**Guide to Installing and Securing a Linux Server**

This guide is split into 6 parts. Each part addresses 1 of the 6 challenges:

Pt1: Installation & securing

Pt2: Appendix A - Users & Groups (Joe B)

Pt3: AppendixB - Automating Users & Groups (Joe P)

Pt4: AppendixC - Bash One-Liner (Adam)

Pt5: AppendixD -Command-Line Productivity (Brad)

Pt6: AppendixE - Bash Scripting Challenge (Conor)

**Installing the OS, configuring a static address, allowing remote access through SSH, securing the server.**

1. Download and install VMWare workstation Player @ https://www.vmware.com/go/getworkstation-win
2. Download RH Enterprise version needed (x86\_64 download version) at [developers.redhat.com/products/rhel/download](https://developers.redhat.com/products/rhel/download)
3. Once installed VMWare right click and run as administrator.
4. Create a new virtual machine and select the installed disc image file for OS, click next.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. Next assign the guest OS as Linux and version Red Hat Enterprise 8 64-bit (Or equivalent), click next.

Graphical user interface

Description automatically generated

1. Assign a virtual machine name and assign a file location to store the VM, next.
2. Assign your needed storage for the VM, next.
3. Next you can assign the various hardware for your VM, do so as needed for your machine. We dedicated around 4GB of memory and 8 – 12 cores for processing.

A screenshot of a computer

Description automatically generated with medium confidence

1. Finish and run your VM. Booting will take some time, let it run.
2. Once booted it will ask you to enter a root password which will be used throughout the su system and then select a storage location. Click done and reboot.
3. Start setup and follow the steps till completion.

Diagram

Description automatically generated

**Make user sudoer:**

* change to root: “**su –**“
* change to etc directory: **cd /etc/**
* enter sudoers file: **sudo visudo**
* Enter insert Mode with i. Locate the below lines and under root copy the same as below for created user.
* ESC then enter **:wq** to write and exit the sudoers file.
* Return to created user with: **su** [user]

Text

Description automatically generated

**Configure Static IP Address:**

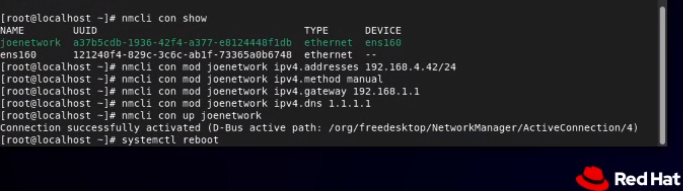
* **cd ~**
* **use “ifconfig”** command to find the network device running. As a group we all have ens160. This is used in the next commands. If you have another use that.

Text

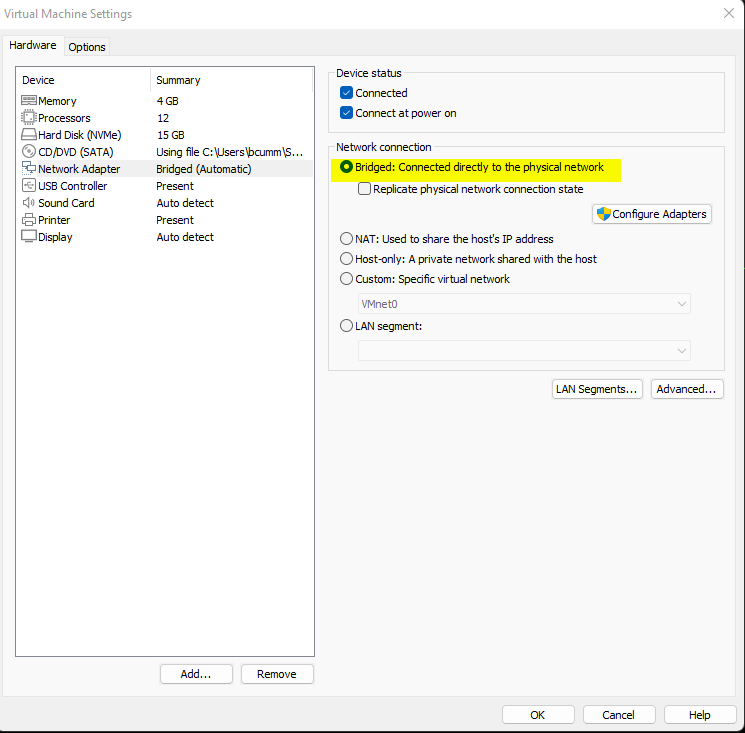
Description automatically generated

To setup the static address connection use the NetworkManager with nmcli. There will be no message provided after entering the commands correctly, only if entered wrong.

* **nmcli connection add con-name [connection\_name] ifname [network\_device] type ethernet**
* **nmcli connection modify [connection\_name] ipv4.addresses [ip\_address/24]**
* **nmcli connection modify [connection\_name] ipv4.gateway [gateway\_address]**
* **nmcli connection modify [connection\_name] ipv4.dns [1.1.1.1]**
* **nmcli connection up [connection\_name]**



In vm settings change Network Type as to bridged.



Verify newly created static ip address with:

* **nmcli device status** – This will display the new connection created and if its connected.
* **ping [ip\_address]** – If correctly configured you will successfully ping the ip address.

Graphical user interface, text

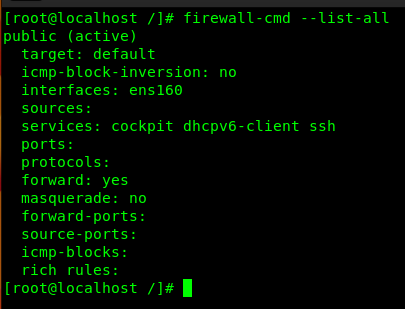
Description automatically generated

**Setting up and configuring firewall:**

* **systemctl unmask firewalld**
* **systemctl start firewalld**
* **systemctl enable firewalld**

Use the following commands to check that your firewall is correctly running.

* **firewall-cmd --check-config**
* **firewall-cmd --state**



**‘firewall-cmd –list-all' -** use this to see what services are running.

**‘firewall-cmd --zone=public --permanent --remove-service service’ -** use this to remove any unwanted services that are allowed.

**‘firewall-cmd --zone=public --permanent --remove-port port\_number’ - use this to remove any unwanted ports**  
**‘firewall-cmd –reload' - reload the firewall to apply changes.**

Add the port for SSHing into the vms.

**firewall-cmd --zone=public --permanent --add-service=ssh**

**firewall-cmd --zone=public --permanent --add-port 1500/tcp**

**Setting up SSH with user.**

* Generate ssh key with: **ssh-keygen**
  + enter password – this should be different than the user’s original password. If it is the same (and knows the original), an attacker will be able to access the private key and other systems which that key is linked.
* Copy the generated public key to the server and associate with user:
* **ssh-copy-id user@ip**
  + **yes**
* **eval $(ssh-agent)**
  + **ssh-add .ssh/[key\_location]**

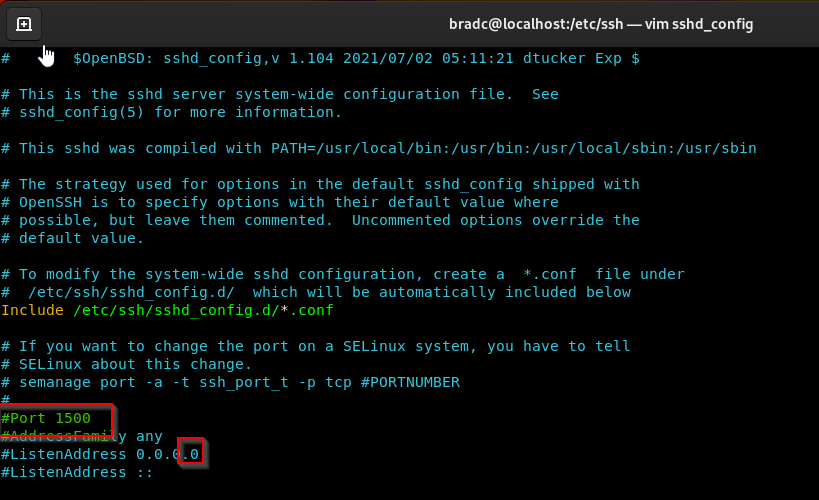
Now user should be able to ssh in with public key

**Securing the server:**

Disable direct root login over ssh and password logins for remote users over SSH.

* **Su –**
* **cd /etc/ssh/**
* **vim sshd\_config**

Locate port and change to the port number assigned to the firewall, previously. Port 1500 in this case.



* Find #PasswordAuthentication line – enter insert mode (i), change to no.
* Find #PermitRootLogin line – enter insert mode (i), change to no.
* Find #PubkeyAuthentication, this should be set to yes.

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

* + this means that only key authentication is enabled on the OpenSSH server as have disabled password and root logins
* Write and Exit with :wq
* **Systemctl reload sshd**

1. Open **/etc/ssh/sshd\_config**
2. PermitEmptyPasswords no
3. PermitRootLogin no

**APPENDIX A: Managing Users and Groups**

Create a directory at the root (/) of the file system for each department. This name should reflect the company (team) and department name that will use the directory.

* **cd /**
* **mkdir Team5\_Finance Team5\_HR Team5\_IT**

Create a group for each department. This name should reflect the company and department name that the group will be assigned.

* **groupadd -g 2000 Team5\_Finance\_Group**
* **groupadd -g 2001 Team5\_HR\_Group**
* **groupadd -g 2002 Team5\_IT\_Group**

Create an administrative user for each department. The user will have the Bash login shell /bin/bash, and primary group of their department.

* **useradd –g 2000 –s /bin/bash finance-admin**
* **useradd –g 2001 –s /bin/bash hr-admin**
* **useradd –g 2002 –s /bin/bash it-admin**

Create an additional two users for each department. The users will have the Bash login shell /bin/bash. The users will belong to their respective group for each department. This should be the user’s supplementary group.

* **useradd -aG 2000 Brad**
* **useradd -aG 2000 Adam**
* **useradd -aG 2001 Joe**
* **useradd -aG 2001 Jo**
* **useradd -aG 2002 Conor**
* **useradd -aG 2002 Oscar**

Ensure that the owner of each of the directories is the department administrator and the group ownership is the group for each department.

* **chown finance-****admin:Team5\_Finance\_Group Team5\_Finance**
* **chown it-****admin:Team5\_IT\_Group Team5\_IT**
* **chown hr-****admin:Team5\_HR\_Group Team5\_HR**

The department administrator will have full access to their respective department directories.

Normal (group member) users in each department will have full access to their respective department folders.

The department folders will ONLY be accessible by users/administrators in each of the respective departments. Ensure that no one else will have permissions to the folders.

* **chmod 770 Team5\_IT**
* **chmod 770 Team5\_Finance**
* **chmod 770 Team5\_HR**

Ensure that only the owner of a file in the department’s directory can delete the file.

* **chmod +t Team5\_Finance**
* **chmod +t Team5\_IT**
* **chmod +t Team5\_HR**

Files created in the directory must have the group owner automatically matching the group owner of the directory.

* **chmod g+s Team5\_Finance**
* **chmod g+s Team5\_IT**
* **chmod g+s Team5\_HR**

Create a document with your team name, e.g. Team4.txt, in each of the department directories. The ownership on this file will be the same as the directory it is located in. The document should contain a list of the names of the team members. This file can be read by any user in the department but can only be modified by the department administrator. No one else has permissions to this file.

* **su finance-admin**
* **cd /Team5\_Finance**
* **vim team5.txt (And add team names)**
* **sudo chmod 740 team5.txt**
* **su it-admin**
* **cd /Team5\_IT**
* **vim team5.txt (And add team names)**
* **sudo chmod 740 team5.txt**
* **su hr-admin**
* **cd /Team5\_HR**
* **vim team5.txt (And add team names)**
* **sudo chmod 740 team5.txt**

**Deliverables:**

Use an appropriate command to verify each user and group has been created.

* **Less /etc/group**
* **Less /etc/passwd**

Use an appropriate command to verify each user’s group assignment

* **Id [username]**

Use an appropriate command to verify the directory creation and the permission settings.

* **cd /**
* **ls –ld**

Use an appropriate command to verify the files are created in their respective directories.

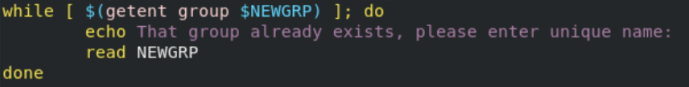
* **ls –lr Team5\_[dept]**

## **APPENDIX B: Automating New Users and Groups**

First, you need to create a new file to create your script in. You can do this by using the touch command followed by the name of the file you want created. You can then use the VIM command followed by the name of the file to edit the file and start to code your script. Once inside the file use I to edit the file.



Your script will start by prompting the user to enter a name for the group, then it will assign the name the user entered to the variable called NEWGRP.



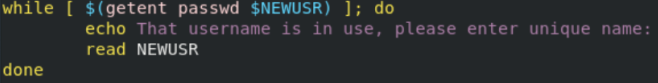
You will then have to make sure that the group name is not already in use, this is done by using the getent command to check the group file and if it is prompt the user to change the name until it can't be found. The script will check the group file.



The groupadd command will then create a group with the name that was assigned to the variable NEWGRP user.



The script will then prompt the user for a username. Then it will assign the name given by the user to the variable NEWUSR.



After this you will use a similar loop to check that the username is not already being used. The script will check the ‘passwd’ file by using the getent command and the variable is there to show that its checking for the input from the user.



The script will then generate the user with the new group as its primary this is done by having –g in the command, using the shell as bash, its home directory will be in /home and the user will expire on the 31st December 2022.



After the user is added there will be a prompt for the current user to set a password for the newly created user. It will then be assigned to the variable NEWUSR.



There will then be a new group directory at root created and owned by the new user and any other users that will be added to this group and only these users will have permissions to the directory.



This line of code shows the permissions for the NEWGRP. The 1 is for the sticky bit and shows that users can only delete files in the directory that they created. The 770 shows that the new user and group have permission to read, write and execute and that anyone not part of the group will have no permissions at all.



Finally echo to the user that the group directory has been created at root and that’s the script finished. You can now stop editing the file by clicking escape and then :wq to write and exit the file and get yourself back to the command line. From here you can then execute your script and it should create a new group with your entered specific details.

**APPENDIX C: Bash One Liner**

Using a combination of pipes, redirects and control statements, produce an output that contains only the service names. The entire task must be accomplished without using any intermediary files. Each service should only be listed once and captured to a file named ‘uniqueservices.txt’. Remove any blank lines or lines that are deemed to be comments.

Objectives:

* Extract all the service names from the file.
* Sort the names alphabetically removing any duplicates.
* Remove any blank lines or lines that do not begin with a letter.
* Capture the final output to a file named uniqueservices.txt.
* Count the lines in the file using a conditional command that is only executed if the previous combined commands are successful.

Full command used:

**awk '{print $1}' /etc/services | sort -u | grep . | sed -n '/^[****[:alpha:]]/p' > /home/uniqueservices.txt && wc -l /home/uniqueservices.txt**

The following is a breakdown of the command, with a brief explanation of each part of the pipeline:

**awk '{print $1}' /etc/services |**

The awk command reads the file line by line, (print $1) outputs only the first column of each line (Service names are at the start of their lines) Piped ( | ) regularly to next command. The pipe allows the next command to use the previous commands output as it’s input.

**sort -u |**

Sort –u will sort alphabetically only unique entries. Piped regularly to next command.

**grep . |**

With the grep command (.) matches to any character, so only lines with characters will be output to the next part of the pipeline, removing blanks. Piped regularly to next command.

**sed -n '/^[[:alpha:]]/p' > /home/uniqueservices.txt &&**

With the sed command, –n effectively discards everything not output by the next part ‘/^[[:alpha:]]/p’ which prints (p) only lines starting with (^) a letter ([[:alpha:]]), and outputs to file (> uniqueservices.txt). Piped using the conditional (&&), which causes the next command only to execute if the previous was successful.

**wc -l /home/uniqueservices.txt**

With ‘wc –l', which counts the number of lines.

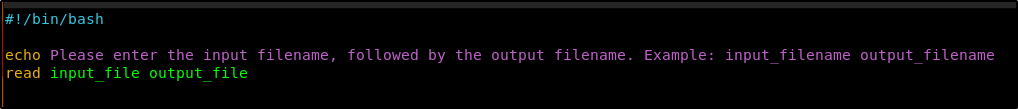
**Appendix D: Command Line Productivity**

Enter: **‘touch scriptname’** inside the directory you want to store script.

**Sudo chmod +x scriptname** to give it the correct execution rights.

Open scriptname with vim editor, as discussed above.

The following images are the code required for the script in order and there will be an explanation below:



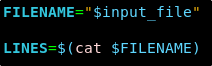
**#!/bin/bash** is known as a shebang. This tells the shell that the file being run is a bash script and is required in some OS distributions.

Echo line will display a message asking to input the input and output filenames.

Read takes those inputs and stores them in the variables input\_file output\_file.

Inserting image...

Echo outputs RHNID, EMAIL, FIRSTNAME, LASTNAME to the output\_file. To access a variable within the script you put a $ infront of the variable name.



Next, setting up for the loop later, which makes use of a row counter.

**FILENAME=”$input\_file”** simply stores the input\_file value into FILENAME variable.

**LINES=$(cat $FILENAME)** stores the FILENAME value into LINES.



Initialising the variable counter to 1.



Conditional loop is setup, with for.

The command ‘LINE’ counts the number of rows in $LINE, this is used for the number loops occurences.

‘do’ tells the shell what to do next.

Following commands are indented within the loop. The loop takes one row at a time and processes the required registration details storing them in their respective variables: RHNID, EMAIL, FIRSTNAME, SURNAME. 

The initial part of the command up until the first, |, will be the same for each command.

Awk is a scripting language used within bash. Here it is used to locate field required in each variable.

**-F**, specifies the delimiter where each column starts and ends.

**-v**, is a variable option for awk. This case ‘i’ is set to counter variable.

‘**FNR == I {print $4}**’, this where it prints the value at field of FNR (the row number) and $4 (column number). In the following commands, column numbers are changed but the command stays the same until the |

**Cut –d “@” -f1**: cuts everything before @.

-f1 is used to specify which side of the @ should be cut.

Tr [:upper:] [:lower:], simply takes the output and changes it to lowercase.



This line is the same as above, and it just stores the output of the first awk command and stores it in the EMAIL variable.

This takes the output of the awk command and cuts the characters before the “ to be stored in the variable firstname.



This takes the output of the awk command and cuts the characters after (as it is –f2) the “ to be stored in the variable surname.



This takes the variables stored in the loop and using stream editor it removes any spaces after commas and appends to a new row output\_file.



Counter variable is incremented by 1 and the loop is repeated until the counter = Lines in file.



Done tells the shell the script is complete.

Exit vim with: ‘**:wq**’ and then to run the script.

In terminal enter, ‘**sudo ./scriptname**’

**Appendix E: Bash Scripting Challenge**

Write a bash script that converts a phrase such as 'Rolling on the floor laughing' to ROTFL.

All other punctuation should be removed, including hyphens (treat hyphens as word separators) e.g. Taylor-Jendernai-Eirron de Mountstuart would be TJEDM.

Make sure git is installed:

**sudo yum install git**

Install Bash Automated Testing System to test our script functions using these commands:

**Git clone https://github.com/bats-core/bats-core**

**Cd**

**Sudo ./install.sh /usr/local**

There are 2 files provided (bats-extra.bash & capitalise.bats) in order to test the script. These MUST be in the same directory as our script file.

For ease, create a directory at root (e.g /appendixE).

Place the 2 provided files in the directory.

Then create your script which MUST BE CALLED capitalise.sh, as the capitalise.bats test file is configured to call upon a script with that name.

With vim, place this code within the script:

Text

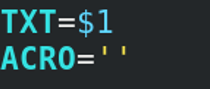
Description automatically generated

As described in part D.



Next, we’re establishing variables. First, we’re taking a value passed to the script in the same line as it’s called by the test file. The first value passed into a script in the same line that calls it called $1. So, we’re putting that value into a new variable called TXT.

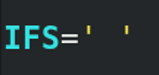
We also declare a variable called ACRO, where our acronym will go.



Hyphens are to be considered word separators, so we’re transforming all hyphens into spaces. This line takes TXT, but replaces ALL instances (//) of ‘-’ with (/) ‘ ‘. (A space)



This line sets the value of the shell variable IFS (Internal field separator). This variable essentially specifies what character’s to be used as a word separator, specifically with the read command, which we soon use.



Now, we’re using read. With –a, it will read our input into an array.

TXTARR is the name of the array we’re creating, and our input is TXT.

<<< acts to expand the contents of TXT into multiple strings (separated by space, as specified by IFS) before being read into TXTARR one at a time, so each part will be stored separately.



Now the sentence has been split into parts, we want only the first letter. So, we’re running a ‘for’ loop, that will run once for each value stored within TXTARR.

As each value of the array is called, it will be stored in variable VAL. The @ sign goes where you would normally put the index of a specific value within TXTARR, and lets each individual value be called one at a time.

‘do’ finishes the loop condition line.

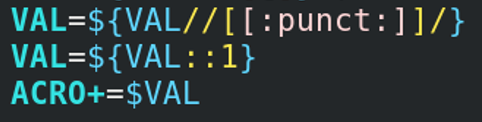


Next are the commands that act on each part of our sentence.

First we use the same syntax as when we turned all hyphens into spaces. This time turning all instances of punctuation into nothing. (Because after (/) there is no character, punctuation will be removed).

Next we’re getting the first letter. Here it will keep characters starting from the number between the colons (nothing = first), and the number after the colons specifies how many characters to keep. So starting at the first, and keeping 1, we have the first letter of this word.

Finally we append (+=) our letter to the ACRO variable.



This closes our loop.



With the first letter of each word stored in ACRO, the last thing to do is convert all letters to uppercase (^^) and output its contents (echo).

