UPES

Xebia

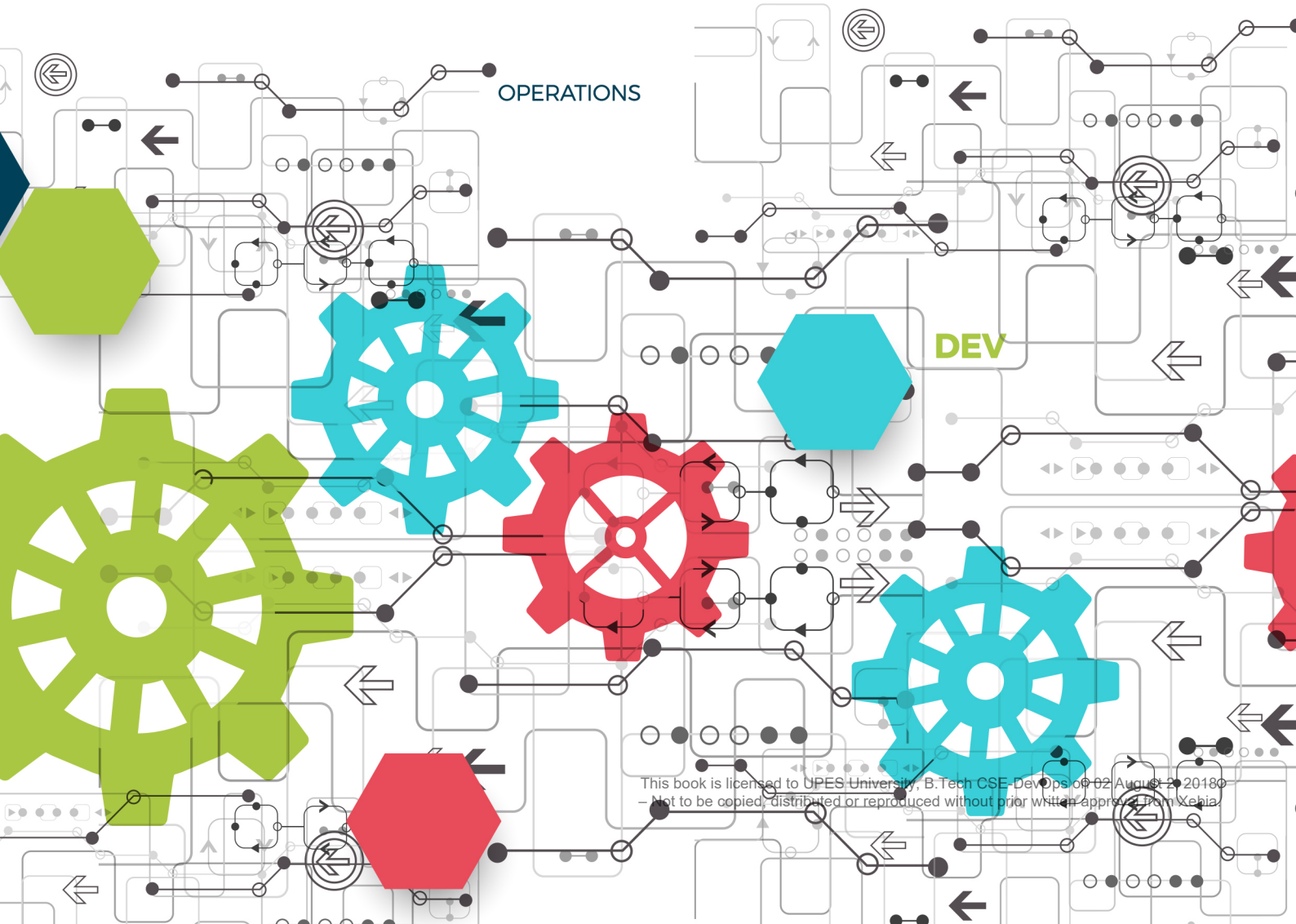
2 YEAR

B.Tech Computer Science and Engineering in DevOps

DEVOPS AUTOMATION

Semester 03 | Facilitator Handbook

Release 1.0.0



Copyright & Disclaimer

B. TECH CSE with Specialization in DevOps

Version 1.0.0

Acknowledgements

We would like to sincerely thank the experts who have contributed to and shaped B. TECH CSE with Specialization in DevOps. Version 1.0.0

SME

Rajagopalan Varadan
A tech enthusiast who loves learning and working with cutting-edge technologies like DevOps, Big Data, Data science, Machine Learning, AWS & Open stack

Course Reviewers.

Aditya Kalia | Xebia
Maneet Kaur | Xebia
Sandeep Singh Rawat | Xebia
Abhishek Srivastava | Xebia
Rohit Sharma | Xebia

Review Board Members.

Anand Sahay | Xebia

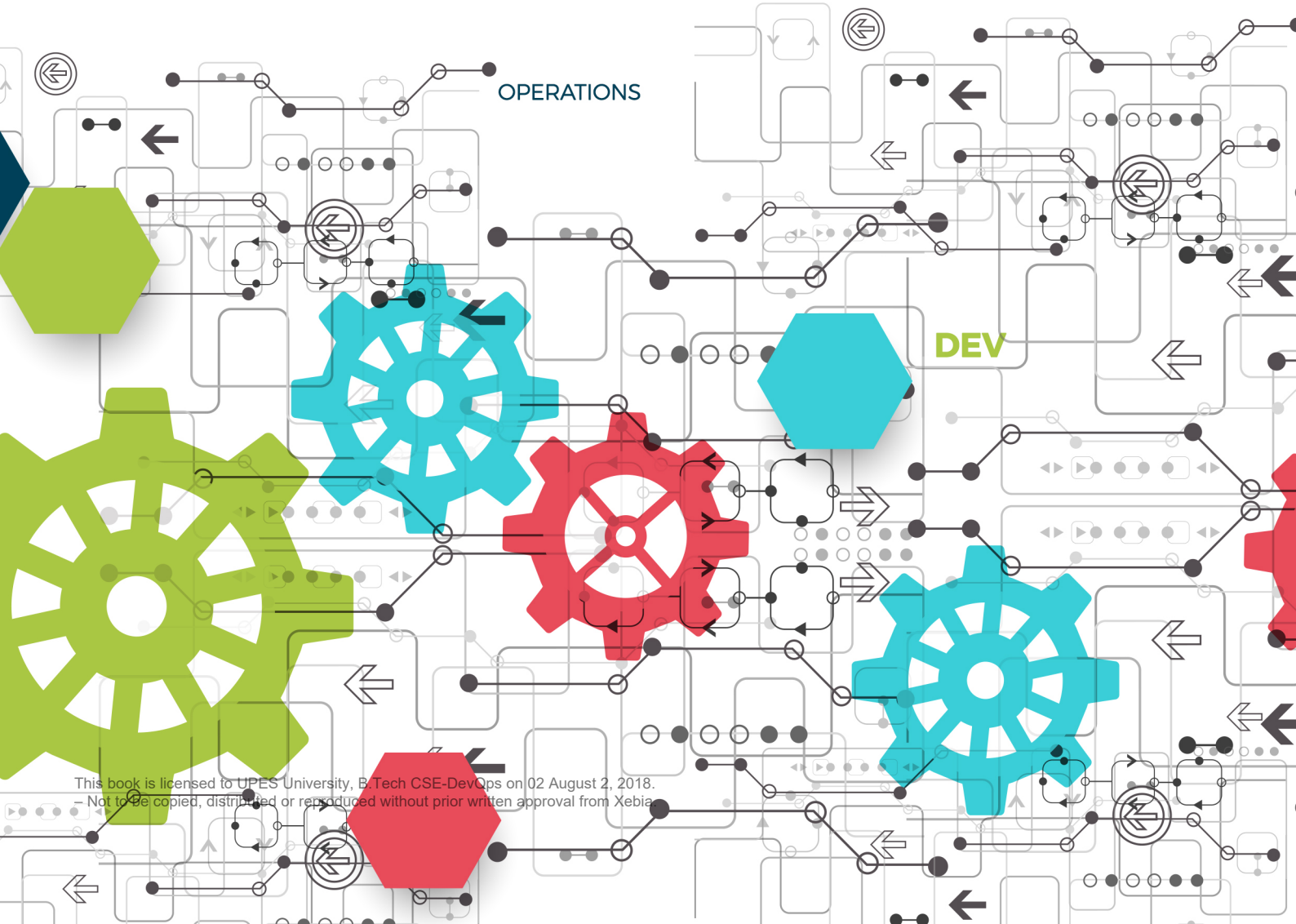
Xebia Group consists of seven specialized, interlinked companies: Xebia, Xebia Academy, XebiaLabs, StackState, GoDataDriven, Xpirit and Binx.io. With offices in Amsterdam and Hilversum (Netherlands), Paris, Delhi, Bangalore and Boston, we employ over 700 people worldwide. Our solutions address digital strategy; agile transformations; DevOps and continuous delivery; big data and data science; cloud infrastructures; agile software development; quality and test automation; and agile software security.

ODW is dedicated to provide innovative and creative solutions that contribute in growth of emerging technologies. As a learning experience provider, ODW strengths include providing unique, up to date content by combining industry best practices with leading edge technology. ODW delivers high quality solutions and services which focus on digital learning transformation.

B.Tech Computer Science and Engineering in DevOps

DEVOPS AUTOMATION

MODULE 2
Advantages of Automation



Contents

MODULE 2

Advantages of Automation

You will learn about the ‘Development Automation’ in this module.

Module Learning Objectives

At the end of the Module you would be able to learn the following

- Identify the types of routine tasks that can be automated.
- Predict where errors can occur and how to prevent them using automation.
- Find the right approach to create a program to automate tasks.

You will be informed about the module objectives.

After completing this module, you will be able understand various ways to automate different kind of day-to-day tasks. You will also find out a solid approach to automating the scenarios.

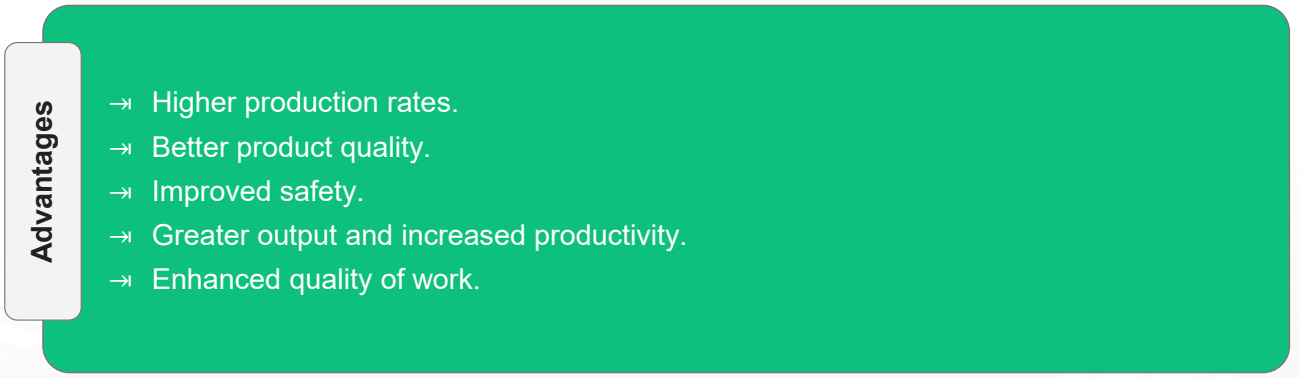
Module Topics

Let us take a quick look at the topics we will cover in this module:

1. Advantages of Automation.
2. Scenarios where automation saves Time and Effort.
3. Scenarios where Automation Prevents Errors.

1. Advantages of Automation

Advantages of Automation are:



Think and recollect how the technologies have evolved over the time. Ask yourself how Industrial Automation helped mankind to grow very quickly and be more productive. Point out the new technologies and match them with the areas where they reduced burden. List down the primary factors of growth in industries and know how these factors have been improvised by automation.

A typical computer can process more quickly than human thinking. Hence the time consumption to complete the work is much less resulting in a significant cost reduction. Following are some major advantages of Automation:

- Automation is mainly attributed to result in higher production rates, better product quality, improved safety and increased output.
- Greater output and increased productivity are achieved by reducing the tedious works given to the IT administrators. Instead, those works are automated by using finely tested programs that can be completely relied upon.
- Unlike humans, the program written is capable of doing the actions specified for any period of time continuously whereas humans are susceptible to distraction, automation will provide enhanced quality of work.

2. Automation Scenarios

The Automation Scenarios are classified into two broad categories:

Scenarios that save time and effort:

In some situations, the same set of work should be done on multiple servers or multiple times on the same server. This category will walk through scenarios of this kind.

Scenarios that prevent errors:

Think of searching for a list of words in hundreds of files having thousands of lines. One cannot completely rely on a human to do this without any errors whereas a program is absolutely capable of doing this, within seconds with hundred percent reliability.

Gather various requirements in IT operations and list those requirements one by one.

Imagine and try to figure out the steps to automate each requirement.

Identify the most significant benefit in automating every one of them and place them under the appropriate category.

Scenario 1 – Archiving Logs

How do I archive all the matching logs and move the archived files to the backup directory?

Scenario

For archiving the log and to backup them:
- **Read Input:** Directory under which all log files are located. ***Example:*** /var/log
- **Read Input:** Extension of log files. Example: log
- **Read Input:** Location of the backup directory
- **Run:** Use archiving executables such as 7z or tar to create archives of the logs matching the given input.
- Move the created archive files to the backup directory.
- Delete the old log files.

The following Scenario explains how to archive the logs.

- For archiving the logs and to backup them, you first need to know where the log files are located and the extension of the log files. Get this as an input from the user.

- To safeguard them to a backup directory, get the location of the backup directory from the user.
- In order to make archives, a lot of compression tools and algorithms are available. Instead of reinventing the wheel, let us use some existing tools that are available in most of the Linux Operating System. Some of them are tar, 7zip, unzip, WinRAR, etc. These binaries will help us turn our fat log files to slim archives that will take much lesser space.
- Finally, once the archives are created, move them to our backup location and deleted the unnecessary log files.
- If we need log files back, then we can extract from the archive files we have.

Scenario

Archive all the matching logs and move the archived files to the backup directory.

Pseudo Code

⇥ **Read Input:** Directory under which all log files are located.
Example: /var/log
⇥ **Read Input:** Extension of the log files. Example: .log
⇥ **Read Input:** Location of the backup directory
⇥ **Run:** Use archiving executables such as 7z or tar to create archives of the logs matching the given input.
⇥ Move the created archive files to the backup directory.
⇥ Delete the old log files.

The following Scenario explains how to archive the logs.

- For archiving the logs and to backup them, you first need to know where the log files are located and the extension of the log files. Get this as an input from the user.
- To safeguard them to a backup directory, get the location of the backup directory from the user.
- In order to make archives, a lot of compression tools and algorithms are available. Instead of reinventing the wheel, let us use some existing tools that are available in most of the Linux Operating System. Some of them are tar, 7zip, unzip, WinRAR, etc. These binaries will help us turn our fat log files to slim archives that will take much lesser space.
- Finally, once the archives are created, move them to our backup location and deleted the unnecessary log files.
- If we need log files back, then we can extract from the archive files we have.

Scenario 2 – Auto-Discard Old Archives

How to Delete archive files that are older than two days automatically?

Scenario

To delete old archives:
- ⇥ **Read Argument:** Location of the archived files.
- ⇥ Get the timestamp of all the archived files in the given location.
- ⇥ **Check:** The timestamp of the file is older than two days
  True: Delete the file
- ⇥ **Check:** The script is scheduled in the cron
  False: Add the script to cron such that it gets executed everyday at 12 AM.

Most of the enterprises will have a log retention policy. Especially, some healthcare industries and banking sectors should retain the log for a period of minimum 5 years. The logs generated on a single day can be Gigabytes or even Terabytes in size. So soon the companies would run out of space to store these logs. In order to make them more efficient, the logs older than the retention period must be deleted so that the space will be optimally utilized. So we need a person here who just deletes the log of terabytes in size and thousands in number every day. What if he deletes the wrong files and forgets to delete the files that should be deleted. There are more possibilities of these faults unless they are automated.

Let's create a program that will automatically find the archived log files that are older than a specified time period. The program should collect the list of older log archives and should delete them all.

- The program needs to know the location of the directory containing the archives. And it needs the date & time beyond which the files will be considered as old and marked for deletion.
- Then let the program get all the files in the given directory and filter the files that are older than the given timestamp. Once it has the list, it can delete all the files in the list.
- To continue the same process everyday, just add it to the task scheduler named 'cron' in Linux that would just execute the program for you exactly at the time you specify it.
- Thus it saves the cost, effort and time of human resources and provides a very high reliability.

Scenario 3 – MySQL (RDBMS) Backups

Scenario

How do I take MySQL Backups every 12 hours and keep it safely on the backup directory?

Pseudo Code

- **Read Arguments:** Location of the backup directory and the MySQL Credentials file (Credentials file should have a hostname, port, username and password of the database server).
- **Check:** The Location of the MySQL Credentials file is valid. If true, read the credentials as local variables, Else throw an error saying credentials not found and write to the log.
- **Run:** MySQL Dump command to retrieve all the data from MySQL. If it returns a non-zero exit code, then exit the program and write the error to the log.
- Redirect the stdout returned above to a new file in the backup directory. This file can be used to restore the database.
- **Check:** The script is scheduled in the cron. If not, Add the script to cron such that it gets executed every 12 hours.

Generally, companies having tons and tons of data will store all of them in databases, such as MySQL, MS SQL, PostgreSQL or OracleDB. Some of them will use NoSQL DBs like MongoDB, Cassandra, etc. These database management systems store the data efficiently within them and will be of great help to retrieve, write and query on the data it has. The applications will fetch the data from the DBs and will properly format it to be human-friendly.

Since the data resides in a single place, it can be a single point of failure. If the infrastructure fails, the data is lost forever. Think of a hard disk failure you have then, how can you retrieve it back and how quickly you can repair it. In reality, it is completely intolerable from enterprises perspective. Millions of customers will be lost if the site is down for an hour. To make this system more secure, it is always better to have redundant data that will quickly help us to make everything immediately work.

- To dump the data of MySQL, you need the location of the server in the form of IP, the username and the password to connect to it. You may also need the particular database to backup from the server. Let us read these configuration settings from a file so that our program will be simple to debug and switch environments.
- Read the location of the configuration file and also get the directory in which the dumped backup data should be kept.
- As we need our program to be more robust, verify the configuration file and the backup directory.
- Then using the backup utility for MySQL named ‘mysqldump’ get all the data from DB and write the output to the file and move the file to the backup location.
- Remember that retrieving such amount of data will affect the DB performance and this should be done during non-peak hours.
- So schedule this on the cron to do the same for the interval at which you need to backup the DB.

Scenario 4 – Email Web Server Summary

Scenario

- How do I email the summary of the web server requests every day?
- Should the summary have the response counts of all the HTTP status codes?

Pseudo Code

- **Read Argument:** Location of Web Server's log files.
- Filter the logs that are generated within the last 24 hours.
- Iterate through the lines in the log and have a counter to track the count of each response code.
- Finally, print the above summary and pipe it to the mail function so that summary can be emailed. If mail function returns a non-zero exit code, then exit the script and write the error message to the logs.
- **Check:** The script is scheduled in the cron. If not, Add the script to cron such that it gets executed every 24 hours.

A web server is nothing but a software that continuously listens for the request from a user through the client called as browsers. Some commonly used browsers are Chrome, Firefox, Safari, Opera, Internet Explorer, etc., and some of the Web servers you might know are Apache, Nginx, Tomcat, etc. A status code is just a three digit integer that is present in the response provided by the web server for the request made by the client. These status codes indicate whether the request is processed without errors, with client-side errors or with server-side errors.

- 1xx: Informational
- 2xx: Success
- 3xx: Redirection
- 4xx: Client Error
- 5xx: Server Error

An IT administrator may often need to check the web server for these kinds of errors and report the summary to their executives for planning the development requirements. Again this is a work that can be automated. A web server usually writes the access logs to the file and those logs will have information such as the status code, client IP, HTTP Method, Path, etc.

- The program to automate needs the location of this log file and hence pass it through the arguments.
- Then it should fetch the logs that are newer than 24 hours and filter the status codes from the logs.
- Once it gets the status codes, it just needs to aggregate their counts and write to a file.
- By using the mail function available in the Linux ‘mailutils’ package, we can read the summary from the file and send it as an email to any desired email IDs.
- To make the program run every day, just schedule it in the cron and it takes care of the rest.

Scenario 5 – Ensure Web Server is Running

Scenario

Should I continuously monitor and Restart the web server if it is not running? How?

Pseudo Code

- **Read Argument**: Web server's listening port.
- **Run:** 'netstat' command to find all the listening ports of the server.
- Filter the above list with the web server's list port.
- If the filtered list is empty, then write the timestamp to log and restart the web server.
- **Check:** The script is scheduled in the cron.
  False: Add the script to cron such that it gets executed every minute.

A web server is meant to be running all the time s so that it can listen for connections made by the browsers and respond back with appropriate content. It can be down due to a variety of reasons. Some of them include, but not limited to, out of memory exception, too many requests in the queue, etc. It is very necessary to ensure that the web server is running all the time and it is almost impossible for any human to continuously sit and monitor.

Let us write a program that will check every minute whether the web server is listening for inbound connections at the HTTP port.

- To do this, our program needs to know the web server port. So get the port of the web server as an input from the user.
- To check the listening ports of a Linux machine, it is quite common to use the 'netstat' utility that is natively available. This utility prints the listening ports of the operating system.
- Search this output for a text pointing to port 80. If anything is found, then we can assume that the web server is running. If nothing is found, it is better to write to the server log and restart the web server. This would generally make the server available again in most of the cases.
- Depending on the criticality of the web application, you can also send an email to any number of persons indicating that the server is down.
- Finally, schedule this to the cron to run every minute. In this way, an attempt to restart the web server will be made at least within 60 seconds of time.

Scenario 6 – User Command Validation

Scenario

How should I allow the users to execute other commands normally whereas the forbidden commands should throw an error?

Pseudo Code

- **Check:** Script has sudo permissions so that it can read files belonging to other users.
  False: Throw an error saying "Run as root or sudo user"
- **Check:** The forbidden commands and their error messages exist in a file /root directory.
  False: Show editor to create and edit the file.
- Read the list of actual users from the /etc/passwd file.
- Add a function to the '.bashrc' file of all users that checks whether the given argument matches any pattern in the forbidden commands file. If the pattern is matched, throw an error showing the provided error message.
- Point this to the trap directive of the '.bashrc' file so that this gets called before the users' command executes.

Not all the people will have the same level of permissions in an organization. The people in the senior role will have the maximum permissions than a new beginner obviously to avoid any fatal mistakes. Think what would happen if a fresher unknowingly deletes all the core files of a production server. The consequence is too much to deal with. The best way is to block access to the core files using Linux file system permissions.

But, what if he attempts to shutdown the server? Here, we need to monitor and decide whether to allow the user's command to execute or not. This can be done with the help of the bash, which is the default terminal on most of the Linux distributions.

- The program should have the root user permissions so that it can access the .bashrc file of all the users in the system. So if the script is not run as a root user, throw the error prompting for superuser permissions.
- Then check whether the file that has the commands to be blocked from the users exists and if not, exit the program saying the error information.
- The list of users and their home directories can be gathered from the /etc/passwd file of the OS.
- Add the validate commands function and the trap directive to the .bashrc file present in the home folder of all users.
- So for every command, the validate commands function will be called. Now, read the forbidden commands file inside this function and check whether the input command matches any pattern in there.
- If matched, then you can show an error message to the user and exit the program without further executing the user's command. If needed, you can add the email functionality to the function so that every forbidden attempt can send the alert in an email.

Scenario 7 – Disk Usage Alarm

Scenario

How to monitor the Disk usage and alert if it is beyond the given threshold?

Pseudo Code

- **Read Argument:** The device name for which the disk usage should be monitored.
- **Read Argument**: Threshold limit beyond which the users should be alerted.
- **Check:** No device exists with the given device name.
  True: Throw an error and write to the logs.
- **Check:** Threshold limit is greater than the maximum size of the Disk.
  True: Throw an error and write to the logs.
- Run the disk utility command to get the usage of the disk.
- **Check:** Disk usage is greater than the Threshold limit.
  True: Write to the /etc/motd file so that it will be shown whenever a user logs in.
- **Check:** The script is scheduled in the cron.
  False: Add the script to cron such that it gets executed every minute.

As we saw in the scenario 2, the operating system can run out of space and this could end in failure of everything. In Linux, everything is a file and we will dive into this in the next module. Files need some space allocated in the hard disk. So if the hard disk is full, no space can be allocated to files that will end in complete paralysis. To avoid this situation, the disk space should be checked and action should be taken before it reaches the critical stage. Let us program a script to do this task.

- So our program needs to know, which hard disk or storage device should be monitored. It also needs the threshold limit beyond, which it should alert. Give them those as inputs.
- To be more robust, validate the given inputs. Whether the given storage really exists and the threshold is greater than the maximum size of the hard disk. If it is, then show the appropriate errors and exit.
- Then check whether the disk usage is greater than the given threshold limit and if it is, then we can optionally send an email or just write it the log file.
- To do this regularly and frequently, schedule it to the cron to get executed every minute.

Scenario 8 – Sending Files to Recycle Bin

Scenario

How to move the deleted files/folders to the recycle bin?

Pseudo Code

- Set the location of the recycle bin directory and the locations of the remove and copy tool.
- If there are no arguments given, then throw an error asking to provide the file path as arguments.
- Get all the arguments using the getopts function over the while loop.
- Map the arguments with their respective actions using the switch case statement. If 'f' is passed in the argument, delete the files/folders permanently by direct remove command. All the other arguments can be passed on to the remove command.
- Create a recycle bin directory if doesn't exist.
- Copy the files/folders to the recycle bin and prefix the current date to those files.
- Provide all other arguments and file paths to the remove utility.

It is absolutely possible to delete an important file accidentally even for a very experienced person. It is better to have a stage between having the file and deleting the file. A place where our deleted files sit for a limited period of time. Windows operating systems have the recycle bin for this functionality whereas the some Linux distributions don't have. So What! Let us create one.

- Set the temporary directory for deleted files, the path of the binary that actually deletes the file and the path of the binary that copies the file.
- We need the files or folders to be deleted as inputs. So check if inputs are given and if not throw an error requesting the instances.
- We also accept some options for our program that allows us to permanently delete the file without moving to recycle bin.
- Use the getopts will enable us to parse options and then create the recycle bin directory if it doesn't exist in the specified path.
- Now copy the file or folder that we attempted to delete using the script to the recycle bin and then delete the actual file using the rm tool.
- In case, if 'f', indicating force, is given in the arguments, then instead of the previous step, delete the file directly.
- If preferred, set this script as an alias to rm tool in your .bashrc file and you have your own version of recycle bin ready.

Scenario 9 – Restoring Files from Recycle Bin

Scenario

How to restore the deleted files/folders from the recycle bin?

Pseudo Code

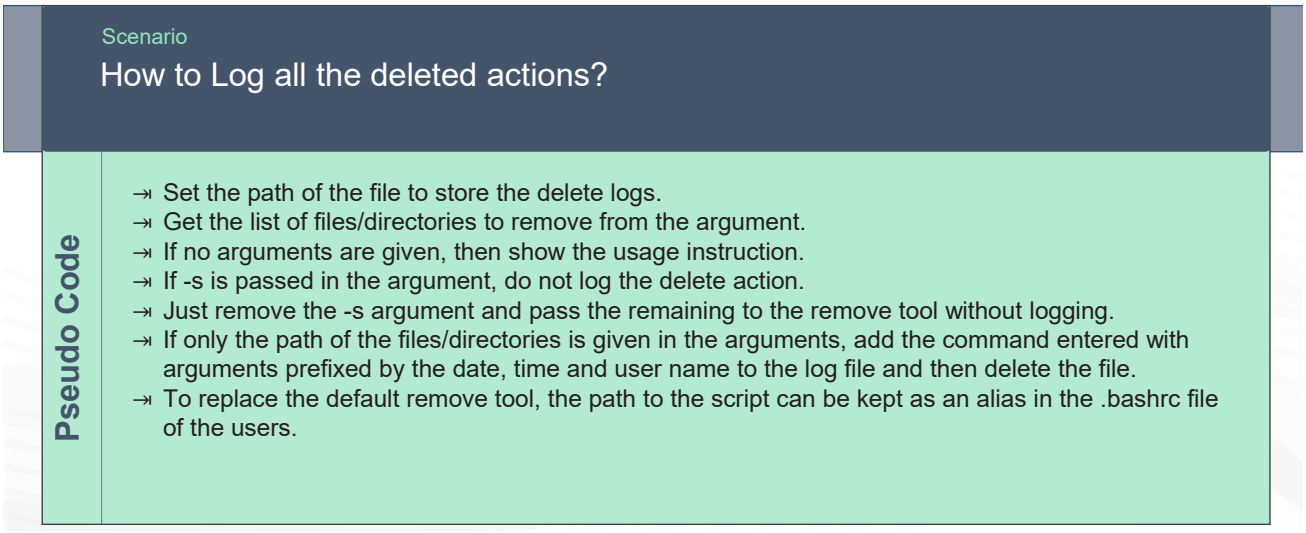
- ⇥ Set the location of the recycle bin directory, current directory and the path of the remove and copy tool.
- ⇥ Check whether the recycle bin directory exists in the current user's home directory. If not, throw an error.
- ⇥ If there are no arguments given, list the files in the recycle bin and exit.
- ⇥ If the pattern is provide in the arguments, search files matching that pattern and if nothing is found, throw an error say no match found.
- ⇥ If many matches are found, show the date, time, size and name of the file or folder and ask the user to choose a version to export.
- ⇥ If zero is entered, exit the application doing nothing. Else check if there is a file/folder with the same name in our current working directory and throw an error if true.
- ⇥ If everything above is fine, then move the chosen file from the recycle bin to the current directory and ask the user whether to delete the other versions of the file in the recycle bin. If yes, then delete other versions of the file. Else Retain them.

In the previous slide, we saw how to send the deleted files to the recycle bin. Now let us explore the ways to restore them back.

- Set the location of the recycle bin, current working directory to store the restored file and the path of the rm and cp tools.

- If the recycle bin doesn't exist at all, then it means no files are there in the recycle bin. So boldly throw an error saying the recycle bin is empty.
- If no arguments are given, then let us list the deleted files in the recycle bin.
- In case if any pattern is provided in the arguments to our program, search for files matching them with their name, date, time and size. If multiple files are found for the same pattern, then ask the user to pick a version that he needs.
- Finally, move the chosen file from the recycle bin to the current directory if there are no conflicts.
- Thus, we can safely delete and restore the files from the recycle bin.
- In order to make the recycle bin really temporary, just add the rm tool to the cron scheduler to delete all the files in the recycle bin.

Scenario 10 – Logging Delete Actions



Any program in Linux can be wrapped by some other program as their own with any modifications. Let us write the program that will abstract the native rm tool and will log all the delete actions made by us through the program.

- Set the location to store the delete logs and fetch the files to delete from the input arguments of the program.
- Since users may not know how our program works, let us show them the usage instructions if no arguments were given.
- If the files to delete are given in the arguments, then send the timestamp, username and the file path to the log file that we initialized previously.
- What if some files should be deleted without making an entry in the log? Simple! Check if the option '-s' is passed to the program and if yes, skip the logging part and delete the file.
- Add this to the .bashrc as an alias to rm tool and that's it.

Scenario 11 – File Formatter

Scenario

How to Read and Display all the file inputs with line numbers?

Pseudo Code

- ⇥ Read the list of files acquired through arguments in a for loop and iterate over each file.
- ⇥ In each iteration, Get the count of lines and the characters with the help of the default word count (wc) utility available in Linux.
- ⇥ List (ls) utility can be used to get the owner of the file.
- ⇥ Print the header bar that displays filename, its lines, its characters and the owner of the file.
- ⇥ Set a counter to indicate the line numbers of the file. Loop through each line of the file and prefix the line content with the respective line number for each iteration.
- ⇥ Do the same for all files and finally show the output through the less utility for better accessibility.

One of the common things to deal with is a bunch of configuration files for various software and daemons running on the operating system. To refer and check the configuration quickly across multiple files, an Administrator needs to open and view all of them one by one and that takes a lot of effort and time. To ease this task, we will pick all the files at once, read their contents, provided some useful data such as the number of lines, characters and size along with the line numbers for all lines in all files. Get all the files passed as an input through arguments and iterate over each file. So with built-in Linux utilities such as wc, ls, we can get the filename, line count, character count and file owner of the files.

Scenario 12 – Encrypting Files

Scenario

How to Encrypt the given files or folders recursively?

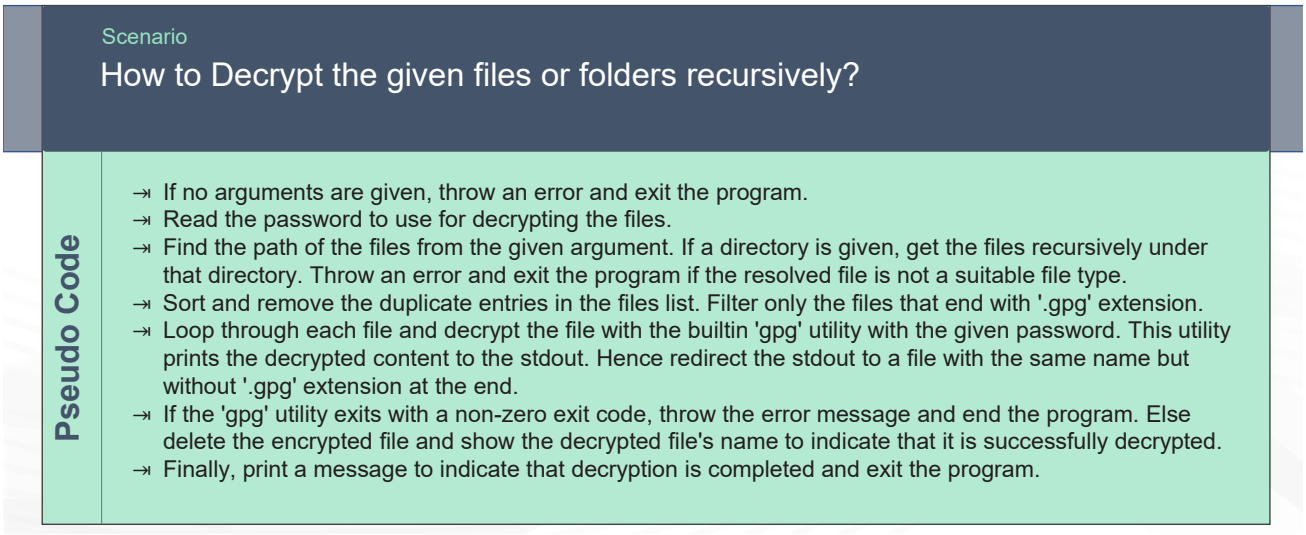
Pseudo Code

- ⇥ If no arguments are given, throw an error and exit the program.
- ⇥ Read the password to use for encrypting the files.
- ⇥ Find the path of the files from the given argument. If a directory is given, get the files recursively under that directory. Throw an error and exit the program if the resolved file is not a suitable file type.
- ⇥ Sort and remove the duplicate entries in the files list.
- ⇥ Loop through each file and encrypt the file with the builtin 'gpg' utility with the given password.
- ⇥ If the 'gpg' utility exits with a non-zero exit code, throw the error message and end the program. Else delete the original file and show the encrypted file's name to indicate that it is successfully encrypted.
- ⇥ Finally, print a message to indicate that encryption is completed and exit the program.

Encryption is just a process of making our files unreadable by adding or removing multiple characters so that it doesn't resemble the actual file in any way. We pass a password to encrypt so that the password will be set along with the encrypted file in an unreadable form. Later the same password is absolutely mandatory to decrypt the file to its original form. 'gpg' is a native utility in Linux to encrypt and decrypt files.

- Our program needs the file or folder paths to encrypt. So get it as an input and if none were given, throw an error. Likewise, read the password to encrypt from the user.
- Now resolve all the files and the folders to a list of the absolute file path and iterate over them.
- In each iteration, call the gpg utility with the file path and pass the password to encrypt.
- Once everything is done, show a success message to the user and exit.

Scenario 13 – Decrypting Files



To decrypt the files that we encrypted in our previous slide, fetch path of the files or folders from the arguments and get the password to decrypt as well.

- Resolve all the files and folders to a list of absolute paths and filter the files that have the .gpg extension.
- Loop through all the gpg files and use the gpg utility to decrypt the files. Remember gpg decrypt will directly print the contents to the output instead of writing to a file.
- So redirect the output to the gpg file's path but without the .gpg extension.
- Now let us remove all the gpg files that got encrypted and that is it.
- Print the success message to the user and exit the program.

Scenario 14 – System Information

Scenario

How to show the most often required status and information of the system?

Pseudo Code

- The most often checked status of a computer while troubleshooting is given below.
- date - Prints the current date and time based on the local time zone of the system
- w - A built-in utility to show currently logged in users, their session duration, their IP and their mode of login.
- last - This utility shows the history of logins made. Since we don't need the entire history, only the last 5 logins can be shown.
- df - It prints the list of available disks with their used space, free space, total space, no. of blocks and their mount points.
- free - This tool provides the details about the installed Memory and the allocated Swap Memory.
- top - This helps in displaying the running processes with their memory and CPU consumptions details.
- netstat - This network utility can provide the information of established and listening connections from the current system.
- ss - A very extensive tool to get the statistics of all the sockets in the system.

Scenario 15 – Bulk File Downloader

Scenario

How to download all the files provided by the list of given URLs?

Pseudo Code

- Set the default directory to store downloaded files.
- Read the path of the downloads list from the arguments. If it is not provided, throw an error and exit the program.
- If a second argument is given, set the given argument as the download directory.
- Check if the download directory exists. If not, create the directory.
- wget - A built-in utility to download the files from a HTTP or HTTPS endpoint.
- Loop through each lines of the download list file and use wget in each iteration to download the file by providing the URL and path to download as inputs.

In some places, for example, when a operating system is installed newly on a machine, it won't have any additional software that is required.

So we need to download all of them again and again if we haven't taken any backups of it. Think of getting the security patches for the servers and software required in production setup. It will be tedious to download one by one.

Write the program that optionally gets the download directory else uses the default directory in the home directory and gets the file from which the list of download URLs will be retrieved. We can just loop over each line from the file containing download URLs and using a built-in utility named wget, it is far easier to download the files by passing in the given download URL.

Scenario 16 – Install LAMP Stack

Scenario

How to install Apache, MySQL, PHP on Linux machine (LAMP stack)?

Pseudo Code

- Update the Apt packages so that apt tool knows where to get the required softwares.
- Pull the latest update and security patches available for your Linux distribution by the 'apt-get upgrade' command.
- Running 'apt-get install Apache2' will install the latest stable version of apache web server (2.4.x).
- Let us install PHP 7 as the programming language of our choice. Most popular open source software such as Wordpress (Blog), Drupal (CMS) and Magento (E-commerce framework) are written in PHP.
- As the data backend, MySQL can be installed. mysql-server listens on port 3306 for incoming connections and mysql-client is capable of initiating a connection to the mysql-server.
- To provide appropriate permissions for the executable code, we change the ownership of the /var/www directory so that apache server will have sufficient permissions to read and execute the code located under this directory. For modern web applications, we need the Apache's rewrite module to be turned on.
- Finally restart the Apache web server for all the changes to take effect.

Apache, MySQL and PHP on Linux popularly so-called LAMP stack is just a couple of softwares capable of running a full-fledged application on their own.

Apache is a web server that handles the HTTP connections made through the browsers by users. MySQL is a Database server that allows to store, retrieve and query data from it. PHP is a general-purpose programming language extensively used for web development. So we have one of them in each kind and that's more than enough for running complete applications. Wordpress, Drupal, Magento are web applications written in PHP and highly compatible with MySQL as the Data backend.

Apt-get is a Linux utility that allows installing various softwares just by running it with the name of the software. It exactly knows the location of the software we request using the apt-list. That's why we update the list at the beginning itself. Apache itself runs as a Linux user so that it can make use of Linux Filesystem permissions. So, we explicitly allow the access to files on the www directory by transferring the ownership of the directory to the www-data user. Thus, on restarting the server, we have a complete environment ready to deploy any suitable applications.

Scenario 17 – Get NIC's IPs

Scenario

How to Find Public IP and private IP of the given network interface card?

Pseudo Code

- Check if NIC name is given in the argument. If not, throw an error and exit the program.
- ip - A built-in tool to get the network related information and modify the network settings of the Linux OS.
- If the given NIC name doesn't exist, throw an error saying that NIC doesn't exist and exit the program.
- Using the ip tool, get the information of the given NIC and pipe the output to the 'sed' utility to fetch the Private IP address of the interface.
- To find the Public IP of the interface, we need some server outside the LAN. Hence the public site http://ifconfig.co can be used to retrieve the Public IP of our system.
- By doing a 'curl --interface eth0 http://ifconfig.co/ip', the Public IP of the eth0 interface will be returned in the stdout and can be used further.

A private IP is an IP address that is used within the Local network it is connected with. This IP can be used for all sort of communications within the Intranet. i.e. the connection has never reached to the internet. A public IP is an IP address assigned automatically to the router which can be publicly accessed from anywhere around the globe. It is unique and can be assigned only to one adapter at a time. Companies and Hosting providers usually buy a bulk amount of Public IPs from ARIN or RIR for their hosting needs. IT administrators check their IPs often to ping and verify their connectivity across machines. Though the commands to find them are short, they will output much data about the interfaces in which the IP address should be searched.

To keep this precise, let us write a program that fetches both the IP of the given network interface.

- Using the native ip utility, the number attached network interfaces can be listed along with their private IPs. By piping the output to another utility named 'sed', we can easily fetch the Private IP alone.
- Method to retrieve the public IP is somewhat different than the methods to get private IP. Because the machine itself doesn't know its public IP since the public IP is attached to the router, not directly to the machine.
- So in this case, the servers outside the network can help us identify our public IP. We need a mirror to see our self would be the best analogy to use here.
- http://ifconfig.co is one of the public sites that provide the public IP from which we are accessing that site. By simply doing a curl on that site, we can get our public IP.
- By displaying both, the IT administrator can easily get exactly what he needs.

What did you Grasp?

1. What is the easiest way to get the name of folders or files at a given location recursively in Linux?

   A) Configuring the Web Server to list down the directory by providing required permissions.
   B) With the help of 'find' command, the folder/file names can be acquired.
   C) Using the 'ls' command coupled with 'sed'.
   D) Doing 'curl' on the localhost specifying the path.

2. What is the use of 'df' command?

A) Show the Memory usage and available memory in the system.
B) List down the running process.
C) Display the list of disk usages and the available disk spaces.
D) Logout from the system.

3. Scenarios Where Automation Prevents Errors

Scenario 1 – Querying User's Command History

Scenario

How to fetch commands executed by all users that match the given pattern?

Pseudo Code

- **Read Argument:** Command pattern to search for
- **Check:** Script has sudo permissions so that it can read files belonging to other users.
  False: Throw an error saying "Run as root or sudo user"
- Get the list of '.bash_history' file of all users.
- Search for the text that matches the given pattern in all the files listed above.
- Show the username and the line number of the file that matches the pattern
- To get the timestamp of the executed command, set the history time format in the '.bashrc' file of all users.
- Now repeat the search and show step to fetch the executed commands with their respective timestamps.

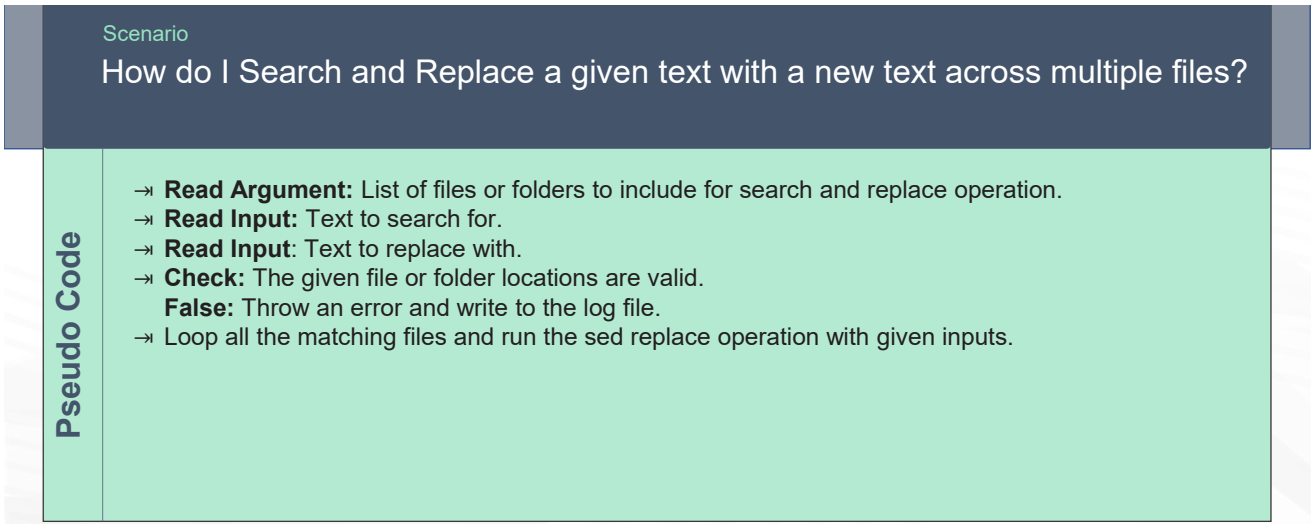
In an organization, several people from multiple teams will be working on the same server doing their day-to-day work such as deploy new software releases, updating security patches, gathering the analytics, doing the proof of concepts, etc. What if some important file is missing and you are asked to find any activities made to that file. You need to log all the actions somewhere. Writing a wrapper for each action is almost impossible as well as unfeasible.

That is where bash comes in a most elegant way. Bash logs all the commands executed on it and stores the log on the .bash_history located in the home directory of the current user. The reverse search (CTRL-R) makes use of this to automatically populate the commands from your history.

Let us write a program that allows us to query on this history of all users. Thus we do not need to manually look into each and every file through thousands of lines of history.

- To access the .bash_history of other users, you need to have a super user (root user) permissions. If the user executing the program doesn't have these permissions, show them an error and exit the program.
- Now get the list of .bash_history files by fetching the list of users from the password file of the Linux. With the help of grep utility, you can provide a pattern and only the commands that can match the pattern can be retrieved.
- Properly format the received output with their usernames and the line number of the files.
- One drawback is that bash history doesn't have any timestamps of the executed command. But it allows us to define the log format. So by adding date-time format to the environment variable, bash will log the executed commands along with their timestamps.
- By asking a program to search for the pattern, there will be no way to miss anything.

Scenario 2 – Bulk Search and Replace



Imagine replacing a text with some other text across hundred files with around ten replacements in each file. How long would it take? What is the probability of making errors though we just copy and paste the text? You might be familiar with search and replace on Word or Excel. But it can't do this operation across files. Is there any utility that allows us to accomplish this? Maybe not natively in Linux.

Why do we need to this kind of an operation anyways? If you have your public IP hardcoded in all your configuration files, and if the IP changes, then you need to update all the files with the new IP. It can be a hostname, username, password or just any plain text. Let us create a program that gets a list of files from the arguments as input. Prompt the user to provide the text to search for and the text to replace with.

- Before executing the operations, validate whether the given file path exists.
- Iterate over each file and use the sed utility to search and replace the given text and write to the file using the '-i' (in-place) option available in sed.
- Thus the search and replace operation is carried out across all files.

Scenario 3 – Alphanumeric Validator

Scenario

I need to check whether the given input is a valid alphanumeric Character.

Pseudo Code

- **Read Input:** A text input that needs to be validated.
- If entered input is empty, throw an error asking to enter any valid text.
- Use the sed utility to replace all the characters except letters and numbers with blank values. Store this modified value in a new variable.
- Compare the modified variable with the input. If both are same, then the given input doesn't have any characters other than letters and numbers.
- Based on the output from the above condition, show an appropriate message on the terminal.

It is very important to verify and validate the input provided by the user. Say you have written a script that modifies some core configuration files of networking and if the user provides some random text instead of an IP, then it may even completely disable the connectivity of your system. To prevent such sort of errors, it is mandatory to validate the provided inputs. Design for failure and nothing will fail.

Let us write a sample program that reads an input from the user.

- If the input is empty, throw an error saying that the given text is invalid.
- Now using the sed utility, just replace all the characters other than text and numbers and check whether the given text is equal to the modified text.
- If both are not the same, we can determine that the given text is invalid.
- Likewise, any different kind of text validations can be done using sed along with regular expressions

Scenario 4 – File Search Manager

Scenario

How to file search manager that allows to find/filter files using timestamp, name, permissions, file ownership, size, etc?

Pseudo Code

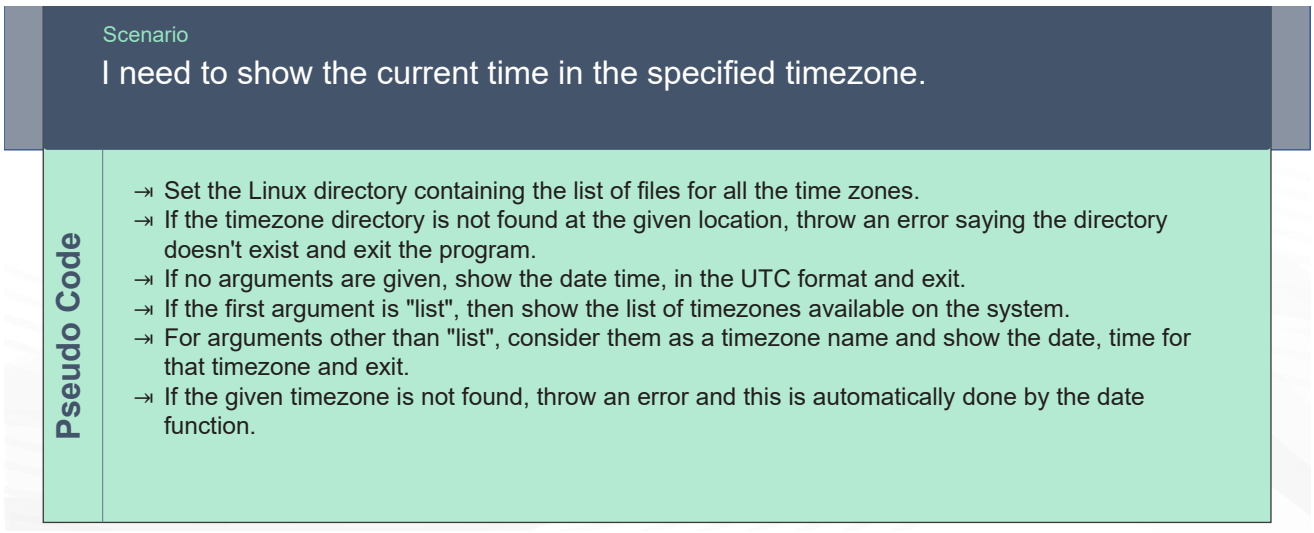
- **Read Argument:** The Directory to search files.
- **Read Input:** name, timestamp, size, owner, permission.
- Construct the find or locate command based on the given input Arguments.
- Run the constructed command and return the list of files matching the given parameters.
- If no match is found, the find command will automatically return an null value to the stdout.

Starting from configuring a web server to deploying a complex application, there will be thousands of files involved and it is very difficult to locate them exactly unless we have any utility to help. Luckily the 'find' utility help us filter the files based on the criteria we give. Yet it is not that much user-friendly.

Let us create a wrapper script that uses the 'find' utility as its core, but provides a nice interface to read the inputs from users.

- Get the directory to search for files from the argument and the name, timestamp, size, owner and permissions of the file as inputs.
- Based on the given arguments construct the filter options that will be passed to the find as arguments.
- Once the filter options are constructed, run the find command with these options and show the list of matched files that are returned by the 'find' utility.
- If the find utility is not able to find any matches, it returns with a non zero exit code and by identifying that show an appropriate error message to the user.

Scenario 5 – Timezone Helper



Time zone has equal importance to other configurations. Linux generally has a separate directory allocated for holding different time zone files. It is usually located at "/usr/share/zoneinfo" directory.

- Set this directory and use it for fetching the list of timezones.
- Sometimes it can be different for some Linux distributions. In that case, show an error saying that timezone directory is not found.
- To list the available time zones, let the user run the program with the 'list' argument. If no arguments were given, then show the date and time of the UTC time zone.
- If any time zone arguments are provided, check whether they exist in the time zone directory and if yes, show the time in that time else throw an appropriate error response.

Scenario 6 – Fetch Latest/Oldest Files

Scenario

How to show the 10 latest/oldest modified files in the given directory?

Pseudo Code

- Set the maximum number of files to display.
- Create an infinite while loop with an switch case statement in which the default switch breaks the loop.
- For each iteration, shift the given arguments so that the loop definitely when it is out of arguments.
- Add switch case conditions to list files recursively, to list hidden files, to show the oldest modified files, to show the help message and to set the number of files to display dynamically.
- If path is not provided in the arguments, list the relative path of the files in the working directory.
- Use the built-in 'find' utility with options constructed in the previous steps to fetch the required files.
- Pipe to sort and to limit the number of the files.
- By using the tradition 'ls' command with above filtered file list, print the file details with some standard ANSI colors.

This script might get very handy in places where you need to quickly locate the latest or oldest files.

- Initially specify the maximum number of files to display which can be overwritten with the user's value from arguments if any.
- By looping through given arguments to map the options with their appropriate functionalities. For example, recursively including files, hidden files, etc.,
- The 'find' utility will be used with the constructed query filters from the options to fetch the required files.
- Once the file list is generated, more details about them can be acquired using the ls command.

Scenario 7 – File Truncator

Scenario

How do I truncate the given file to the specified size?

Pseudo Code

- Check if the given number of arguments is equal to two. If not, show the usage message and exit the program.
- Set the first argument to the size variable and the second argument to the path variable.
- dd - It is one of the core GNU utility that facilitates a lot of block level and file level operations.
- 'bs' argument passed to it mentions the number of bytes to take at a time
- 'seek' argument refers the block to begin with.
- 'if' argument reads from a file instead of stdin and 'of' writes to a file instead of stdout.
- Thus by using these parameters, a file can be adjusted to any size either by removing the excess data or by adding null data to the file.

'dd' is a GNU utility available in Linux operating systems to perform file level and block level operations. This program can help in truncating log files which are pretty much very large in size. We may require only a part of it.

- Get the size that the file should have and the path of the file as inputs from the arguments.
- If the given number of arguments is not equal to two, show an error message and exit the program.
- By using the necessary arguments, dd utility can help in truncating or extending the file to exactly match the specified size.
- If the file size is greater than the specified size, then additional data will be trimmed. If the file size is lesser than the specified size, then null characters will be appended till the file reaches that size.

Scenario 8 – Colorful difference Viewer

Scenario

How do I colourize the standard output by traditional diff tool?

Pseudo Code

⇥ If two arguments are not given to the script, throw an error and exit the program. To show the differences better between two files, let us add colours indicating whether a part is removed, added, modified or left untouched.
⇥ Set colours for all differences such as red colour for removed content, green for new content, blue for modified and white background colour to differentiate file name from its content. Use the builtin 'diff` tool to list down the differences. 'N' argument treats absent files as empty, 'a' treats all files as text, 'r' to be recursive and 'u' to use plus / minus characters instead of angle brackets.
⇥ Now we need to choose a color for the text by finding the pattern it has. Then prefix the content with ANSI color and suffix with ANSI Reset color.

For example, if an administrator has changed a configuration file and wants to double check whether all the planned changes have done, he usually needs to compare the new file with old one. In another case, he may be required to compare the file with some other file on a different server. The built-in diff tool will be helpful in doing this. But it is not that great since it doesn't provide any color indications. So it is difficult to understand the differences by simply looking at the given different symbols.

Let us write a program that adds colours to the output of the diff tool which will vastly improve the accessibility of using this.

- Get the path of the two files that need to be compared. Initialize the colors for the various differences.
- Run the diff utility with the following arguments - 'N' argument treats absent files as empty, 'a' treats all files as text, 'r' to be recursive and 'u' to use plus/minus characters instead of angle brackets.

- Next, choose a colour for the text by finding its pattern. Then prefix the content with ANSI colour and suffix with ANSI Reset colour.
    - Changed content - Blue - ***CONTENT**** (or) ---CONTENT---- (or) @CONTENT (or) !CONTENT
    - Removed content - Red - <CONTENT (or) -CONTENT
    - Added content - Green - >CONTENT (or) +CONTENT
    - Files - White BG - Only in FILENAME (or) Index FILENAME (or) diff FILENAME

Scenario 9 – Text Transform

Scenario

How to transform the given input text to Uppercase, Lowercase and First letter Uppercase output?

Pseudo Code

- ⇥ Read the input from the argument and if none is given, throw an error and exit the program.
- ⇥ To Transform the entire text to Uppercase, echo the input and pipe the stdout to the 'awk' utility. awk has a 'toupper' function that transforms the given input to Uppercase.
- ⇥ Similarly there is a 'tolower' function in awk that can transform all the characters to Lowercase.
- ⇥ For transforming the First letter of the sentence to Uppercase and the remaining letters to Lowercase, Use the 'substr' function of awk to splice the first letter and transform it through 'toupper' function and using the same 'substr', get and pass the remaining letters to 'tolower' function.
- ⇥ To make these methods convenient for use, define them as separate functions that receives the input as arguments.

There will be places where you may required to transform the case of the given text. It may be to validate/map the input such as Y or y can be used to indicate yes in the confirmation prompt. Or to check whether it matches any other particular text or just to match the strings case insensitive.

- Read the text to transform from the input and if none is given, exit the program by throwing an error message.
- ‘awk’ is a native tool with lot of text formatting options in Linux operating systems.
- It has functions to transform the given text to lower case and to upper case as well.
- By passing the given text to these functions in awk, the given text can be transformed to the required case.
- It is also possible to combine these functions to make the first letter to upper case while the remaining letter to lower case characters.

What did you Grasp?

1. Choose the functionality of the 'dd' GNU utility available in Linux operating systems?

   A) Remove the old log files from the default log folder.
   B) Manage user permissions on a single place.
   C) Check the suspicious actions made on the Linux Kernel.
   D) Allows disk level and block level operations of the file system.

2. Which of the following files has the history of commands executed by the users?

   A) .bashrc
   B) .profile_history
   C) .bash_history
   D) .profile

In a nutshell, we learnt:

1. Various scenarios required in IT Operations and have an idea to create an approach to automate the scenario.
2. Gathering the inputs required for the program and listing of the outputs that need to be displayed back.
3. Validating the given inputs strictly as much as possible so that the program can be more robust and less prone to surprising errors.
4. Returning a meaningful message in the error response that will help the users to debug and troubleshoot the program.

Release Notes

B. TECH CSE with Specialization in DevOps

Semester three -Year 02

Release Components.
Facilitator Guide, Facilitator Course Presentations, Student Guide, Mock exams and relevant lab guide.

Current Release Version.
1.0.0

Current Release Date.
2 July 2018

Course Description.
Xebia, has been recognized as a leader in DevOps by Gartner and Forrester and this course is created by Xebia to equip students with set of practices, methodologies and tools that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.

The language used in this course is US English. Our sources of reference for grammar, syntax, and mechanics are from The Chicago Manual of Style, The American Heritage Dictionary, and the Microsoft Manual of Style for Technical Publications.

Bugs reported	Not applicable for version 1.0.0
Next planned release	Version 2.0.0 Feb 2019