Testing in DevOps

DOu – Certified Tester in DevOps (CTD)
Exercise Solutions

HO-1.3.3(HO-0)
Exercise - Demonstrate how to apply the main features of a configuration management tool: check-in, check-out, merge, conflict resolution, branching

- Perform an exercise to use Configuration Management Tool such as Git for following operations:
  - GitHub Web Fork
  - Using GitHub Web
  - Using GitHub Desktop
  - Using Git cmd
  - Branching & Merging
  - Conflict Resolution

HO-1.3.3(HO-0)
Exercise Solution - Demonstrate how to apply the main features of a configuration management tool: check-in, check-out, merge, conflict resolution, branching

- GitHub Web Fork:
  - Open the browser and go to https://github.com/login
  - Login to your github web account
  - Launch URL https://github.com/umangsaltuniv/FirstRepo
  - Click “Fork” at right top section
  - “FirstRepo” repository will be added on your GitHub account
  - Click “Code” button
  - Click “Download ZIP” button
  - Repo code will be downloaded on your machine
  - Unzip the folder & see the code(HelloJava.java)

HO-1.3.3(HO-0)
Exercise Solution - Demonstrate how to apply the main features of a configuration management tool: check-in, check-out, merge, conflict resolution, branching

- GitHub Web :
  - Create new repository(e.g. github-web) on GitHub web
  - Upload code(HelloJava.java) to github web from your machine(where you have downloaded **FirstRepo** code from GitHub in previous exercise) by clicking “uploading an existing file” link
  - Write your comments in “Commit changes” section
  - Click “Commit changes” **button**
  - See the code in appropriate repo(e.g. github-web) on GitHub web

HO-1.3.3(HO-0)
Exercise Solution - Demonstrate how to apply the main features of a configuration management tool: check-in, check-out, merge, conflict resolution, branching

- GitHub Desktop :
  - Click “Code” button of your repo that you have created in previous exercise(e.g. github-web)
  - Click “Open with GitHub Desktop” button
  - Click “Open GitHubDesktop” button
  - GitHub Desktop UI will be launched
  - Browser **C:\DO-United\GitHubRepo** path under “Local path” section
  - Click Clone
  - GitHub Desktop UI will open the cloned repository and code will be downloaded on local machine under **C:\DO-United\GitHubRepo**
  - Open the code file “HelloJava.java” on machine and do code changes, save & close the file
  - Make sure Current branch is selected as “main” on GitHub UI
  - Enter Summary comment under “Summary” text box section that is above to “Description” textbox
  - Click “Commit to main” button
  - Click “Push origin” button
  - Once push is done then go to your appropriate repository(e.g. github-web) on GitHub Web & Refresh the page
  - Code changes will be displayed there in github web

HO-1.3.3(HO-0)
Exercise Solution - Demonstrate how to apply the main features of a configuration management tool: check-in, check-out, merge, conflict resolution, branching

- Git :

  Pre-requisites:

  Create **git** folder in C drive(You can keep it in any drive).

  - Launch GitBash from C:\Program Files\Git\git-bash.exe
  - Run following commands in GitBash:
    - git --version
    - cd /C/git
    - git config --global user.name "<your github username>"
    - git config --global user.email "<your email id>“
    - git config user.name
    - git config user.email

HO-1.3.3(HO-0)
Exercise Solution - Demonstrate how to apply the main features of a configuration management tool: check-in, check-out, merge, conflict resolution, branching

- Git(Cont…) :
  - Launch below command in GitBash:
    - git clone https://github.com/umangsaltuniv/FirstRepo.git
      Note: Use your repo link that you have forked in first exercise in above command. You can get the repo link by clicking on clipboard copy button under "Code" button
  - Go to **C:\git\FirstRepo** folder
  - Do changes in HelloJava.java file, Save & close the file
  - Go to GitBash
  - Launch below commands in GitBash:
    - cd FirstRepo
    - git commit -m "myfirstcommit" HelloJava.java
      Note: You can give your own comment instead of "myfirstcommit"
    - git push origin master
    - git log
    - Go to github web, refresh the appropriate repo page and see the code changes there

HO-1.3.3(HO-0)
Exercise Solution - Demonstrate how to apply the main features of a configuration management tool: check-in, check-out, merge, conflict resolution, branching

- Branching & Merging:
  - Create new repo(e.g. branching-merging) on github web(e.g. by using "uploading an existing file" option) having HelloJava.java file in it
  - Launch GitBash and run below commands on GitBash
    - cd /C/git
    - git clone https://github.com/umangsaltuniv/Branching-Merging.git

      Note: Use your repo link that you have created in this exercise in above command
    - cd branching-merging
    - git status
    - git branch <your new branch name>   (e.g. mybranch)

      Note: You can keep any branch name
    - git checkout mybranch
    - touch HiJava.java
  - Go to C:\git\branching-merging folder
  - Add code in HiJava.java file, save & close the file
  - Go to GitBash and launch below commands
    - git add .   (Keep space & dot after add)
    - git commit -m "Added HiJava file"

HO-1.3.3(HO-0) Exercise Solution - Demonstrate how to apply the main features of a configuration management tool: check-in, check-out, merge, conflict resolution, branching

- Branching & Merging(Cont…):
    - git push origin mybranch
  - Go to GitHub web repo & refresh the repo page, now branch count will be 2(main & mybranch). main branch will contain only HelloJava.java where as mybranch will contain HelloJava.java & HiJava.java
  - Launch below commands on GitBash
    - git checkout main
    - git merge mybranch
    - git push origin main
  - Go to GitHub web repo & refresh the repo page, main branch will contain HelloJava.java & HiJava.java
  - After merging, if you want to delete your branch then run below commands on GitBash
    - git branch -d mybranch
    - git push origin --delete mybranch
  - Go to GitHub web repo & refresh the repo page, now mybranch will be deleted from there

DOu DevOps united

HO-1.3.3(HO-0)
Exercise Solution - Demonstrate how to apply the main features of a configuration management tool: check-in, check-out, merge, conflict resolution, branching

- Resolve Conflict:

  Note: By deafult GitHub creates main branch but we are creating master branch to execute this exercise. For other tools(e.g. bitbucket etc) default branch naming may vary

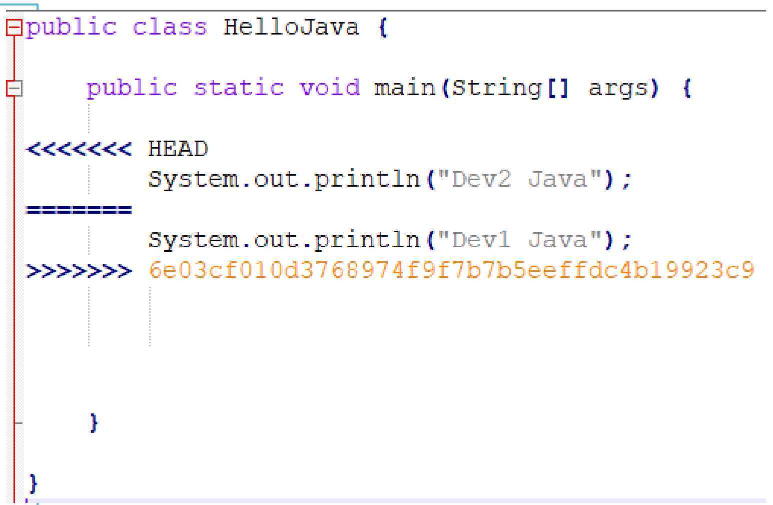
  - Create new repository(e.g. resolveconflict) on github web (e.g. by using "uploading an existing file" option) having HelloJava.java file in it
  - Create new branch(master) in resolveconflict repo
    - Click "main" button on resolveconflict repo page
    - Type **master** branch in "Find or create a branch section"
    - Click "Create branch master from main" link
  - Create 2 folders(e.g. dev1, dev2) on local machine in **git** folder of C drive(You can keep them in any drive)

HO-1.3.3(HO-0)
Exercise Solution - Demonstrate how to apply the main features of a configuration management tool: check-in, check-out, merge, conflict resolution, branching

- Resolve Conflict(Cont…):
  - Launch cmd and run below commands
    - cd C:\git\dev1
    - git init
      Note: .git file will be created in dev1 folder
    - git remote add origin https://github.com/umangsaltuniv/resolveconflict.git
      Note: Use your repo link that you have created in this exercise in above command
    - git pull origin master
  - Launch another cmd and run below commands
    - cd C:\git\dev2
    - git init
      Note: .git file will be created in dev2 folder
    - git remote add origin https://github.com/umangsaltuniv/resolveconflict.git
      Note: Use your repo link that you have created in this exercise in above command
    - git pull origin master

HO-1.3.3(HO-0)
Exercise Solution - Demonstrate how to apply the main features of a configuration management tool: check-in, check-out, merge, conflict resolution, branching

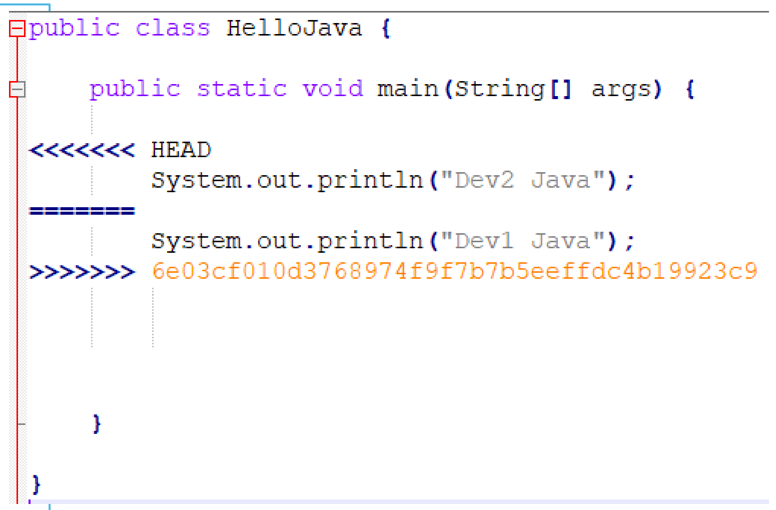
- Resolve Conflict(Cont…):
  - **For dev1:**
  - Modify the code for dev1 in HelloJava.java file, save, close the file and run below commands from dev1 cmd
    - git status
    - git add <filnename.java> (e.g. git add HelloJava.java)
    - git commit -m "dev1commit"
    - git push origin master
  - Go to github web, refresh the appropriate repo page and see the code changes there
  - **For dev2:**
  - Modify the code for dev2 in HelloJava.java file, save, close the file and run below commands from dev2 cmd
    - git status
    - git add <filnename.java> (e.g. git add HelloJava.java)
    - git commit -m "dev2commit"
    - git push origin master
  - Conflict error will be displayed



```
public class HelloJava {

    public static void main(String[] args) {

<<<<<<< HEAD
        System.out.println("Dev2 Java");
=======
        System.out.println("Dev1 Java");
>>>>>>> 6e03cf010d3768974f9f7b7b5eeffdc4b19923c9



    }

}
```

HO-1.3.3(HO-0)
Exercise Solution - Demonstrate how to apply the main features of a configuration management tool: check-in, check-out, merge, conflict resolution, branching

- Resolve Conflict(Cont…):
  - Resolve the Conflict
    - git pull origin master
    - git diff <filename.java>
  - open the HelloJava.java file from dev2 folder, do the changes manually(remove HEAD and other tags, Syso Dev1 code line), save & close the file
    - git add <filename>
    - git commit -m "conflictresolved"
    - git push origin master
  - Go to github web to see changes(Syso Dev2 code line will be displayed)
  - Go to github web, refresh the appropriate repo page and see the code changes there(Syso Dev2 code line will be displayed in HelloJava.java file)



```
public class HelloJava {

    public static void main(String[] args) {

<<<<<<< HEAD
        System.out.println("Dev2 Java");
=======
        System.out.println("Dev1 Java");
>>>>>>> 6e03cf010d3768974f9f7b7b5eeffdc4b19923c9



    }

}
```