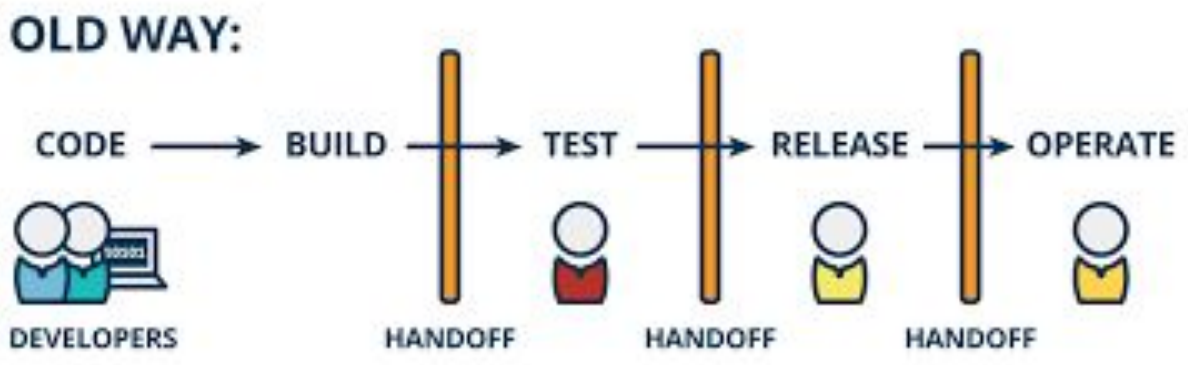
# BATCH 5.0 DEVOPS BRIDGE CLASS

## The Core Principles of DevOps

DevOps revolves around a set of core principles that guide its implementation and practices. These principles include:

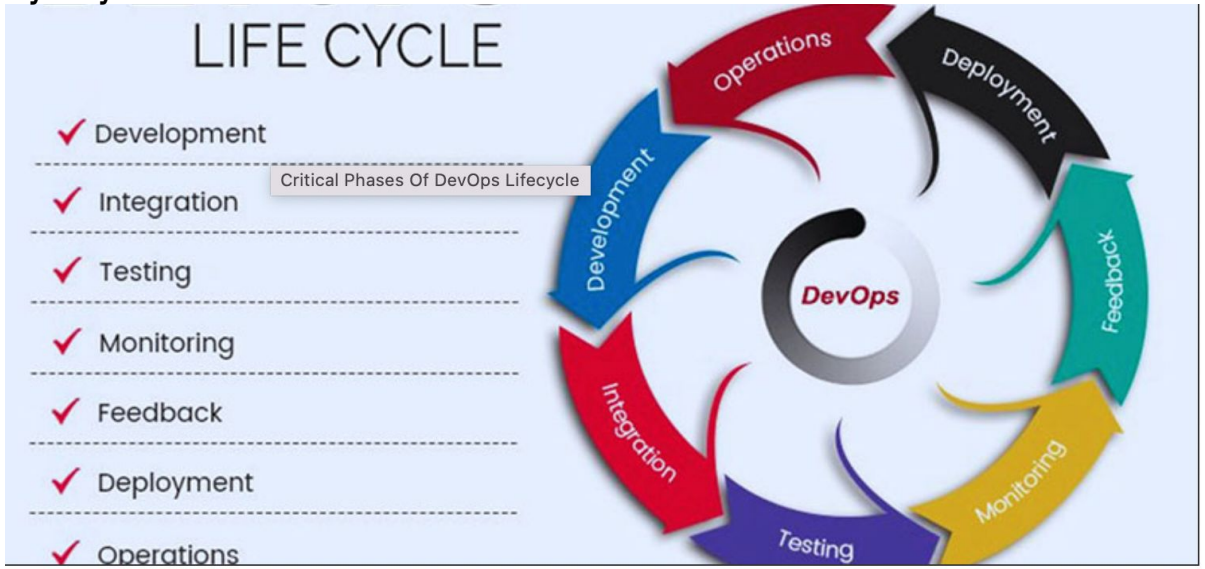
- **Collaboration** – DevOps encourages open communication and collaboration between development and operations teams, fostering a culture of shared responsibility and accountability.
- **Automation** – Automation is a cornerstone of DevOps. It streamlines processes, reduces human error, and allows for faster delivery and improved consistency.
- **Continuous Integration and Continuous Delivery (CI/CD)** – CI/CD is the practice of automatically building, testing, and deploying software changes to production. This approach ensures a continuous flow of high-quality software releases.
- **Feedback and Monitoring** – DevOps emphasizes the importance of monitoring and gathering feedback from both technical and business stakeholders to improve processes and continuously iterate on products.
- **Continuous Learning and Improvement** – A commitment to continuous learning and improvement is essential in a DevOps culture. Teams should regularly evaluate their processes and tools to identify areas for growth and optimization.



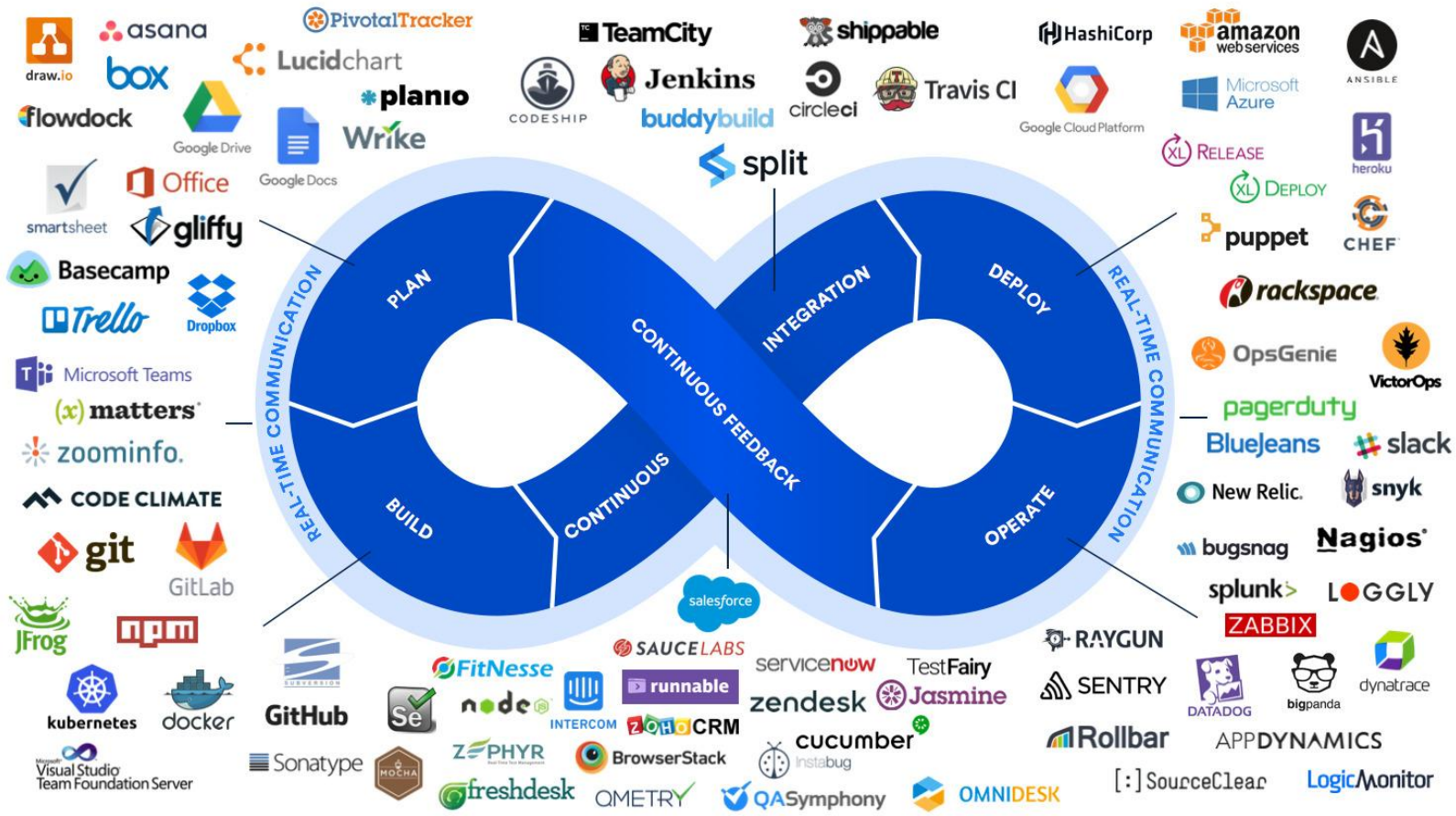
# Benefits of Implementing DevOps in Your Organization

Adopting a DevOps mindset can bring about significant benefits for your organization. These include:

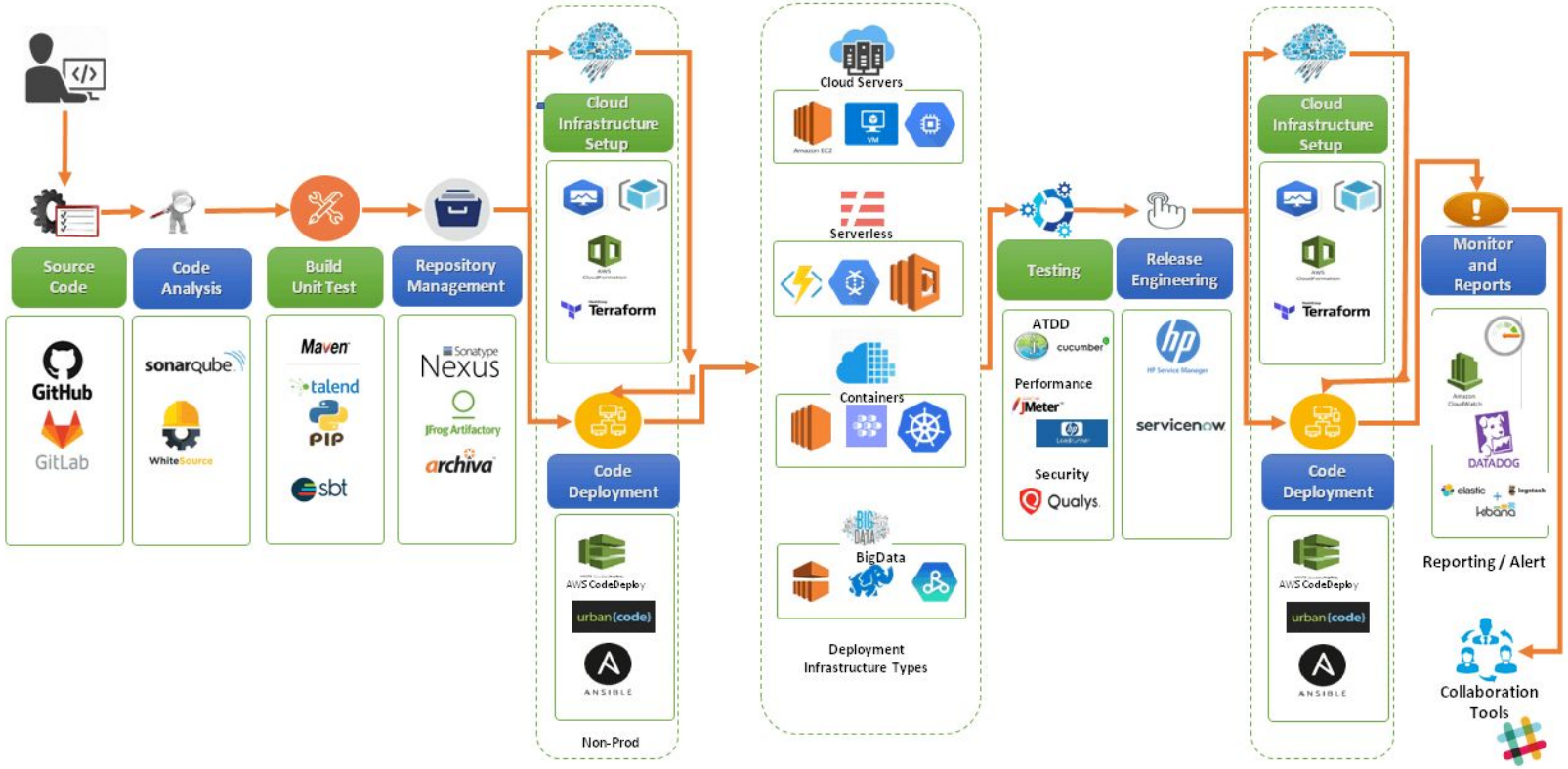
- **Faster Delivery and Time-to-Market** – DevOps practices enable quicker software releases, allowing your organization to respond more effectively to market changes and customer demands.
- **Improved Quality and Reliability** – Through automation, continuous integration, and testing, DevOps helps ensure that the software delivered is of high quality, reliable, and secure.
- **Enhanced Collaboration and Communication** – DevOps fosters a culture of open communication and collaboration, breaking down silos between teams and leading to better alignment, shared goals, and more efficient processes.
- **Cost Savings** – By reducing the time spent on manual tasks and streamlining workflows, DevOps can lead to significant cost savings for your organization.
- **Increased Customer Satisfaction** – With faster delivery and higher-quality products, your organization can better meet customer expectations, resulting in increased satisfaction and loyalty.



# The Big Picture



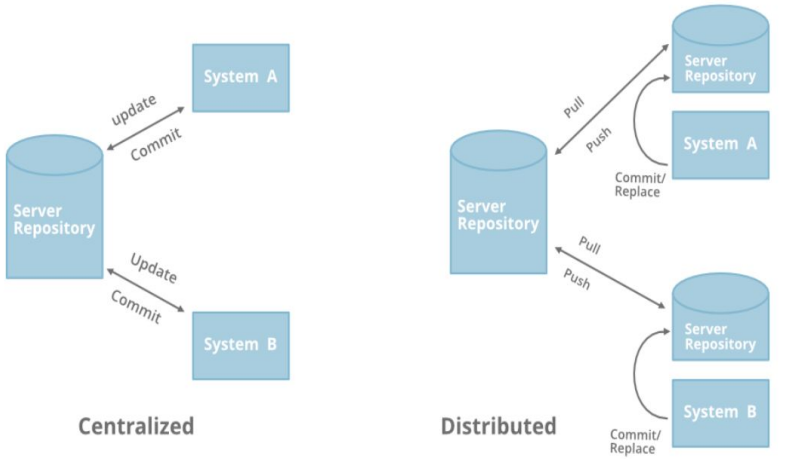
# The DevOps Flow



**Git** is not **Github**. Git is the version control software, and Github is a git repository hosting service which offers all the source code management provided in git. Github is where you upload your git repository.

## Centralized Version Controlling

The systems such as CVS, Subversion, and Perforce have a single server that contains all the versioned files, and a number of clients that check out files from that central place.




## Distributed Version Controlling

In a DVCS (such as Git), clients don't just check out the latest snapshot of the files; rather, they fully mirror the repository, including its full history. Thus, if any server dies, and these systems were collaborating via that server, any of the client repositories can be copied back up to the server to restore it. Every clone is really a full backup of all the data.

# Three Stages in GIT

## The Working Tree/untracked (current state of the project ) -- git status

The Working Tree is the area where you are currently working. It is where your files live. This area is also known as the "untracked" area of git. Any changes to files will be marked and seen in the Working Tree

```
D:\jQueryUI>git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        currency-spinner.html
        datepicker-dateformat.html
        datepicker-restricting-date.html
        datepicker-with-dropdown.html
        datepicker.html
        globalize.culture.ar-YE.js
        globalize.culture.de-DE.js
        globalize.culture.en-US.js
        globalize.js
        spinner-min-max.html
        spinner-restrict-keyboard.html
        spinner-step-value.html
        spinner.html

nothing added to commit but untracked files present (use "git add" to track)

D:\jQueryUI>
```

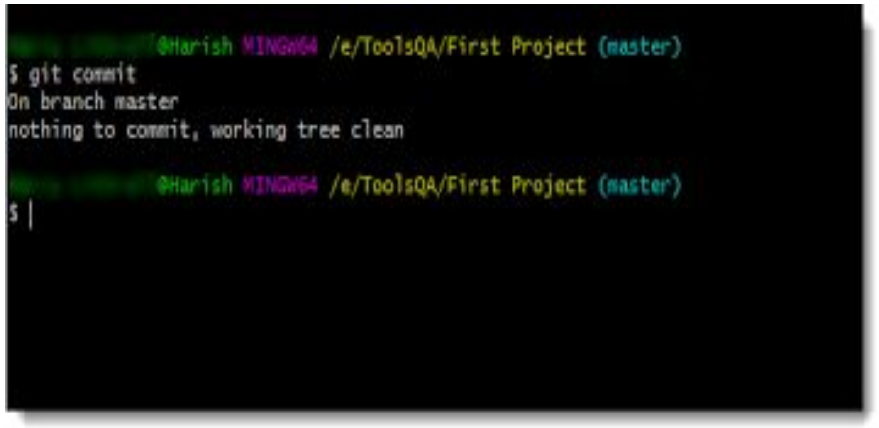
## The Staging Area (Index) -- git add

The Staging Area is when git starts tracking and saving changes that occur in files. These saved changes reflect in the .git directory. That is about it when it comes to the Staging Area.

```
~/p/p/slides (:gh-pages) git status
# On branch gh-pages
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   css/theme/jr0cket.css
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   getting-started-with-git.html
#       modified:   getting-started-with-git.org
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       images/
~/p/p/slides (:gh-pages) _
```

# Local Repository/Commit Region -- git commit

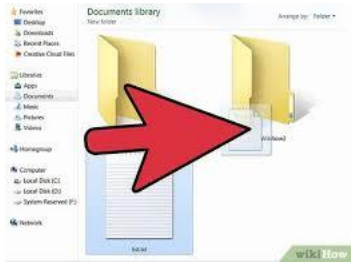
The Local Repository is everything in your .git directory, add items from your Staging Area to your Local Repository




- git init → Create a new git repository
- git add “newfile” → Add a new file to your staging area
- git commit → Adds staged changes to your local repository
- git push “remote” “ branch” → Push local repository changes to your hosting service
- git pull “remote” “ branch” → pull code from your hosting service to your local directory
- git branch → See local branches
- git branch “newName” → Create new local branch
- git checkout “branchName” → Switch branches
- git diff → See the actual difference in code between your working tree and your staging area
- git status → Show which files are being tracked v. untracked
- git log → Show recent commit history
- git show “commit_id” → show details of specific commit
- git stash → stash working directory
- git help → manpages for git
- git help “gitCommand” → man pages for specific git command

Files Edit


Git Folder



## 1) Working tree status
## Command : git status

```
D:\jQueryUI>git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        currency-spinner.html
        datepicker-dateformat.html
        datepicker-restricting-date.html
        datepicker-with-dropdown.html
        datepicker.html
        globalize.culture.ar-YE.js
        globalize.culture.de-DE.js
        globalize.culture.en-US.js
        globalize.js
        spinner-min-max.html
        spinner-restrict-keyboard.html
        spinner-step-value.html
        spinner.html

nothing added to commit but untracked files present (use "git add" to track)

D:\jQueryUI>
```

## 2) The Staging Area
## Command :git add

```
~/p/p/slides (:gh-pages) git status
# On branch gh-pages
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   css/theme/jr0cket.css
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   getting-started-with-git.html
#       modified:   getting-started-with-git.org
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       images/
~/p/p/slides (:gh-pages) _
```

## Local Repository/Commit Area
## Command : git commit

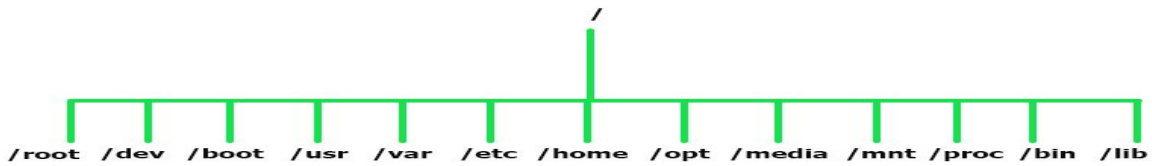
```
@Harish MINGW64 /e/ToolsQA/First Project (master)
$ git commit
On branch master
nothing to commit, working tree clean

@Harish MINGW64 /e/ToolsQA/First Project (master)
$
```

Git Push
to Repo

## Linux File Management

In Unix, there are three basic types of files –

- **Ordinary Files –** An ordinary file is a file on the system that contains data, text, or program instructions. In this tutorial, you look at working with ordinary files.
- **Directories –** Directories store both special and ordinary files. For users familiar with Windows or Mac OS, Unix directories are equivalent to folders.
- **Special Files –** Some special files provide access to hardware such as hard drives, CD-ROM drives, modems, and Ethernet adapters. Other special files are similar to aliases or shortcuts and enable you to access a single file using different names.




- The **/root** directory is the home directory for the root user.
- **The /dev directory contains device files such as /dev/sda.**
- Static boot files are located in the **/boot** directory.
- Applications and user utilities are found in the **/usr** directory.
- The **/var/logs** directory contains log files of various system applications.
- **All system configuration files are stored in the /etc directory.**
- The **/home** directory is where user folders are located. These include Desktop, Documents, Downloads, Music, Public, and Videos.
- For add-on application packages, check them out in the **/opt** directory.
- The **/media** directory stores files for removable devices such as USB drives.
- The **/mnt** directory contains subdirectories that act as temporary mount points for mounting devices such as CD-ROMs.
- **The /proc directory is a virtual filesystem that holds information on currently running processes. It's a strange filesystem that is created upon a system boot and destroyed upon shutdown.**
- The **/bin** directory contains user command binary files.
- The **/lib** directory stores shared library images and kernel modules.

**Editors in Linux**

**Vi/Vim Editor**

Vim is a powerful command-line based text editor that has enhanced the functionalities of the old Unix Vi text editor.

$ vim (name of the file)
To start writing or editing
shift + i on your keyboard (“I” for insert)

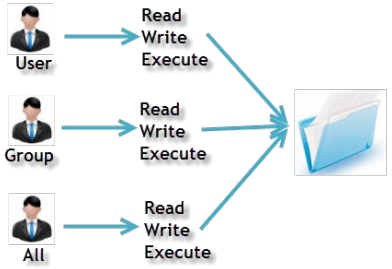
**Nano**

Nano is a newer text editor in Linux systems. It’s simpler and easier to use than vim.

$ nano (name of the file)

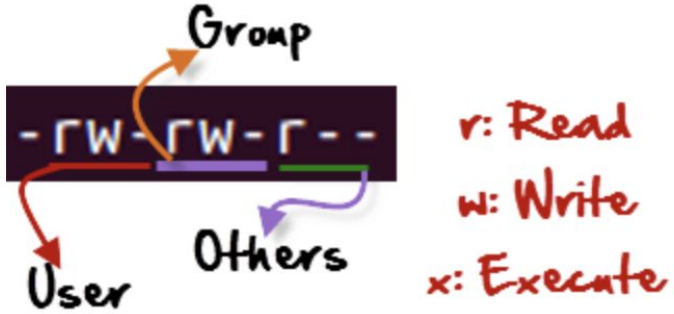
- `h` - Moves the cursor to the left by one character; you can also press the left arrow.
- `j` - Moves the cursor one line down; you can also press the down arrow.
- `k` - Moves the cursor one line up; you can also press the up arrow.
- `l` - Moves the cursor to the right by one character; you can also press the right arrow.
- `w` - Moves the cursor one full word to the right.
- `b` - Moves the cursor one full word to the left.
- `0` - Moves the cursor to the beginning of the current line.
- `$` - Moves the cursor to the end of the current line.
- `~` - Changes the case of the current character.
- `dd` - Deletes the current line.
- `D` - Deletes everything on the line to the right of the cursor’s current position.
- `x` - Deletes the current character.
- `u` - Undo the last command.
- `.` - Repeats the last command.
- `:w` - Saves current file, but does not exit.
- `:wq` - Saves current file, and quits.

Owners assigned Permission On Every File and Directory




| User Denotations | |
|---|---|
| u | user/owner |
| g | group |
| o | other |
| a | all |

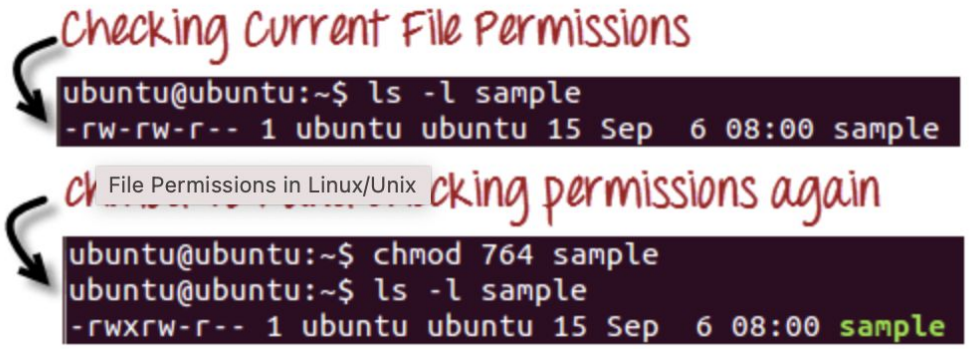





**r** = read permission
**w** = write permission
**x** = execute permission
**–** = no permission

## Absolute(Numeric) Mode in Linux

In this mode, file permissions are not represented as characters but a three-digit octal number.

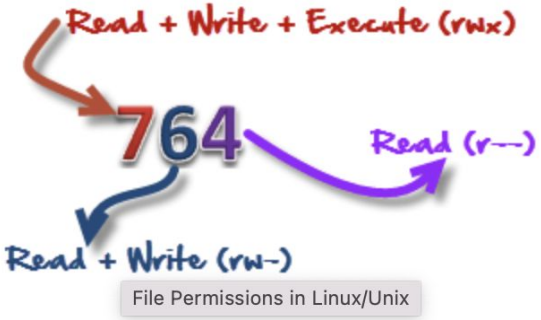



| Number | Permission Type | Symbol |
|---|---|---|
| 0 | No Permission | — |
| 1 | Execute | –x |
| 2 | Write | -w- |
| 3 | Execute + Write | -wx |
| 4 | Read | r– |
| 5 | Read + Execute | r-x |
| 6 | Read +Write | rw- |
| 7 | Read + Write +Execute | rwx |

'764' absolute code says the following:

- Owner can read, write and execute
- Usergroup can read and write
- World can only read

**This is shown as '-rwxrw-r–**



# AWS FUNDAMENTALS