

KillTest

높은 품질, 더 나은 서비스



연습 문제

<https://www.killtest.kr>

1년간 무료 업데이트 제공

Exam : EX374

Title : Red Hat Certified Specialist
in Developing Automation
with Ansible Automation
Platform exam

Version : DEMO

1.You are given access to a Git repository with the URL `https://github.com/example/repo.git`. Clone the repository to your local machine in the directory `/home/user/projects`.

Answer:

```
cd /home/user/projects
git clone https://github.com/example/repo.git
```

Explanation:

Cloning a repository retrieves all files, branches, and history to your local system. The `git clone` command initiates this process by downloading the specified repository.

2.After cloning a repository, you need to create a new branch named `feature-update`. Create the branch and switch to it.

Answer:

```
cd /home/user/projects/repo
git branch feature-update
git checkout feature-update
```

Explanation:

The `git branch` command creates a new branch, while `git checkout` switches to it, allowing changes to be made without affecting the main branch.

3.Make a file named `config.txt` in the cloned repository, add sample configuration details, and commit the file with the message "Add initial config".

Answer:

```
cd /home/user/projects/repo
echo "server=127.0.0.1" > config.txt
git add config.txt
git commit -m "Add initial config"
```

Explanation:

New files must be staged with `git add` before committing. The `git commit` command saves the changes with a descriptive message.

4.You realize you need to modify the `config.txt` file to add a new configuration `port=8080`. Update the file and commit with the message "Add port configuration".

Answer:

```
echo "port=8080" >> config.txt
git add config.txt
git commit -m "Add port configuration"
```

Explanation:

Appending to a file and committing the changes ensures version control captures the updated state, allowing tracking of incremental modifications.

5.You need to check the history of commits in the repository. Display a concise log of the last 3 commits.

Answer:

```
git log --oneline -n 3
```

Explanation:

`git log --oneline` provides a simplified view of commits, while `-n 3` limits the output to the last three entries, helping analyze recent changes.