Automating Tasks

Scheduling a Single Task

> Scenario

Periodically, the developers at Develetech need to execute a task after hours. The schedule is not predictable and they need to be able to manage these tasks themselves. You will use the at command to satisfy this requirement.

> Objectives

- Completing this activity will help you to use content examples from the following syllabus objectives:
    - 2.6 Given a scenario, automate and schedule jobs

Schedule a task to run for two minutes into the future from your current time

1. You will schedule a task that deletes a file from your home directory two minutes into the future.
    - Log in as `student01` with `Pa22w0rd` as the password.
    - In your home directory, use the `touch` command to create a file named `fileA`
    - Check the current time on your system by using the `date` command.
    - Enter at now + 2 minutes to access the interactive mode of the at command.
    - Enter `rm -f ~/fileA` and then press `Ctrl+D` to return to `Bash`.
    - Enter `atq` to view the scheduled job.
    - After two minutes, ensure that the command executed by checking the contents of your home directory to see if `fileA` was removed.

Scheduling Repeated Tasks

> Scenario

Develetech adopted a new policy that requires all users to fill in their time sheets every day. You'll create a daily reminder for all user systems.

> Objectives

- Completing this activity will help you to use content examples from the following syllabus objectives:

  > 2.6 Given a scenario, automate and schedule jobs

1. Schedule a cron job to email a reminder every day at a specified time

    - Enter `sudo crontab -u cmason -e` to specify a cron job for Chris Mason.
    - Verify that Vim opens a temporary file automatically.
    - Type the following line in the file:
    - `MM HH * * * /bin/echo "Please fill in your time sheet."`

- Replace `MM` and `HH` with the appropriate minute and hour time values in 24-hour time format. Ensure that the time you enter is three minutes ahead of the current system time. This way, you'll be able to see the message during the lab.
- For example, if the time is 2:30 P.M., you'd type:
- `33 14 * * * /bin/echo "Please fill in your time sheet."`
- Save and close the file.
- From the desktop menu, select the icons at the top-right, then select `student01→Log Out`.
- `Select Log Out`.

2. Verify that *Chris Mason* received the reminder for the scheduled job

- Log in as *Chris Mason*.
  - You can ignore the Welcome screen, or you can step through the wizard to dismiss it.
- Open a terminal.
- Wait for the time to pass for the cron job to execute.
- Remember, you can use date to check the time. You can also check the time from the desktop menu in the GUI.
- Enter `mail`
- Enter `1` to read the contents of the first email message.
- Verify that the mail contains a reminder to fill in the time sheet.
- Press `q` to quit the mail service.
- Log out as `Chris Mason` and log back in as your `student01` account.

Implementing Version Control Using Git

> Scenario

The development team needs a way to easily manage the different versions of the code they write. Multiple developers will be working in conjunction on the same project, so they need to a way to minimize conflicts while being able to revert to older versions of code, if necessary. So, you'll set up a Git repository for the developers so that they have a distributed version control system to work from.

> 5.2 Given a scenario, carry out version control using Git

1. Install and configure a Git repository

- Enter `sudo systemctl kill packagekit` to halt any updates the system may be doing.
- Enter `sudo yum -y install git --disablerepo=internal-repo` and wait for the installation process to complete.
- Enter `git config --global user.name 'Student User'`
- Enter `git config –-global user.email 'student01@develetech.com'`
- Create a directory in your home directory called dev-project and use the cd command to enter the directory.
- Enter `git init` to designate the dev-project directory as a Git repository.
- A message is returned from Git indicating the repository is initialized.
- Enter `ls -a` to view the `.git directory` created by the initialization process.

2. Create and manage a project using Git

- Use a text editor to create a file named `HelloWorld.txt`

- Enter the following text in the `HelloWorld.txt` file:
- `Hello, World! From Student01`
- Save your changes and close the editor.
- Enter git status to check the status of the `HelloWorld.txt` file.
- The file is marked as "Untracked", meaning it is not yet managed by Git.
- Enter `git add HelloWorld.txt` to enable Git to manage the file.
- Enter `git commit -m "Initial Commit"`
- This updates Git with the version information for the `HelloWorld.txt` file.
- Enter `git status` to check the status of the `HelloWorld.txt` file.
- The output indicates that there is nothing to commit because the `HelloWorld.txt` file version is now managed by Git.

3. Commit a change to the Git repository

-) Use a text editor to open the `HelloWorld.txt` document, and add the following on a new line:
- `Git version control test`
- Save your changes and close the editor.
- Enter `git status` and observe that Git reports the `HelloWorld.txt` file as modified, but that the changes are not yet committed to the repository.
- Enter `git add HelloWorld.txt` to stage the changes.
- Enter `git commit -m "Revision 1"` to commit the changes to the master copy of the file.
- Enter `git status` and notice that there are now no changes to commit to the repository.
- Enter `git log` to view the revision history of the repository.
- Times and dates for the initial commit and the revision have been logged.