DevOps – Elective - Virtual Hackathon - 1

Date : 26/08/2021 **Duration : 3 Hrs**

Marks : 30 **Student /s Roll Numbers (Team of Two)**

Guidelines :

- *Copying of material in any form is strictly not allowed for assessments.*
- *One copy of Final deliverable submission on a GitHub account from a team of two students is allowed. Emailing copies are not allowed.*
- *Late submission and excuse for late submission is not allowed*

Project case study :

In a company named XYZ, Programmer A and Programmer B have been assigned new module development work on two shifts. They started writing their programming code in a new directory created under the root folder in their machine. They finally completed the work on time by following the given work instructions as per the application maintenance task schedule sheet.

Your team is instructed to follow similar guidelines mentioned below and complete the job within the duration mentioned.

Note: Install all the relevant software required before starting the tasks and mention the list of installations done.

Naming Conventions :

- ***HEAD : Main directory***

- ***Working Directory (WH) : <YourName>.DevOps.*** ***ex : ./umas/Team1/DevOps***
- ***Program Names: PRG1,PRG2,PRG3..suffixed by your team names : ./umas/Team1/Devops/PRG1.txt***
- ***Branch Names : BRANCH1, BRANCH2, BRANCH3.. ex: ./umas/Team1/Devops/Branch1/PRG1.txt***
- ***Login to your Git Hub repository and create a new repository DevOps***
- ***Identify each commit as C1,C2,C3 ..in the diagrams***

Instructions - 1:

1. Programmer A creates two files **(PRG!,PRG2)** in working directory and makes changes different changes in those files and separately tracks in local git repository
2. Checks the status and take s the log of different activities. At this point, he has two files under source control
3. He also keeps the tracking copies on remote Git Hub repository **DevOps** using Git commands
4. **<Takes the snapshot of the Screen- SCREEN SHOT 1 >**

Instructions - 2 :

1. After some time, the same programmer starts working on a different **PROFIT-Loss** module on a new **Branch1** and modifies the second file **PRG2** two times and keeps under source control.
2. While Programmer A is managing his version of files in **Branch1**, Programmer B working in evening shift adds different module called **PROFIT-Correction** on **Branch2** and edits the same file PRG2. He also maintains different versions of the two changes in git separately

3. After two days of module development code changes, both Programmer A and Programmer B **merges** their files and pushes local main repository to remote git hub repository **DevOps**
4. They delete both **Branch1 and Branch2**
5. He checks the status and maintains the history of the files using Git commands
6. **<Take the snapshot of the git commands and git hub screen – SCREEN SHOT -2>**

Instructions - 3: : Identify merge conflict and solve

1. Next day, **Programmer A** clones the latest DevOps directory from the Git Hub DevOps and starts working on it.
2. Programmer A creates **Branch3** makes some changes on **PRG1.** He does not want to put under tracking until the modification is complete on the file.
3. Now, He switches to main branch to edit the same file **PRG1** for some modification.
4. Merge conflict appears when the same copies of the file **PRG1** is edited in two branches. Create Merge Conflict situation
5. Resolve the conflict and complete the Commit
6. **<Take the snapshot (SCREEN SHOT 3) of the Git commands executed for the above steps i.e before Merge Conflict and after solving the Merge Conflict>**
7. Where is the HEAD Pointer now?

Instructions - 4: Work on Bug Fix on high priority on a Separate branch

1. Meantime, A new high priority BUG fix work is assigned to **Programmer B** who comes in second shift. As he is still working on the file **PRG2** he does not want to commit the changes done in this file . However, he is asked to complete the BUG Fix job ASAP.
2. He uses the Git commands and saves existing work and takes up new BUG FIX job in **PRG3** file in **Branch3**
3. After he commits the bugfix file, he gets back to his previous work in **<Branch2> file PRG2 and** completes the work by adding a few lines to **PRG2**
4. He also pushes the latest version to Git Hub repository and checks it on Git hub for the new file **PRG3** and new **Branch3**
5. **<Take the snapshot (SCREEN SHOT 4) of the Repository after Bug fix having the Git commands executed for the above steps>**

Instructions - 5 :

1. After the Bug Fix issue is resolved by evening Shift Programmer B , Programmer A decides to work on **Branch2.** He used some additional Git commands and achieved the expected State.
2. He makes changes in **PRG2** but then he realizes the filename should be all lowercase. He Change the filename to lower case letter a **prg2** and completes the Commits. He merges with main branch and pushes the main to remote Git Hub repository
3. **<Take the snapshot (SCREEN SHOT 5) of the screen with the Git commands executed for the above steps>**
4. **Show the state of all the commits and branches diagrammatically**

Instructions - 6 : Rewrite the commit date

1. Now the coding work got completed but with one week delay in the Project plan. Hence, Programmer A was asked to change the date of the last commit before submitting the final local copy of main branch to Git hub repository. He does the date changes in the files.
2. While he was working on the date changes, his manager notices a typographic error in **prg2.**
3. Manager instructs him to revert back to the first version of the commit he has just amended on **prg2** .
4. Programmer A performs the undoing of the commit and retrieves the first version of **prg2,** modifies and commits the file in local branch.
5. He checks the history by using some Git commands .
6. **<Take the snapshot of the screen (SCREEN SHOT 6) with the Git commands executed for the above steps>**

Instruction 7:

1. Programmer A combines all the programs in main branch.
2. He deletes all the other branches except main branch and bug fix branch.
3. Programmer now stores the latest version to remote Git hub repository suing Git commands
4. **<Take the snapshot of the screen (SCREEN SHOT 7) with the Git commands executed for the above steps>**

Instruction 8 : Pushes the existing state of development work in a private remote server directory for integration Testing

1. Programmer A creates a README.md file and updates all the Git commands used so far and puts on a remote private server machine for future use.
2. He pushes the latest version of the Bug fix branch and master branch separately.
3. He verifies the Remote private server directory by logging to the remote Private server using his credentials.
4. **<Take the snapshot of the screen (SCREEN SHOT 8) with the Git commands executed for the above steps>**
5. **Draw the complete workflow of all the commits diagrammatically.**

Final Deliverables : On Git hub Account

- *Capture all Eight Screen Shots and Final workflow diagram in one Single file*
- *Upload this Final PDF file to your Git Hub account for Evaluation.*

Rewards: First Twenty five Students who complete the Hackathon in less than one hour using innovative methods will get an opportunity to CHAT with Google Cloud Head over a CUP of Tea/Coffee. Coffee or Tea will be sponsored to them…!!!@@@ - This is a personal TREAT.