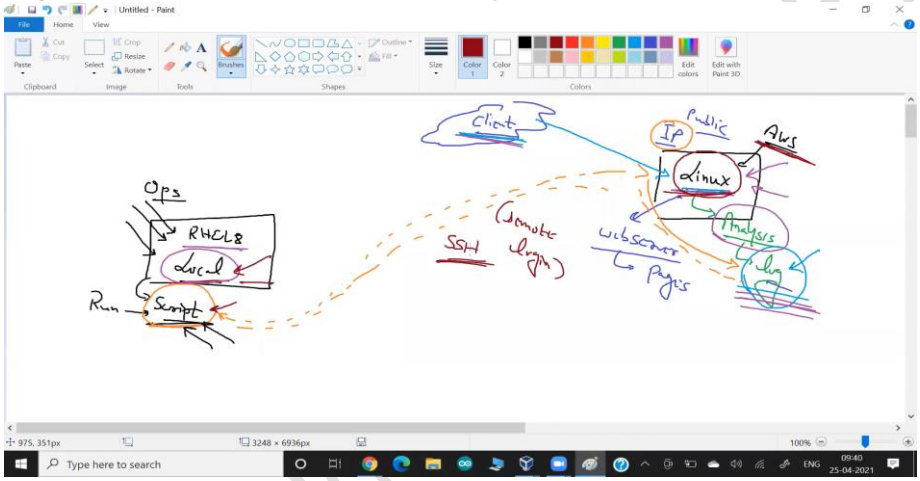
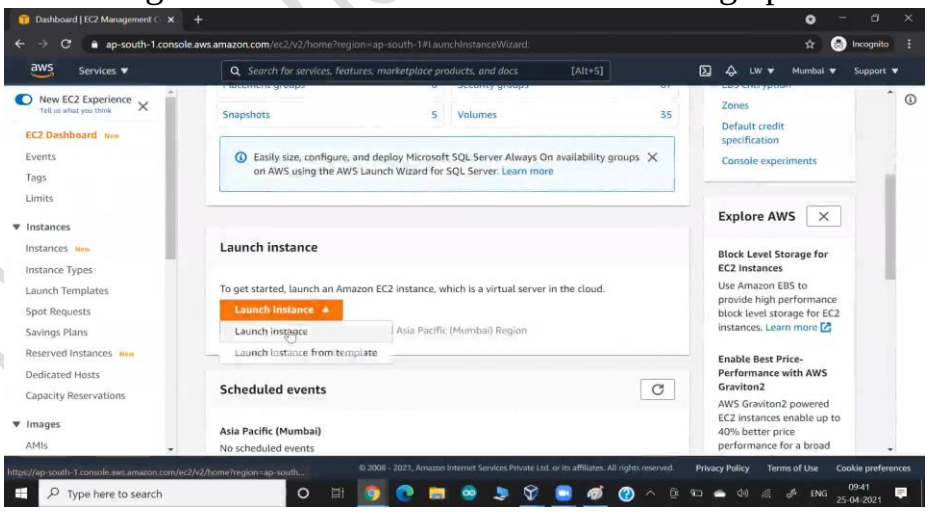
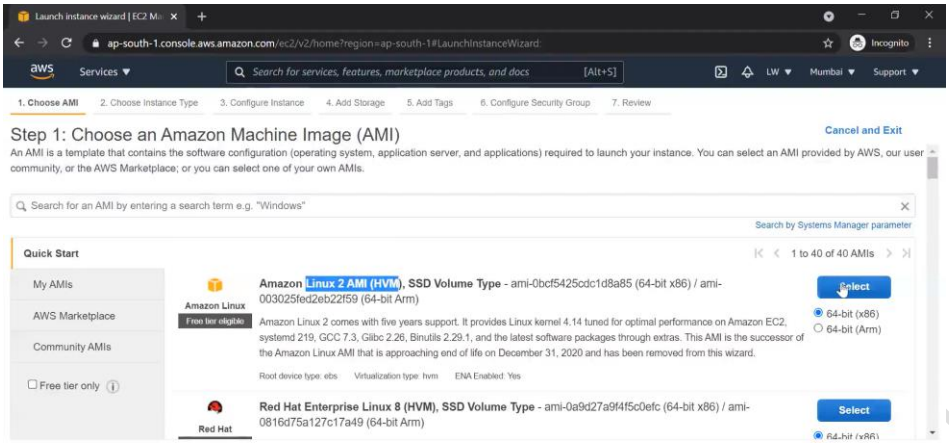
Shell and Shell Scripting Session No.2.1

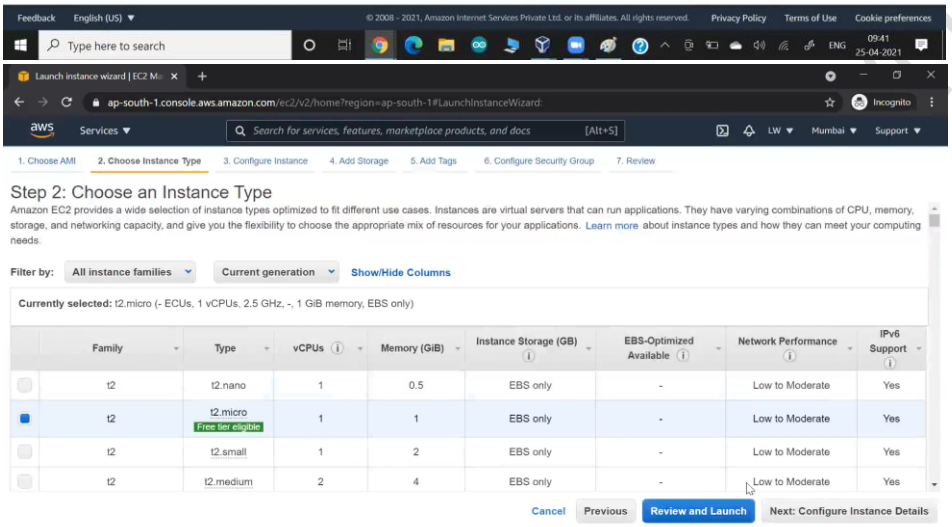
- This session plans to create the script for log analysis of the other server over the network.
- For this we launch one instance or OS on AWS cloud and set up a web server in it. Whenever any client hits the URL, the server will create some logs for this. And we want to analyze those logs. For this, we will create a script on our system that will contact the web server analyze the log, and return the result to our system.

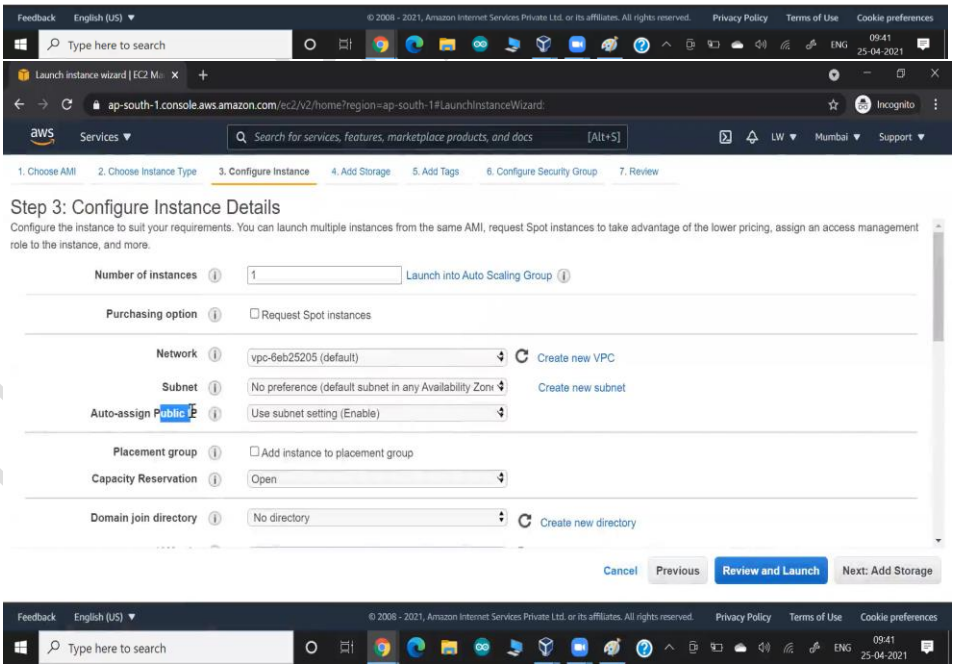


- Launching EC2 instance on the cloud and setting up a web server in it :

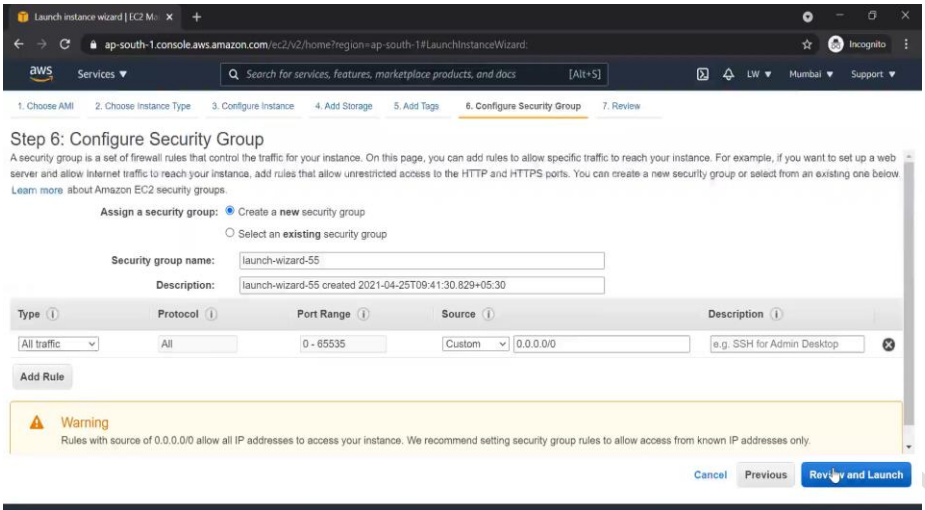


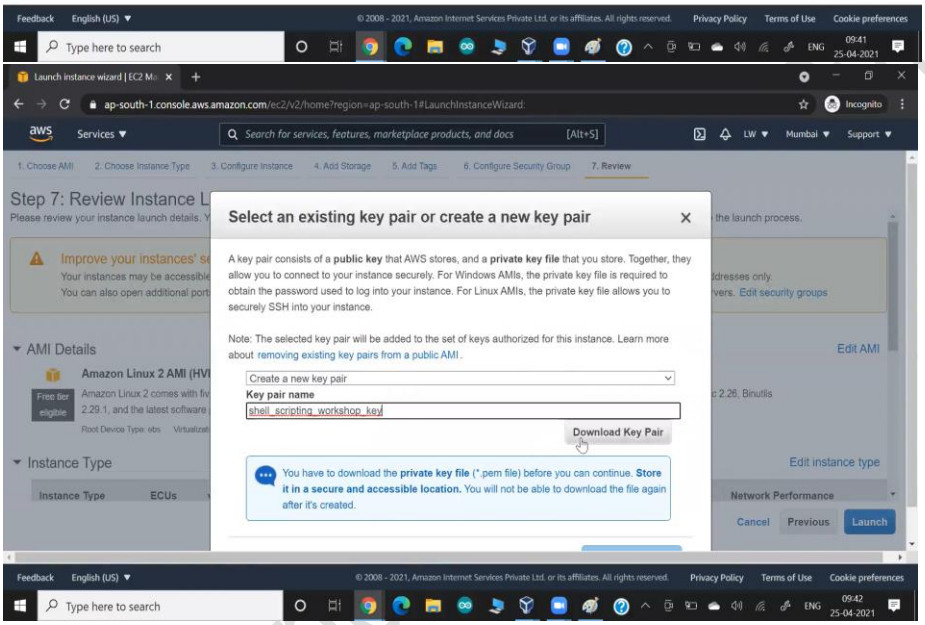

Launch instance wizard | EC2 Ma
ap-south-1.console.aws.amazon.com/ec2/v2/home?region=ap-south-1#LaunchInstanceWizard:
Incognito
Services
Search for services, features, marketplace products, and docs [Alt+S]
LW
Mumbai
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 1: Choose an Amazon Machine Image (AMI)
Cancel and Exit
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.
Search for an AMI by entering a search term e.g. "Windows"
Search by Systems Manager parameter
Quick Start
1 to 40 of 40 AMIs
My AMIs
AWS Marketplace
Community AMIs
Free tier only
Amazon Linux
Free tier eligible
Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0bcf5425cdc1d8a85 (64-bit x86) / ami-003025fed2eb22f59 (64-bit Arm)
Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is approaching end of life on December 31, 2020 and has been removed from this wizard.
Root device type: ebs
Virtualization type: hvm
ENA Enabled: Yes
Select
64-bit (x86)
64-bit (Arm)
Red Hat
Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-0a9d27a9f4f5c0efc (64-bit x86) / ami-0816d75a127c17a49 (64-bit Arm)
Select


Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences
Type here to search
ENG
09:41
25-04-2021
Launch instance wizard | EC2 Ma
ap-south-1.console.aws.amazon.com/ec2/v2/home?region=ap-south-1#LaunchInstanceWizard:
Incognito
Services
Search for services, features, marketplace products, and docs [Alt+S]
LW
Mumbai
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 2: Choose an Instance Type
Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.
Filter by:
All instance families
Current generation
Show/Hide Columns
Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)
Family
Type
vCPUs
Memory (GiB)
Instance Storage (GB)
EBS-Optimized Available
Network Performance
IPv6 Support
t2 t2.nano 1 0.5 EBS only - Low to Moderate Yes
t2 t2.micro Free tier eligible 1 1 EBS only - Low to Moderate Yes
t2 t2.small 1 2 EBS only - Low to Moderate Yes
t2 t2.medium 2 4 EBS only - Low to Moderate Yes
Cancel
Previous
Review and Launch
Next: Configure Instance Details

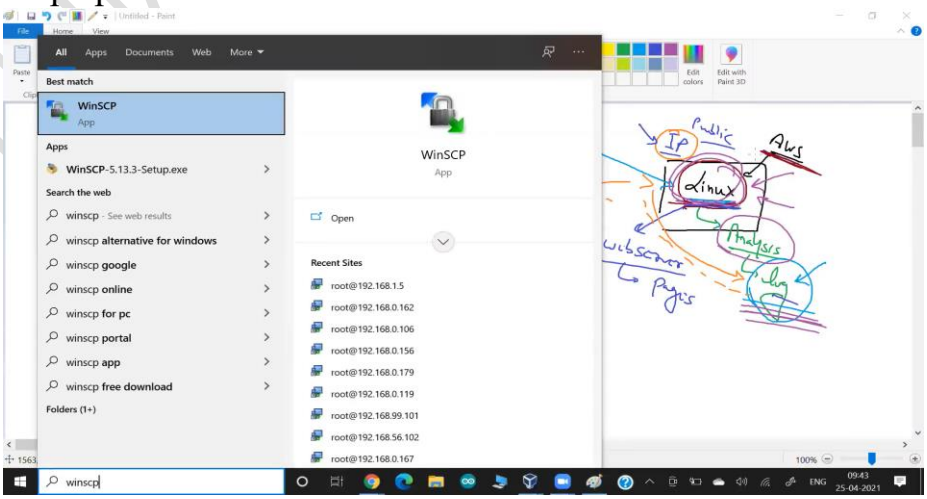

Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences
Type here to search
ENG
09:41
25-04-2021
Launch instance wizard | EC2 Ma
ap-south-1.console.aws.amazon.com/ec2/v2/home?region=ap-south-1#LaunchInstanceWizard:
Incognito
Services
Search for services, features, marketplace products, and docs [Alt+S]
LW
Mumbai
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 3: Configure Instance Details
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.
Number of instances
1
Launch into Auto Scaling Group
Purchasing option
Request Spot instances
Network
vpc-6eb25205 (default)
Create new VPC
Subnet
No preference (default subnet in any Availability Zone
Create new subnet
Auto-assign Public IP
Use subnet setting (Enable)
Placement group
Add instance to placement group
Capacity Reservation
Open
Domain join directory
No directory
Create new directory
Cancel
Previous
Review and Launch
Next: Add Storage

Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences
Type here to search
ENG
09:41
25-04-2021

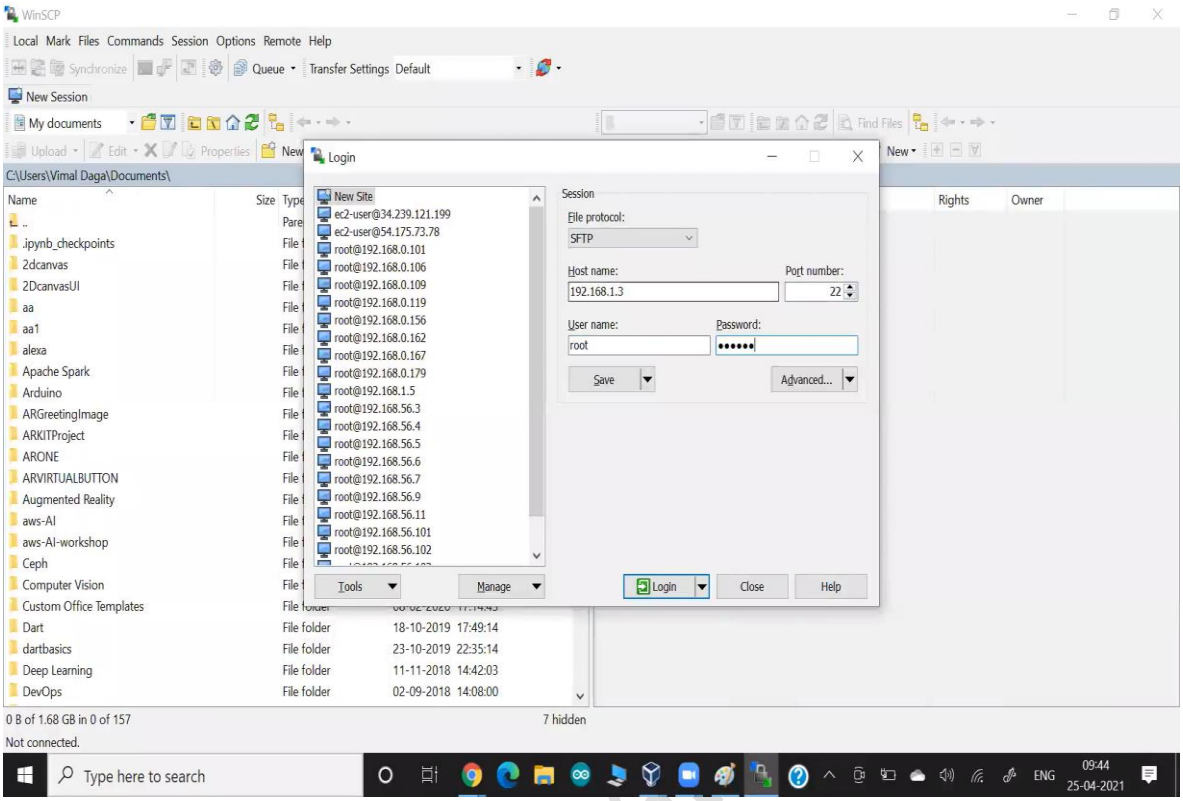


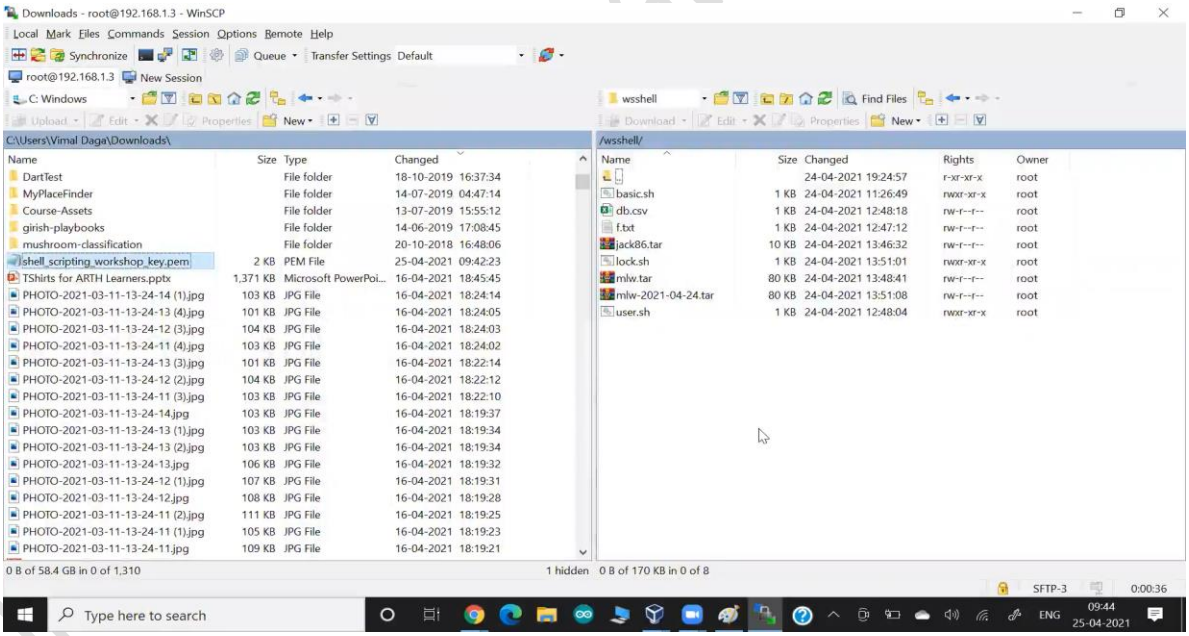


- This key is very important for this practical, as it will help us to do remote login via SSH.
- Now to transfer this file to our RHEL VM8, we will use the tool "WinSCP". It is a tool to transfer files to another system that will use the "scp" protocol.



[Shell And Shell Scripting]





- Now we are going to configure the web server in our Linux OS on AWS.

```
[ec2-user@ip-172-31-32-253 ~]$
[ec2-user@ip-172-31-32-253 ~]$
[ec2-user@ip-172-31-32-253 ~]$ whoami
ec2-user
[ec2-user@ip-172-31-32-253 ~]$ sudo su - root
[root@ip-172-31-32-253 ~]#
[root@ip-172-31-32-253 ~]# yum install  httpd
```

i-09001aed9b01c919f (scriptos1)

Public IPs: 65.1.65.19 Private IPs: 172.31.32.253

```
Installed:
  httpd.x86_64 0:2.4.46-1.amzn2

Dependency Installed:
  apr.x86_64 0:1.6.3-5.amzn2.0.2                apr-util.x86_64 0:1.6.1-5.amzn2.0.2          apr-util-bdb.x86_64 0:1.6.1-5.amzn2.0.2
  generic-logos-httpd.noarch 0:18.0.0-4.amzn2   httpd-filesystem.noarch 0:2.4.46-1.amzn2     httpd-tools.x86_64 0:2.4.46-1.amzn2
  mailcap.noarch 0:2.1.41-2.amzn2               mod_http2.x86_64 0:1.15.14-2.amzn2

Complete!
[root@ip-172-31-32-253 ~]#
[root@ip-172-31-32-253 ~]# cd /var/www/html/
[root@ip-172-31-32-253 html]# ls
[root@ip-172-31-32-253 html]# cat > index.html
home page by LW !1
[root@ip-172-31-32-253 html]# ls
index.html
[root@ip-172-31-32-253 html]# cat > vimal.html
vimal daga
[root@ip-172-31-32-253 html]# ls
index.html  vimal.html
[root@ip-172-31-32-253 html]# systemctl start httpd
[root@ip-172-31-32-253 html]# systemctl enable  httpd
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to /usr/lib/systemd/system/httpd.service.
[root@ip-172-31-32-253 html]#
```

i-09001aed9b01c919f (scriptos1)

Public IPs: 65.1.65.19 Private IPs: 172.31.32.253

- These commands will create web pages and start the web service.
- Below is the file where it will create logs

```
[root@ip-172-31-32-253 html]#
[root@ip-172-31-32-253 html]# cd /var/log/httpd/
[root@ip-172-31-32-253 httpd]# ls
access_log  error_log
[root@ip-172-31-32-253 httpd]# cat access_log
```

```
t/537.36 (KHTML, like Gecko) Chrome/90.0.4430.82 Mobile Safari/537.36"
157.35.247.52 - - [25/Apr/2021:04:18:42 +0000] "GET /rahul HTTP/1.1" 404 196 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36"
59.97.100.47 - - [25/Apr/2021:04:18:43 +0000] "GET / HTTP/1.1" 304 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36"
59.97.100.47 - - [25/Apr/2021:04:18:43 +0000] "GET / HTTP/1.1" 304 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36"
157.35.247.52 - - [25/Apr/2021:04:18:43 +0000] "GET /rahul HTTP/1.1" 404 196 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36"
59.97.100.47 - - [25/Apr/2021:04:18:43 +0000] "GET / HTTP/1.1" 304 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36"
157.35.247.52 - - [25/Apr/2021:04:18:43 +0000] "GET /rahul HTTP/1.1" 404 196 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36"
157.35.247.52 - - [25/Apr/2021:04:18:44 +0000] "GET /rahul HTTP/1.1" 404 196 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36"
125.99.245.194 - - [25/Apr/2021:04:18:45 +0000] "GET / HTTP/1.1" 304 - "-" "Mozilla/5.0 (Linux; Android 11; GM1901) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/87.0.4280.141 Mobile Safari/537.36"
157.35.247.52 - - [25/Apr/2021:04:18:45 +0000] "GET /rahul HTTP/1.1" 404 196 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36"
157.35.247.52 - - [25/Apr/2021:04:18:45 +0000] "GET /rahul HTTP/1.1" 404 196 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36"
125.99.245.194 - - [25/Apr/2021:04:18:46 +0000] "GET / HTTP/1.1" 304 - "-" "Mozilla/5.0 (Linux; Android 11; GM1901) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/87.0.4280.141 Mobile Safari/537.36"
[root@ip-172-31-32-253 httpd]#
```

i-09001aed9b01c919f (scriptos1)

Public IPs: 65.1.65.19 Private IPs: 172.31.32.253

- Now as clients start hitting, it will create more logs.
- To know the total how many hits our site got, we use the “wc” command with the “-l” option. This command will count all the lines in this file.

```
[root@ip-172-31-32-253 httpd]# wc  -l  access_log
14161 access_log
[root@ip-172-31-32-253 httpd]# wc  -l  access_log
18805 access_log
[root@ip-172-31-32-253 httpd]# wc  -l  access_log
20584 access_log
[root@ip-172-31-32-253 httpd]#
```

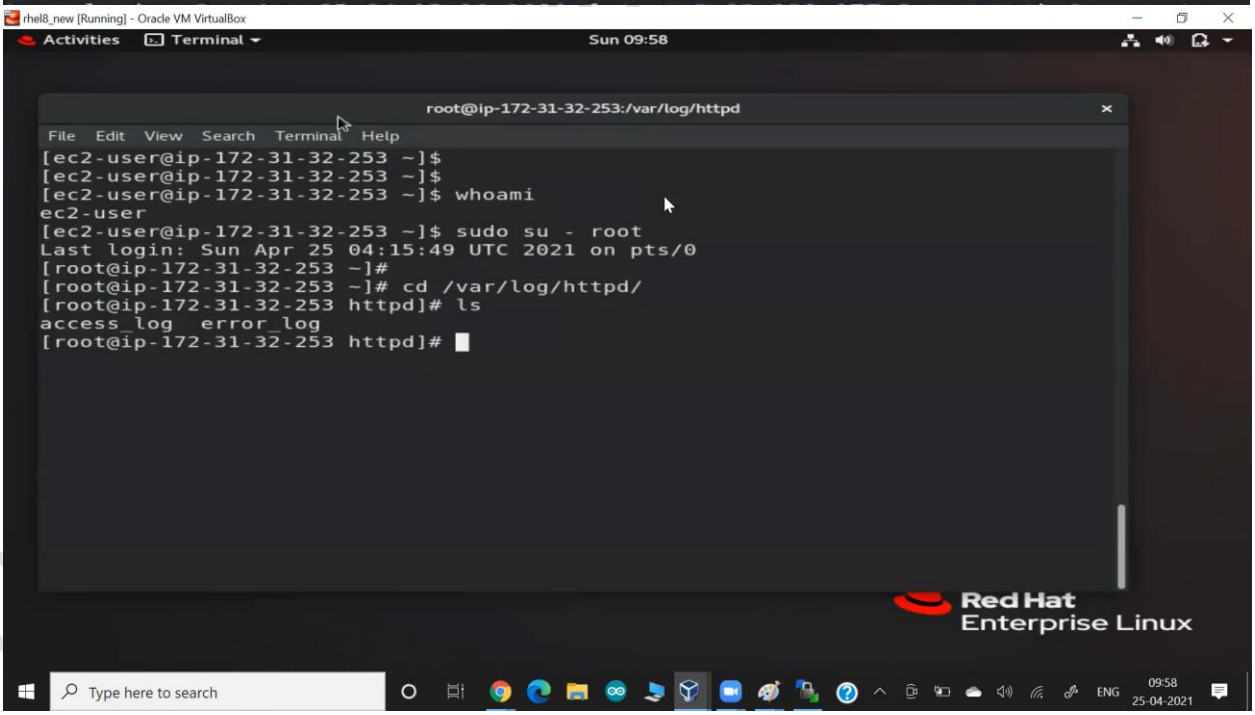
- Each line in this file contains information about the connection like the IP of the client status code or time of connection.
- For using the private key, first we have to change its permissions

```
chmod  o-r  shell_scripting_workshop_key.pem
```

```
chmod  g-r  shell_scripting_workshop_key.pem
```

- Now we will connect to the web server instance via ssh.

```
[root@localhost wsshell]# ssh  -l ec2-user  -i  shell_scripting_workshop_key.pem
      65.1.65.19
```



- Now, we will learn some more basic concepts before creating a script.
- Firstly, we want to receive all the unique IPs from the file and we know that the IP occupies the first field in each line.
- For retreiving, we have the “cut” command but it has limited options therefore we will use another command that is the “awk” command.

- This “awk” command will retrieve entire IPs of all clients from the file

```
File  Edit  View  Search  Terminal  Help
[root@ip-172-31-32-253 httpd]# awk '{ print $1 }'  access_log
125.99.245.194
152.57.110.161
125.99.245.194
223.230.136.186
223.230.136.186
152.57.110.161
96.125.133.36
157.38.48.250
96.125.133.36
157.38.48.250
203.81.243.9
```

- But this list includes some IPs that are repeated many times.
- There is a command in Linux that will sort your data

```
root@ip-172-31-32-253:/var/log/httpd
File  Edit  View  Search  Terminal  Help
[root@ip-172-31-32-253 httpd]# awk '{ print $1 }'  access_log  | sort
```

```
96.125.133.36
96.125.133.36
96.125.133.36
96.125.133.36
96.125.133.36
96.125.133.36
96.125.133.36
96.125.133.36
96.125.133.36
96.125.133.36
96.125.133.36
96.125.133.36
96.125.133.36
96.125.133.36
96.125.133.36
96.125.133.36
96.125.133.36
98.196.110.129
98.196.110.129
98.196.110.129
98.196.110.129
98.196.110.129
98.196.110.129
[root@ip-172-31-
```

- And to get the unique IPs from this entire list, we will use the command “unique”

```
root@ip-172-31-32-253:/var/log/httpd
File  Edit  View  Search  Terminal  Help
[root@ip-172-31-32-253 httpd]# awk '{ print $1 }'  access_log  | sort | uniq
```

```
49.37.175.203
49.37.3.215
49.37.81.75
49.37.83.143
49.37.83.97
49.37.84.22
49.37.86.165
59.91.76.10
59.92.134.200
59.93.94.252
59.97.100.47
60.254.90.81
61.2.248.23
65.2.10.190
66.249.79.110
66.249.79.112
66.249.79.114
83.143.245.162
92.238.202.161
96.125.133.36
98.196.110.129
[root@ip-172-31-
```

- And last we will get the total number of unique clients connected, with this command

```
[root@ip-172-31-32-253 httpd]# awk '{ print $1 }'  access_log  | sort | uniq |
   wc -l
529
[root@ip-172-31-32-253 httpd]#
```

- Now next what we want to do is, get the total number of times each IP hits the server.

```
root@ip-172-31-32-253:/var/log/httpd
File  Edit  View  Search  Terminal  Help
[root@ip-172-31-32-253 httpd]# awk '{ print $1 }'  access_log  | sort  |
   uniq -c
```

```
      3 49.37.169.217
      5 49.37.170.214
      3 49.37.175.203
     18 49.37.3.215
      3 49.37.81.75
      4 49.37.83.143
      7 49.37.83.97
      6 49.37.84.22
      7 49.37.86.165
     11 59.91.76.10
      8 59.92.134.200
      3 59.93.94.252
    586 59.97.100.47
      5 60.254.90.81
      5 61.2.248.23
      9 65.2.10.190
      3 66.249.79.110
      1 66.249.79.112
      2 66.249.79.114
      2 83.143.245.162
      2 92.238.202.161
    176 96.125.133.36
```

- Now we want to see the top 3 IPs among them because it might be possible that someone is trying to hit the server which can lead to some threat

```
[root@ip-172-31-32-253 httpd]# awk '{ print $1 }'  access_log  | sort  |
   uniq -c |  sort  -n
    722 223.190.187.13
    752 106.206.205.2
    898 157.33.13.114
    951 1.39.200.14
   1000 49.36.155.60
   1519 103.195.252.44
   1793 47.8.38.67
   2700 157.33.107.73
   2760 223.181.80.114
   3634 157.34.209.91
   4921 106.208.202.109
   6151 103.16.70.94
   6790 157.34.34.54
   6963 49.34.47.52
  12556 117.200.0.206
  14884 183.83.42.124
  23247 103.109.14.93
  25838 13.232.145.206
```

- This command will sort your data based on the number and based on the data we can do the firewall setting accordingly.
- We know that `date` is the command in Linux and it has various options to customize the output

```
[root@ip-172-31-32-253 httpd]# date +%e/%b/%G
25/Apr/2021
[root@ip-172-31-32-253 httpd]#
```

- And now we require to see how many clients hit today only

```
[root@ip-172-31-32-253 httpd]# grep $(date +%e/%b/%G)  access_log  |  awk '{ pri
nt $1}' | sort | uniq -c | sort -n -k1
```

- This command will give all the IPs that hit the server today
- To make this data more clear and readable, we can do some more things here

```
[root@ip-172-31-32-253 httpd]# cat access_log  | awk '{ print "requests from thi
s ip " $1 }'  | sort | uniq -c | sort -n
```

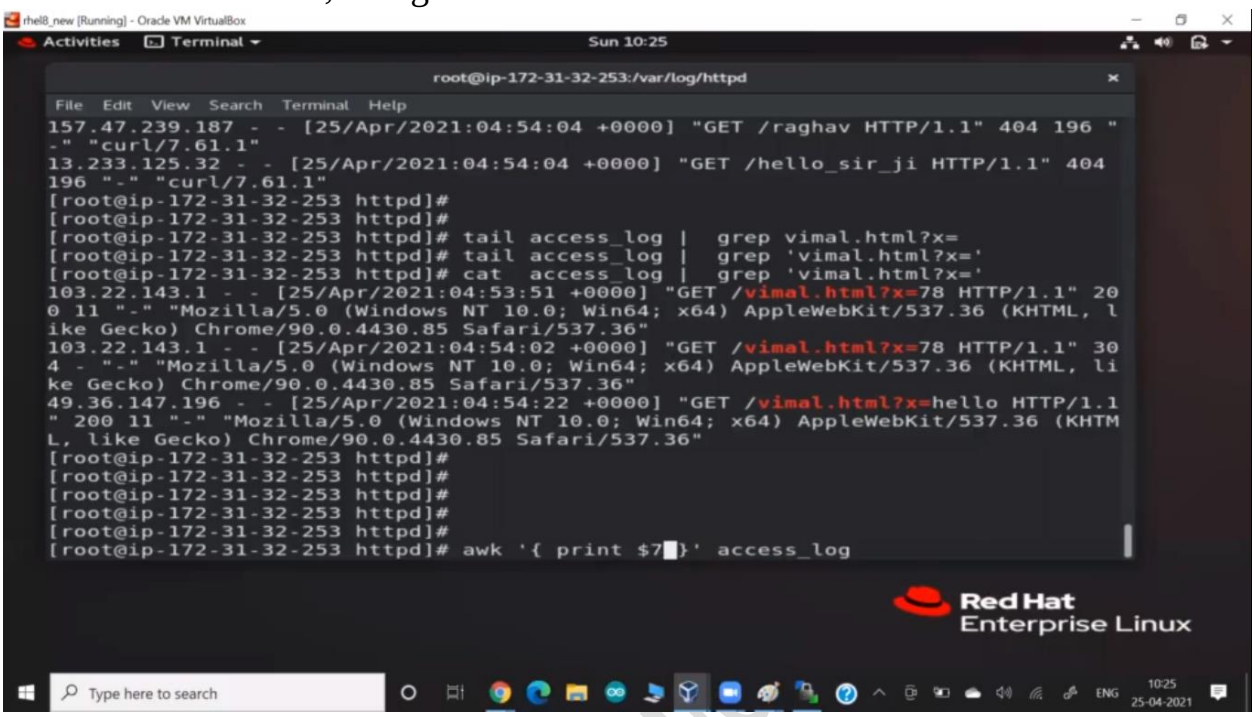
```
    898 bytes requests from this ip 157.33.13.114
    951 bytes requests from this ip 1.39.200.14
   1000 bytes requests from this ip 49.36.155.60
   1265 bytes requests from this ip 157.47.239.187
   1338 bytes requests from this ip 157.38.88.39
   1625 bytes requests from this ip 103.195.252.44
   1744 bytes requests from this ip 106.206.205.2
   2041 bytes requests from this ip 47.8.38.67
   3037 bytes requests from this ip 157.41.59.56
   3634 bytes requests from this ip 157.34.209.91
   4442 bytes requests from this ip 125.99.245.194
   4940 bytes requests from this ip 223.181.80.114
   6963 bytes requests from this ip 49.34.47.52
   9727 bytes requests from this ip 157.33.107.73
  12556 bytes requests from this ip 117.200.0.206
  13174 bytes requests from this ip 106.208.202.109
  14884 bytes requests from this ip 183.83.42.124
  18149 bytes requests from this ip 223.190.187.13
  19798 bytes requests from this ip 103.16.70.94
  21117 bytes requests from this ip 157.34.34.54
  25838 bytes requests from this ip 13.232.145.206
  30118 bytes requests from this ip 103.109.14.93
  82656 bytes requests from this ip 13.233.125.32
[root@ip-172-31-32-253 httpd]#
```

- "tail" is the command in Linux that will give you the last ten lines of a file

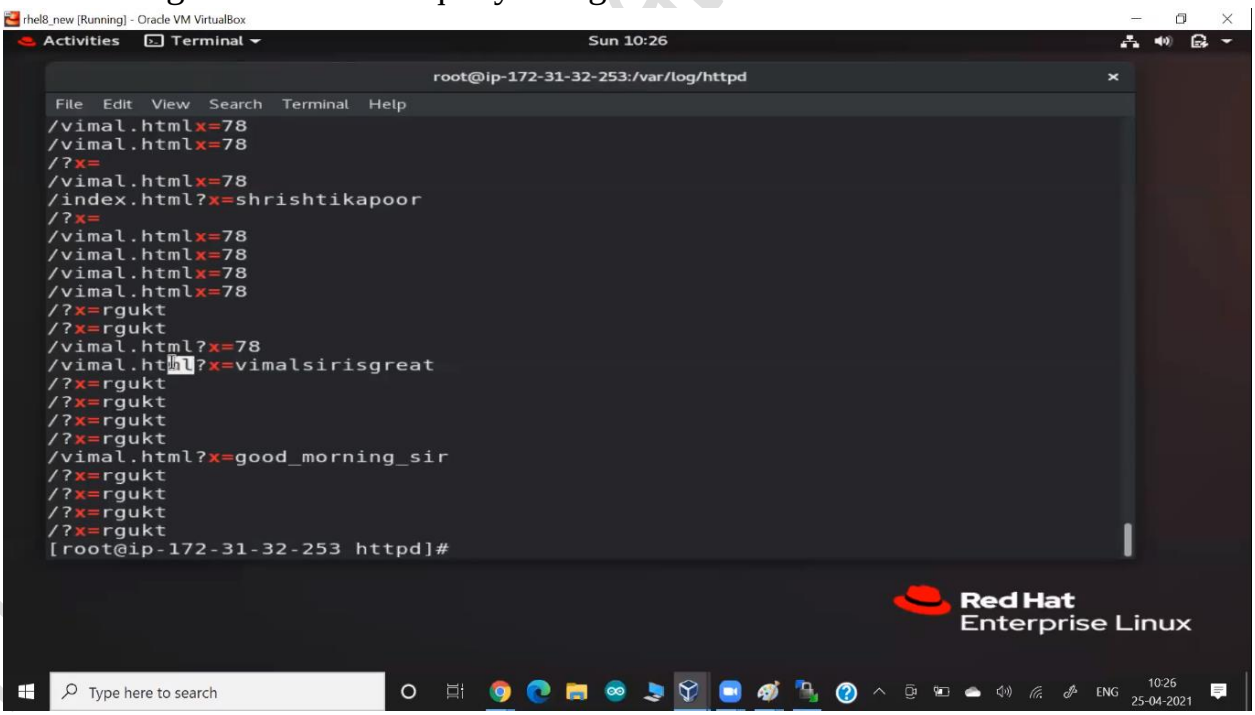
```
[root@ip-172-31-32-253 httpd]# tail access_log
13.233.125.32 - - [25/Apr/2021:04:53:11 +0000] "GET / HTTP/1.1" 200 19 "-" "curl
/7.61.1"
13.233.125.32 - - [25/Apr/2021:04:53:11 +0000] "GET / HTTP/1.1" 200 19 "-" "curl
/7.61.1"
106.208.202.109 - - [25/Apr/2021:04:53:11 +0000] "GET /pawan_karan HTTP/1.1" 404
 196 "-" "curl/7.71.1"
106.208.202.109 - - [25/Apr/2021:04:53:11 +0000] "GET /pawan_karan HTTP/1.1" 404
 196 "-" "curl/7.71.1"
13.233.125.32 - - [25/Apr/2021:04:53:11 +0000] "GET / HTTP/1.1" 200 19 "-" "curl
/7.61.1"
157.34.34.54 - - [25/Apr/2021:04:53:11 +0000] "GET /bobbySingh.html HTTP/1.1" 40
4 196 "-" "curl/7.55.1"
13.233.125.32 - - [25/Apr/2021:04:53:11 +0000] "GET / HTTP/1.1" 200 19 "-" "curl
/7.61.1"
13.233.125.32 - - [25/Apr/2021:04:53:11 +0000] "GET / HTTP/1.1" 200 19 "-" "curl
/7.61.1"
::1 - - [25/Apr/2021:04:53:11 +0000] "OPTIONS * HTTP/1.0" 200 - "-" "Apache/2.4.
46 () (internal dummy connection)"
157.34.34.54 - - [25/Apr/2021:04:53:11 +0000] "GET /bobbySingh.html HTTP/1.1" 40
4 196 "-" "curl/7.55.1"
```

- Now I want to get all the web pages that were searched by the clients. And we know that it is the 7th field in the file.

- And below command, will get this 7th field



- But it also gets us the entire query string



- I want to get only the web pages, not the query string, hence we will use the below command for this purpose

```
[root@ip-172-31-32-253 httpd]# awk '{ print $7 }' access_log  | cut -d? -f1   |
 sort | uniq -c  | sort -n -k1
   2105 /appick_manager
   2109 /store/subscriptions
   2119 /store
   2126 /store/category
   2128 /store/partners
   2362 /great_vimal_daga
   2363 /raghav
   3634 /abhi.html
   3750 /manzoor.html
   4299 /IIEC-Rise
   4745 /love_u_sir
   6767 /mranali
  10000 /:)
  11591 /hello_sir_ji
  27625 /itzsrv
  28225 /vimal.html
  31221 /U_have_been_looped
  34715 /pawan_karan
  44411 /bobbySingh.html
 168570 /
```

- But among all these web pages, many web pages are wrong means it doesn't even exist hence the server returns status code 404. So, I want only those web pages that have status code 404.
- This command will print all the lines that have status code 404.

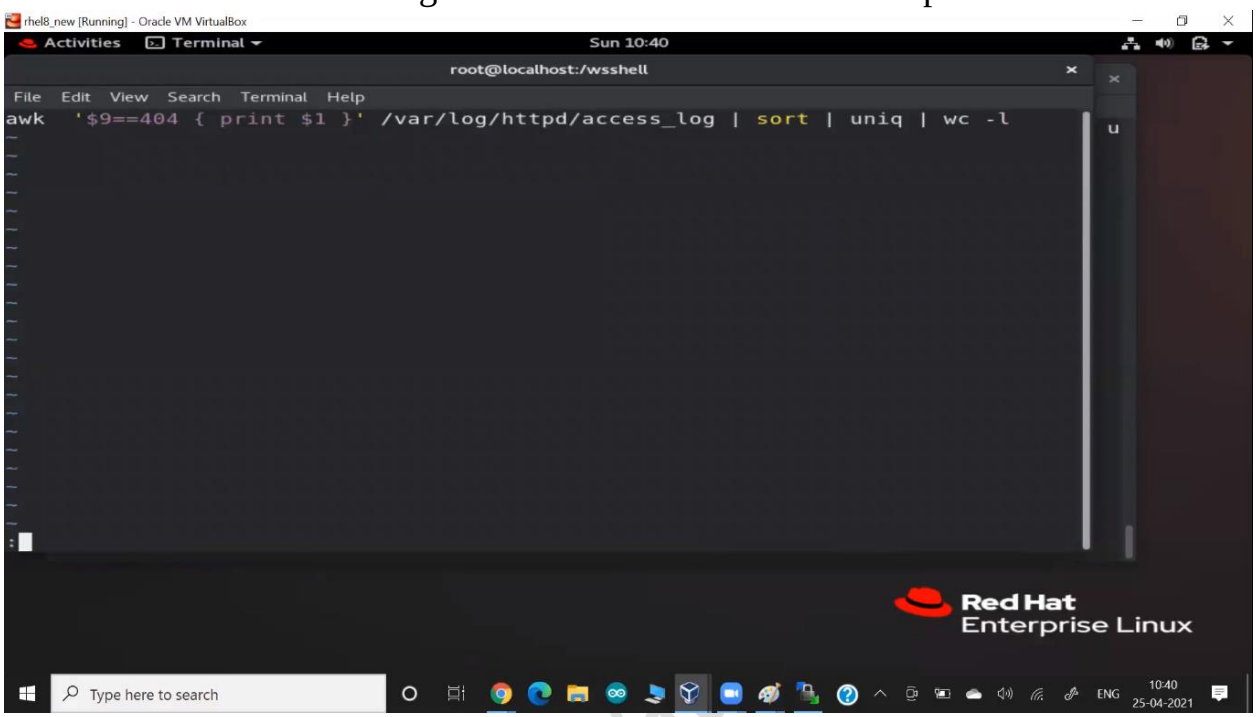
```
[root@ip-172-31-32-253 httpd]#
[root@ip-172-31-32-253 httpd]# awk  '$9==404 { print $0 }' access_log
```

```
root@ip-172-31-32-253:/var/log/httpd
File  Edit  View  Search  Terminal  Help
196 "http://65.1.65.19/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36"
152.57.67.77 - - [25/Apr/2021:04:19:32 +0000] "GET /favicon.ico HTTP/1.1" 404 19
6 "http://65.1.65.19/" "Mozilla/5.0 (Linux; Android 9; ASUS_X01BDA) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/87.0.4280.101 Mobile Safari/537.36"
103.22.143.1 - - [25/Apr/2021:04:19:32 +0000] "GET /lw.html HTTP/1.1" 404 196 "-
" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gec
ko) Chrome/90.0.4430.85 Safari/537.36"
103.240.163.50 - - [25/Apr/2021:04:19:33 +0000] "GET /shreeja HTTP/1.1" 404 196
"-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
ecko) Chrome/89.0.4389.128 Safari/537.36"
157.33.107.73 - - [25/Apr/2021:04:19:33 +0000] "GET /favicon.ico HTTP/1.1" 404 1
96 "http://65.1.65.19/" "Mozilla/5.0 (Linux; Android 10; M2007J20CI) AppleWebKit
/537.36 (KHTML, like Gecko) Chrome/90.0.4430.66 Mobile Safari/537.36"
106.223.177.79 - - [25/Apr/2021:04:19:33 +0000] "GET /Sarthak HTTP/1.1" 404 196
"-" "Mozilla/5.0 (Linux; Android 9; Redmi Y2) AppleWebKit/537.36 (KHTML, like Ge
cko) Chrome/89.0.4389.105 Mobile Safari/537.36"
103.240.163.50 - - [25/Apr/2021:04:19:33 +0000] "GET /favicon.ico HTTP/1.1" 404
196 "http://65.1.65.19/shreeja" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) Apple
WebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.128 Safari/537.36"
157.38.92.188 - - [25/Apr/2021:04:19:33 +0000] "GET /favicon.ico HTTP/1.1" 404 1
96 "http://65.1.65.19/" "Mozilla/5.0 (Linux; Android 11; SM-M315F) AppleWebKit/5
37.36 (KHTML, like Gecko) Chrome/89.0.4389.105 Mobile Safari/537.36"
```

- Now I want to run the below command which will give me the total number of clients that get the status code 404.

```
[root@ip-172-31-32-253 httpd]# awk  '$9==404 { print $1 }' access_log | sort | u
niq | wc -l
536
```

- But we are required to run this command over the network on the server where our website is running. So first we have to create a script for this.



- But if I run this script here, it will get the data from our local system. But our requirements are different. So, for this first, we have to copy this script on the target system, and then only we can run it over there.
- If you are trying to run any command that needs root power, you cannot run it from any other user unless it has a sudo power. In the AWS, there is a default user that has a sudo power means root power and that user is ec2-user.
- And for running any privilege command we need to use sudo before that command.

```
[root@localhost wsshell]# ssh  -l ec2-user  -i  shell_scripting_workshop_key.pem
      65.1.65.19
Last login: Sun Apr 25 05:11:33 2021 from 103.22.143.1

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
1 package(s) needed for security, out of 15 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-32-253 ~]$ whoami
ec2-user
[ec2-user@ip-172-31-32-253 ~]$ cat /etc/shadow
cat: /etc/shadow: Permission denied
[ec2-user@ip-172-31-32-253 ~]$
[ec2-user@ip-172-31-32-253 ~]$ sudo cat /etc/shadow
```

- And now, I want to run my script over the network on the server where the web server is running. So, we first transfer this file there. For transfer from one system to another we have the “scp” command.

```
[root@localhost wsshell]#
[root@localhost wsshell]# scp     -i  shell_scripting_workshop_key.pem        we
b.sh   ec2-user@65.1.65.19:/tmp
web.sh                                        100%   77     1.0KB/s   00:00
```

This command will transfer the script from our local system to the target system in/tmp folder.

- And then we can easily run this script there with the “ssh” command.

```
[root@localhost wsshell]# ssh  -l ec2-user  -i  shell_scripting_workshop_key.pem
      65.1.65.19      sudo  bash /tmp/web.sh
547
[root@localhost wsshell]#
```

- In Linux, if you want to run two commands together in one go then there is a symbol, “&&”, we have to use between them.

```
[root@localhost wsshell]# date &&  cal
Sun Apr 25 10:49:06 IST 2021
      April 2021
Su Mo Tu We Th Fr Sa
                1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

- But there is a condition in this command, if the first command runs successfully then only the second will run otherwise both will fail.

```
[root@localhost wsshell]# date1 &&  cal
bash: date1: command not found...
Similar command is: 'date'
[root@localhost wsshell]#
```

- To solve this, we have another option which is to use “; ” between them. This will remove this problem and run both commands, no matter which one fails.

```
[root@localhost wsshell]# date1; cal
bash: date1: command not found...
Similar command is: 'date'
      April 2021
Su Mo Tu We Th Fr Sa
                1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

- Now to run two commands with “ssh”, we have to use this syntax

```
[root@localhost wsshell]#
[root@localhost wsshell]# ssh  -l ec2-user  -i  shell_scripting_workshop_key.pem
      65.1.65.19      "date;  hostname"
Sun Apr 25 05:22:17 UTC 2021
ip-172-31-32-253.ap-south-1.compute.internal
[root@localhost wsshell]#
```

- Always keep both commands in double quotes otherwise ssh will take the first command only and the next command will run in the local system.

- There is a command in Linux that translates your file data into your desired way. Like the below command will translate all small letters into a capital letter

```
[root@localhost wsshell]# cat my.txt
this is vimal from lw
hello hi
this is lw
lw vimal
hi
[root@localhost wsshell]# cat my.txt  | tr 'a-z'  'A-Z'
THIS IS VIMAL FROM LW
HELLO HI
THIS IS LW
LW VIMAL
HI
```

- The below command will give a new line to each word that is divided by a space and do further sorting on it.

```
[root@localhost wsshell]# cat my.txt  | tr -s  ' '  '\n'   |  sort  | uniq -c |
 sort -n
      1 from
      1 hello
      1 is
      1 is
      2 hi
      2 this
      2 vimal
      3 lw
```

- As we know date is one command in Linux and cal is one command in Linux, but internally we know that they are functions of programs that we can run because it has their PATH set. If we remove the path then we won't be able to run any of these commands.

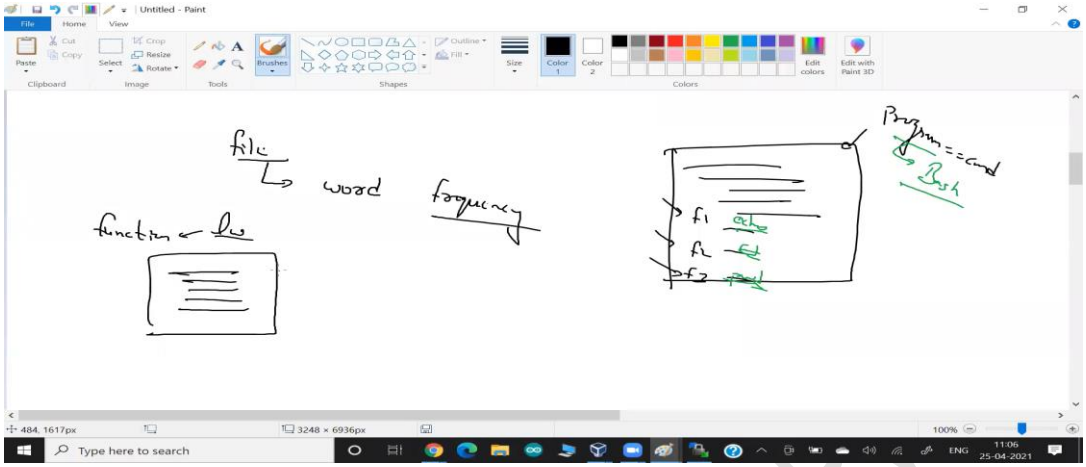
```
[root@localhost ~]# echo $PATH
/wsshell:/maven3/bin/:/usr/share/Modules/bin:/usr/local/bin:/usr/local/sbin:/usr
/bin:/usr/sbin
[root@localhost ~]# PATH=
bash: sed: No such file or directory
[root@localhost ~]# echo $PATH

bash: sed: No such file or directory
```

```
[root@localhost ~]# which date
/usr/bin/which: no date in ()
bash: sed: No such file or directory
[root@localhost ~]# date
bash: date: No such file or directory
bash: sed: No such file or directory
[root@localhost ~]# cal
bash: cal: No such file or directory
bash: sed: No such file or directory
```

- Now, we can able to run only bash program functions. Because this program is still running. Type "e" and then press the tab key two times. Below are all those commands that are included in the bash program.

```
[root@localhost etc]# d
declare  dequote  dirs     disown   do       done
[root@localhost etc]# d
declare  dequote  dirs     disown   do       done
[root@localhost etc]# d
declare  dequote  dirs     disown   do       done
[root@localhost etc]# e
echo    elif    enable  eval    exit
egrep   else    esac    exec    export
[root@localhost etc]# e
```

- What is the function? → whenever you want to run certain lines again and again, you put those statements in a box give it a name, and then call it with that name only



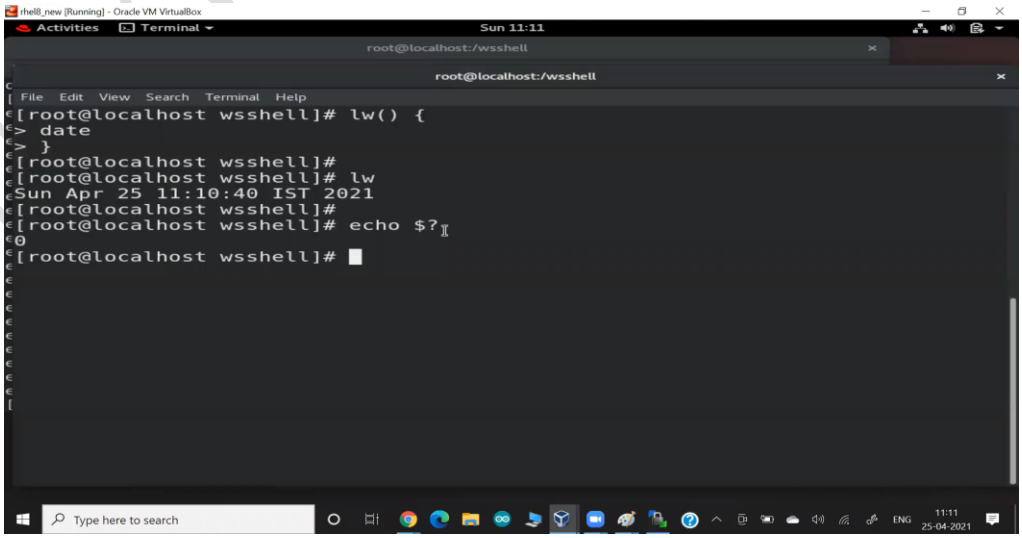
- Syntax of creating function in shell :

```
[root@localhost wsshell]# lw() {
> echo "hi"
> }
[root@localhost wsshell]#
```

```
File  Edit  View  Search  Terminal  Help
lw() {
        echo "hi $1 .."
}

lw vimal
```

```
[root@localhost wsshell]# bash f.sh
hi vimal ..
[root@localhost wsshell]#
```

- Now if we see the exit code of the function



- But function has no exit code, command or program has the exit code but Shell is treating it as a command that's why it giving us the status code. So the next point is "how to control the exit code of the function".
- For this we have a keyword "return" that we use in the function. It will decide the exit code for this function.

```
[root@localhost wsshell]# lw() { date; return 11; }
[root@localhost wsshell]# lw
Sun Apr 25 11:12:00 IST 2021
[root@localhost wsshell]# echo $?
11
```

- Now to remove the function that we created on the shell, we use the "unset" command.

```
[root@localhost wsshell]# unset lw
[root@localhost wsshell]# lw
bash: lw: command not found...
Failed to search for file: Cannot update read-only repo
[root@localhost wsshell]#
```