DevSecOps with AWS - 88S

Table of Contents

27-JAN-2026 Session-14

Inode Symlink Hardlink

[ec2-user@ip-172-31-31-187 ~]$ ls

linux.txt

[ec2-user@ip-172-31-31-187 ~]$ ls -li

total 0

8768976 -rw-r--r--. 1 ec2-user ec2-user 0 Jan 30 11:34 linux.txt

[ec2-user@ip-172-31-31-187 ~]$ stat linux.txt

File: linux.txt

Size: 0 Blocks: 0 IO Block: 4096 regular empty file

Device: 10301h/66305d ***Inode: 8768976*** Links: 1

Access: (0644/-rw-r--r--) Uid: (1000/ec2-user) Gid: (1000/ec2-user)

Context: unconfined_u:object_r:user_home_t:s0

Access: 2026-01-30 11:34:36.344353743 +0000

Modify: 2026-01-30 11:34:36.344353743 +0000

Change: 2026-01-30 11:34:36.344353743 +0000

Birth: 2026-01-30 11:34:36.344353743 +0000

[ec2-user@ip-172-31-31-187 ~]$ vim linux.txt

[ec2-user@ip-172-31-31-187 ~]$ stat linux.txt

File: linux.txt

Size: 13 Blocks: 8 IO Block: 4096 regular file

Device: 10301h/66305d Inode: 8768996 Links: 1

Access: (0644/-rw-r--r--) Uid: (1000/ec2-user) Gid: (1000/ec2-user)

Context: unconfined_u:object_r:user_home_t:s0

Access: 2026-01-30 12:58:21.325056651 +0000

Modify: 2026-01-30 12:58:21.325056651 +0000

Change: 2026-01-30 12:58:21.325056651 +0000

Birth: 2026-01-30 12:58:21.325056651 +0000

Inode

Stores metadata of the file/directory

File type -> file/directory/link

Owner group

Permissions

Size

Create, update, etc

Actual memory block address

Inode does not store actual file content.

Symlink == shortlink == softlink

1. Short cut to the file/folder
2. Inode number is different for symlink
3. If actual file is removed, then symlink will break
4. Uses for backward compatability, switching versions and rollback is easy
5. You can create symlink for folders and files also
6. You can create symlink across different filesystem

Steps:

1. Create one directory(links)
2. Go to link directory
3. Create link for linux.txt
   Command: ln -s
   $ ln -s ../linux.txt link
4. Check link status
   Command: stat link

File: link -> ../linux.txt
Size: 12 Blocks: 0 IO Block: 4096 ***symbolic link***
Device: 10301h/66305d ***Inode: 8768976*** Links: 1

Note: Inode numbers of Symlinks are different

which dnf

/usr/bin/dnf

$ ls -l /usr/bin/dnf

lrwxrwxrwx. 1 root root 5 Feb 7 2025 /usr/bin/dnf -> dnf-3

$ ls -l /usr/bin/yum

lrwxrwxrwx. 1 root root 5 Feb 7 2025 /usr/bin/yum -> dnf-3

YUM -> Before RHEL-8 YUM is the package manager

RHEL-8 DNF is the package manager

```
$ touch dnf-3
$ ls
dnf-3  links
$ ln -s dnf-3 dnf
$ ls
dnf  dnf-3  links
$ touch dnf-4
$ ls
dnf  dnf-3  dnf-4  links
$ ls -l
lrwxrwxrwx. 1 ec2-user ec2-user  5 Jan 31 13:20 dnf -> dnf-3
-rw-r--r--. 1 ec2-user ec2-user  0 Jan 31 13:16 dnf-3
-rw-r--r--. 1 ec2-user ec2-user  0 Jan 31 13:25 dnf-4
drwxr-xr-x. 2 ec2-user ec2-user 18 Jan 31 12:13 links

$ ln -sfn dnf-4 dnf
$ ls -l

lrwxrwxrwx. 1 ec2-user ec2-user  5 Jan 31 13:26 dnf -> dnf-4
-rw-r--r--. 1 ec2-user ec2-user  0 Jan 31 13:16 dnf-3
-rw-r--r--. 1 ec2-user ec2-user  0 Jan 31 13:25 dnf-4
drwxr-xr-x. 2 ec2-user ec2-user 18 Jan 31 12:13 links
```

This is the switching versions safely without downtime.

If anything, problem occurs in dnf -4 we can revert back

```
$ ln -snf dnf-3 dnf
$ ls
dnf  dnf-3  dnf-4  links
$ ls -l
total 0
lrwxrwxrwx. 1 ec2-user ec2-user  5 Jan 31 13:46 dnf -> dnf-3
-rw-r--r--. 1 ec2-user ec2-user  0 Jan 31 13:16 dnf-3
-rw-r--r--. 1 ec2-user ec2-user  0 Jan 31 13:25 dnf-4
drwxr-xr-x. 2 ec2-user ec2-user 18 Jan 31 12:13 links
```

App-3

App-4

```
$ mkdir app-3
ls
app-3  dnf  dnf-3  dnf-4  links
 $ ln -s app-3 app
$ ls -l
total 0
lrwxrwxrwx. 1 ec2-user ec2-user  5 Jan 31 13:56 app -> app-3
drwxr-xr-x. 2 ec2-user ec2-user  6 Jan 31 13:53 app-3

$ mkdir app-4
$ ln -s app-4 app
$ ls -l
total 0
lrwxrwxrwx. 1 ec2-user ec2-user  5 Jan 31 13:56 app -> app-3

 $ ln -sfn app-4 app
 $ ls -l
total 0
lrwxrwxrwx. 1 ec2-user ec2-user  5 Jan 31 13:57 app -> app-4
```

/var/lib/jenkins -> jenkins-2.8

Jenkins-2.9

/var/lib/jenkins -> jenkins-2.9

Hardlink

1. You can't create hardlink in another filesystem

   $ touch aws.txt

   $ vim aws.txt

   $ ls -l

   -rw-r--r--. 1 ec2-user ec2-user 20 Jan 31 14:18 aws.txt

   $ ln aws.txt /tmp/aws-1.txt

   ln: failed to create hard link '/tmp/aws-1.txt' => 'aws.txt': Invalid cross-device link

2. Hardlink is same as original file but can be different name

   ln aws.txt aws-1.txt

   $ ls -l

   -rw-r--r--. 2 ec2-user ec2-user 20 Jan 31 14:18 aws-1.txt

   -rw-r--r--. 2 ec2-user ec2-user 20 Jan 31 14:18 aws.txt

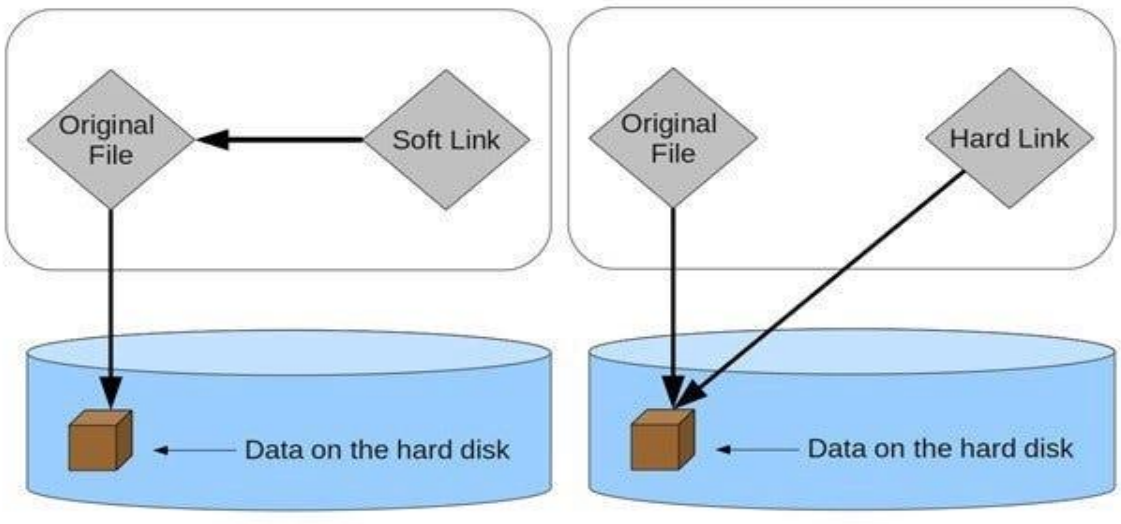
3. Hardlink inode numbers are same.

   $ ls -li

   8769002 -rw-r--r--. 2 ec2-user ec2-user 20 Jan 31 14:18 aws-1.txt

   8769002 -rw-r--r--. 2 ec2-user ec2-user 20 Jan 31 14:18 aws.txt

4. They points to same memory location, so memory is not wasted



5. You can't create hardlink to folders

6. Hardlinks are used for backup.

7. When all links are deleted, then only file will be deleted.

```
$ rm aws.txt
 $ ls -l
-rw-r--r--. 1 ec2-user ec2-user 20 Jan 31 14:18 aws-1.txt
$ cat aws-1.txt
I am learning linux
```

Using inode number and find command within the same File System.

```
$ ln aws-1.txt app-3/aws.txt
$ ln aws-1.txt app-4/aws.txt
```

```
cd app-3/
$ ls
app-4  aws.txt
$ cd app-4
$ ls
aws.txt
[ec2-user@ip-172-31-28-112 ~]$ ls -li
1639586 drwxr-xr-x. 2 ec2-user ec2-user 34 Jan 31 15:40 app-3
8768999 drwxr-xr-x. 2 ec2-user ec2-user 21 Jan 31 15:40 app-4
8769002 -rw-r--r--. 3 ec2-user ec2-user 20 Jan 31 14:18 aws-1.txt
$ sudo find / -inum 8769002
/home/ec2-user/app-3/aws.txt
/home/ec2-user/app-4/aws.txt
/home/ec2-user/aws-1.txt
```

Disadvantages of doing manual

1. Time taking

2. Human errors

3. Everytime from the scratch

4. Can't rely

5. If scale is increased manual way is not possible

Scripting

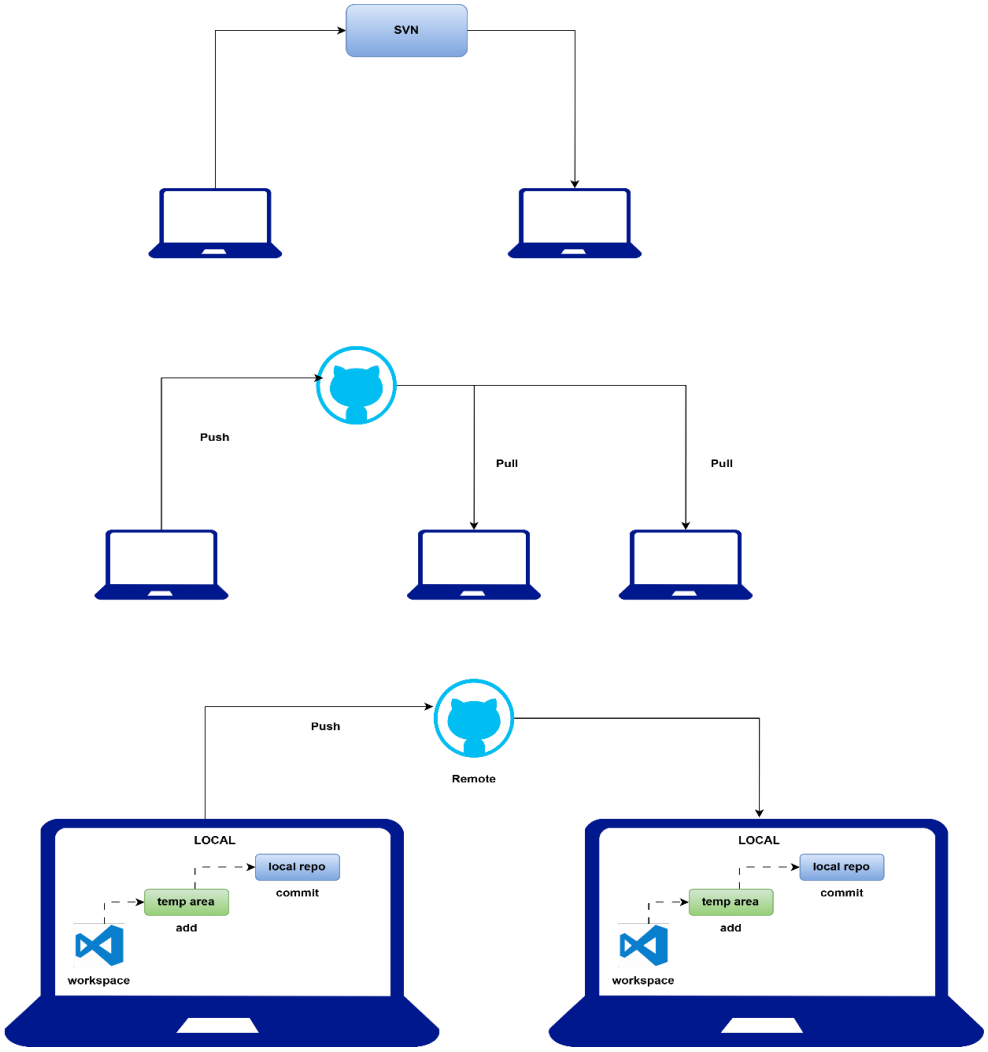
Commands in a file and run it called scripting.

Where to store scripts?

Version control system

Git -> VCS

1. Version control
2. Who did the changes, when did the changes
3. You can track who did the changes
4. Restoring is easy
5. Review is also easy -> Pull Requests
6. Secure place to store the code



Centralised vs distributed

SVN -> sub version control

10 files

9 files

Staging/temp area -> add 9 files to staging area

Then commit to local repo

Git commands

1. Clone the repo -> first time download

   Main/master is the default branch

   #!/bin/bash -> shebang

   Shebang is the interpreter of the code in shell script that checks and compiles for errors

   Bash scripting == shell scripting

2. Add files to staging area

   Git add <filename>

   Git config --global user. email "info@joindevops.com"

   Git config --global user.name "Sivakumar Reddy M"

3. Commit to local repo

   Git commit -m "hello world completed"

   Git push origin main

   Don't edit files inside linux server...

28-JAN-2026 Session-15

1. Create repo either in organization or directly
2. Clone repo in laptop
3. write something
4. git add., git add <filename>
5. git commit -m "message"
6. git push origin main

git config --global user.name

git config --global user.email

write only in laptops

don't write in linux servers

only pull linux

95% thinking+design 5% coding

pseudo code -> no syntax, just assumption

if today is sunday, enjoy the holiday otherwise go to school

```
today=Wednesday
if(today != sunday){
        print ("goto school")
}
else{
        print("take holiday, enjoy")
}
```

15

15/2 -> 1

if reminder is 0 -> even

otherwise, odd

```
number=15
reminder=15%2
reminder=1
if(reminder == 0){
        print("even number")
}
else{
        print("odd number")
}
```

variables

data types

conditions

functions

loops

error handling

let x=1, y=0

derive the formula

submit finally

variables are like containers that holds something

DRY -> Don’t repeat yourself

change in one place reflect everywhere you refer

easy to maintain

readable

PERSON1=Suresh

PERSON2=Ramesh

$VAR_NAME

${VAR_NAME}

sh 03-conversation.sh Suresh Ramesh

args == parameters

$1=Suresh

$2=Ramesh

echo "Enter your password"

read -s PASSWORD

read -s encrypts the password

[ec2-user@ip-172-31-28-112 Shell-practice]$ sh 05-Variables.sh

Enter your username::

Sivaram

Username is

Enter your password

I want a command to be executed and take the output into variable, how to do that?

VAR_NAME=$(command)

Special variables

==========

$1 $2 ... $N args passed to script

All variables passed to script: $@

Number of variables passed to script: $#

Script name: $0

Present which directory you are in: $PWD

Who is running this script: $USER

Home direcory of the user: $Home

PID of the script: $$

sleep 100 &

Background process id: $!

Exit status of previous command: $?

$@ vs $*

==========

$@ treats args separately

$* treats as single args

data types

==========

variables are holding data..

mobile number -> numbers

integers -> -33,768 -> 33,768

float -> 45.90

decimanl -> long number

complex -> 4+8i

names -> string

major? -> yes or no boolean

skills -> devops aws docker kubernetes -> list of skills

skills -> map or dictionary

devops: 4 -> key -> value

docker: 3

kubernetes: 2

areane4 -> 5

speed round up -> 8.2km/sec -> 8km/sec

copied code

integer speed=8km/sec

integer speed=8.2km/sec

number

string

everything is string in shell

position/index in coding starts from 0

conditions
==========

if or when

```
if [ expression ]; then
        code here
fi

if [ expression ]; then
        code here
elif [ expression ]; then
        code here
else
        code here
fi
```

EXIT code
==========
0-127

0 -> success
anything else -> failure

nginx

dnf install nginx -y

if root user then success

otherwise failure

root user id -> 0

other users -> greater than 0