1. Design and implement processes and communications
  └ **1.1 Design and implement traceability and flow of work**
    └ **1.1.1 Design and implement GitHub Flow**

1. What is GitHub Flow and how does it differ from Git Flow?
2. What are the steps of a standard GitHub Flow?
3. How do you implement branch protections in GitHub?
4. How are pull request (PR) reviews enforced in GitHub repositories?
5. What are status checks and how are they configured in GitHub Actions?
6. How can you automate deployments with GitHub Actions during GitHub Flow?
7. How do environment protection rules function in GitHub workflows?
8. How do you manage secrets and credentials in GitHub Actions?
9. What are best practices for committing code in a GitHub Flow process?
10. How do you roll back changes in GitHub Flow?

---

1. What is GitHub Flow and how does it differ from Git Flow?
GitHub Flow is a lightweight, trunk-based workflow focused on continuous delivery using a main branch and short-lived feature branches. Unlike Git Flow, it avoids long-lived branches like develop or release.

---

2. What are the steps of a standard GitHub Flow?
1. Create a feature branch from main
2. Make and commit changes locally
3. Push the branch to GitHub
4. Open a pull request (PR)
5. Review, approve, and merge the PR
6. Automatically or manually deploy after merge

---

3. How do you implement branch protections in GitHub?
In repository settings > Branches:
- Require PR reviews before merging
- Require status checks to pass
- Restrict who can push to the branch

---

4. How are pull request (PR) reviews enforced in GitHub repositories?
Enable "*Require pull request reviews before merging*" under branch protection rules. You can specify required reviewers and enforce code owner review.

---

5. What are status checks and how are they configured in GitHub Actions?
Status checks are CI workflows (e.g., build/test) that must pass before a PR can be merged. Define them in .github/workflows/*.yml and mark them as required in branch protection settings.

---

6. How can you automate deployments with GitHub Actions during GitHub Flow?
Use workflows triggered on push or pull_request events. Use jobs and steps to deploy to environments like Azure or AWS. Example:
on: push to main → deploy job runs → production environment

7. How do environment protection rules function in GitHub workflows?
They restrict deployment via GitHub Actions. Configure environments with:

- Required reviewers
- Wait timer
- Custom deployment branches

---

8. How do you manage secrets and credentials in GitHub Actions?
Store secrets in *Repository Settings > Secrets and variables > Actions*. Access them using ${{ secrets.SECRET_NAME }} in workflow YAML.

---

9. What are best practices for committing code in a GitHub Flow process?

- Use atomic commits (one change per commit)
- Write clear commit messages
- Rebase before merging
- Avoid pushing directly to main

---

10. How do you roll back changes in GitHub Flow?
Use git revert to create a new commit that undoes changes. Push the revert commit and merge via PR, triggering redeployment if needed.