Interview Tips Day 2: Github for DevOps - Some Common Questions with Basic to advance Commands

1. What is git & github?

Git :

Git is a distributed version control system used for tracking changes in source code during software development. It allows multiple developers to collaborate, manage projects, and maintain a history of code changes.

GitHub:

GitHub is a web-based platform built around Git. It hosts Git repositories in the cloud, offering tools for collaboration, code review, issue tracking, and project management for developers and teams.

2. Why should I use github for DevOps?

Version Control: GitHub facilitates version control, enabling teams to track changes, manage different versions of code, and collaborate efficiently.

1. **Collaboration:** It fosters teamwork by providing a centralized platform for developers to work together on projects, merge changes, and review code through pull requests.
2. **CI/CD Integration:** GitHub seamlessly integrates with Continuous Integration/Continuous Deployment (CI/CD) tools, streamlining the software development lifecycle.
3. **Issue Tracking and Management:** It offers robust issue tracking tools, enabling teams to report, manage, and resolve bugs or feature requests effectively.
4. **Community and Open Source:** GitHub is a hub for open-source projects, fostering a community where developers can contribute, learn, and share code, accelerating innovation and learning in DevOps practices.

3. What are the commands used in GitHub for DevOps?

1. **git add**: Adds changes in the working directory to the staging area.

   Example:

```
git add <file_name>
```

2. **git commit**: Records changes to the repository with a commit message.

   Example:

   ```
   git commit -m "Commit message describing changes"
   ```

3. **git push**: Uploads local repository content to a remote repository.

   Example:

   ```
   git push origin <branch_name>
   ```

4. **git pull**: Fetches changes from a remote repository and merges them into the local branch.

   Example:

   ```
   git pull origin <branch_name>
   ```

5. **git branch**: Lists, creates, or deletes branches within a repository.

   Example:

   ```
   phpCopy code
   git branch
   git branch <new_branch>
   git branch -d <branch_name>
   ```

6. **git merge**: Merges changes from one branch into another branch.

   Example:

   ```
   git checkout <branch_name>
   git merge <source_branch>
   ```

7. **GitHub Actions**: Used to automate workflows within your GitHub repository. YAML-based configuration files define the actions triggered by events.

   Example:

   ```
   name: CI/CD Pipeline
   on:
     push:
       branches: [main]
     pull_request:
       branches: [main]

   jobs:
   ```

```
  build:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      # Add more steps for building, testing, and deploying code
      # Example:
      # - name: Build
      #   run: |
      #     npm install
      #     npm run build
      # - name: Test
      #   run: npm test
      # - name: Deploy
      #   run: |
      #     ssh user@server 'bash -s' < deploy_script.sh
```

8 **`git clone`**: Clones a repository onto your local machine.

Example:

```
git clone <repository_URL>
```

4. How do you use Git branches in a CI/CD pipeline for a complex app?

Example: Imagine our app is like a cake being baked. Each baker (developer) works on a different layer of the cake (branch) with its own flavor. When all layers are ready, they're stacked (merged) together for icing and decorations (testing) before serving (deploying).

5. Explain the role of GitHub webhooks in a CI/CD workflow.

Example: Think of webhooks as messages sent by the oven timer. Whenever the cake batter (code changes) is ready for baking (pushed to the repository), the oven (CI server) automatically starts baking (running tests).

6. What are Blue-Green deployments, and how can GitHub Actions help?

Example: Imagine having two playgrounds, one where kids are playing (Blue) and another that's empty (Green). With GitHub Actions, we can smoothly switch the "play" sign (route traffic) from the busy playground to the empty one after safety checks (tests) are done.

7. How to secure GitHub repos and CI/CD pipelines?

Example: Securing GitHub repos is like locking valuable items in a safe. We set special locks (access controls) and codes (two-factor authentication) to keep things safe. For pipelines, we keep secret ingredients hidden (encrypted) while the cake is being made.

8. Why use GitHub Actions over other CI/CD tools?

Example: It's like having a magical recipe book that not only guides cooking but also does the dishes. With GitHub Actions, our recipe (YAML file) not only cooks the cake (builds and tests) but also cleans up the kitchen (deployment) seamlessly.

9. What is a pull request, and how does it benefit the CI/CD process on GitHub?

Example: A pull request is like asking for a second opinion before baking a cake. Before mixing new flavors (merging code), we ask a friend (CI/CD tests) to taste-test (check) and make sure the cake is delicious (code is good) for everyone.

10. Explain the concept of GitHub Actions and their role in automating workflows.

Example: Think of GitHub Actions as magical kitchen fairies. They follow our recipe (workflow) step-by-step, helping with mixing ingredients (builds), taste-testing (testing), and serving the cake (deployment) automatically.

11. How can GitHub facilitate collaboration among team members in a DevOps environment?

Example: GitHub is like a giant whiteboard where everyone shares their ideas and helps draw the perfect cake recipe. We all add ingredients (code changes), suggest flavors (discuss issues), and taste-test (review) together before baking.

12. Discuss the importance of documentation within GitHub repositories for DevOps teams.

Example: Documenting in GitHub is like leaving helpful notes in a cookbook. We write down secret spice mixes (configuration steps) and cake decorating tips (best practices) so anyone (new team members) can bake a perfect cake (deployments) easily.