1. Design and implement processes and communications
└─ 1.1 Design and implement traceability and flow of work
└─ 1.1.2 Design and implement feedback cycles

1. What are feedback cycles in DevOps and why are they critical?
2. How do you configure notifications in Azure DevOps pipelines?
3. How do you set up GitHub issue automation for feedback tracking?
4. What integrations are available between GitHub issues and Azure Boards?
5. How do you configure branch protections with required status checks for feedback?
6. How do you use GitHub Actions to automate issue creation on failure?
7. What are the best practices for actionable notification configuration?
8. How can you integrate feedback from Azure Monitor into GitHub or Azure DevOps?
9. How do you set up pull request templates to improve developer feedback cycles?
10. How do service hooks enable automated feedback loops across tools?

---

1. **What are feedback cycles in DevOps and why are they critical?**
Feedback cycles provide continuous insights (build status, test results, deployment issues) to developers, enabling rapid detection and correction, reducing lead time and improving quality.

---

2. **How do you configure notifications in Azure DevOps pipelines?**
Go to *Project Settings* → *Notifications* → *Create Subscription* → *Select event* (e.g., build failure) → *Define filters* (e.g., pipeline name) → *Set recipients* (users or groups).

---

3. **How do you set up GitHub issue automation for feedback tracking?**
Use GitHub Actions with *actions/create-issue* or similar to automatically open issues on failures, e.g., test failures or workflow errors, linking to logs or artifacts.

---

4. **What integrations are available between GitHub issues and Azure Boards?**
Install *the Azure Boards GitHub app* → *Link repos* → *Use linked commits, PRs, and issues to automatically update Azure Boards work items* for traceability.

---

5. **How do you configure branch protections with required status checks for feedback?**
In GitHub → Settings → Branches → Add branch protection rule → Enable required status checks (e.g., CI pipeline) → Ensure passing checks before merging.

---

6. **How do you use GitHub Actions to automate issue creation on failure?**
In a workflow, use a *jobs.<job>.if: failure() condition* → *Add a step* using a GitHub token to call the REST API or a prebuilt action like peter-evans/create-issue.

---

7. **What are the best practices for actionable notification configuration?**
Notify only relevant users/groups, include clear context (links to builds, commits), avoid noise (e.g., no "success" spam), and prioritize critical failures.

---

8. **How can you integrate feedback from Azure Monitor into GitHub or Azure DevOps?**
Use *Azure Monitor Action Groups* → *Webhooks, Logic Apps, or Azure Functions* → *Post alerts as GitHub issues or Azure DevOps work items* automatically.

9. **How do you set up pull request templates to improve developer feedback cycles?**
In .github/PULL_REQUEST_TEMPLATE.md, define required sections (e.g., linked issues, testing done) to standardize information and reduce reviewer cycles.

10. **How do service hooks enable automated feedback loops across tools?**
Service hooks in Azure DevOps or GitHub connect events (build failures, PR updates) to external systems like Slack, Teams, or custom webhooks, triggering automated feedback.