibe was there. My first gig was doing private lessons. It went great. Then they decided to promote me to a classroom teacher. I taught a class called Single String Technique.

Paul Gilbert
Linus Benedict Torvalds is a Finnish-American software engineer who is the creator and, historically, the main developer of the Linux kernel, used by Linux distributions and other operating systems such as Android and Chrome OS.

References:

- DevOps For Dummies By Emily Freeman
- The Future of DevOps By Tom Smith
- Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale By Jennifer Davis