3. Build and release pipelines

└ 3.2 Design and implement pipelines

└ 3.2.3 Integration of GitHub repos with Azure Pipelines

---

1. What are the prerequisites for integrating GitHub repositories with Azure Pipelines?
2. How do you connect a GitHub repository to an Azure Pipeline?
3. What authentication methods are supported for GitHub-Azure Pipelines integration?
4. How do you configure pipeline triggers for GitHub repository events?
5. How do you set up build validation for GitHub pull requests in Azure Pipelines?
6. How can you use YAML pipelines with GitHub repositories?
7. What permissions are required for Azure Pipelines to access a GitHub repository?
8. How are service connections managed and secured for GitHub integration?
9. What are best practices for handling secrets in GitHub-Azure Pipelines workflows?
10. How do you monitor and troubleshoot integration issues between GitHub and Azure Pipelines?

---

1. What are the prerequisites for integrating GitHub repositories with Azure Pipelines?

- An *Azure DevOps* organization and project
- A *GitHub* repository (personal or organization)
- Sufficient permissions on both platforms (repo admin or owner, Azure Pipelines Contributor)
- *Azure Pipelines* app authorized in *GitHub* (installed as needed)

---

2. How do you connect a GitHub repository to an Azure Pipeline?

- In Azure DevOps, create a new pipeline
- Choose GitHub as the code source
- Authorize Azure Pipelines to access your GitHub account
- Select the repository and configure the pipeline (YAML or classic editor)

---

3. What authentication methods are supported for GitHub-Azure Pipelines integration?

- *OAuth* via Azure Pipelines GitHub App (recommended)
- Personal Access Token (PAT) for classic connections
- GitHub Apps for organization-wide integration

---

4. How do you configure pipeline triggers for GitHub repository events?

- In the YAML pipeline file, use the trigger: keyword for branch updates and pr: for pull requests
- Example:

```
trigger:
  branches:
    include:
      - main
pr:
  branches:
    include:
      - main
```

---

5. How do you set up build validation for GitHub pull requests in Azure Pipelines?

- Enable branch protection in *GitHub*
- Add *Azure Pipelines* as a required status check for PRs
- Configure the pipeline to run on pull requests using the `pr:` trigger in YAML

---

6. How can you use YAML pipelines with GitHub repositories?

- Place an `azure-pipelines.yml` file in the root of the GitHub repo
- Configure the pipeline in *Azure DevOps* to use this YAML file
- Updates to the YAML file in *GitHub* automatically update the pipeline definition

---

7. What permissions are required for Azure Pipelines to access a GitHub repository?

- Repo-level access for the *Azure Pipelines* app
- At minimum, read code and read/write for status checks
- Admin access may be needed for initial setup and webhook management

---

8. How are service connections managed and secured for GitHub integration?

- Use the *Azure Pipelines GitHub App* for secure, managed integration
- Service connections are created in *Project Settings > Service connections*
- Use least privilege and restrict access to pipelines needing the connection

---

9. What are best practices for handling secrets in GitHub-Azure Pipelines workflows?

- Store secrets in *Azure Pipelines* variable groups or *GitHub Secrets*
- Reference secrets in pipeline YAML using secure variables
- Never hard-code secrets in repository or pipeline files

---

10. How do you monitor and troubleshoot integration issues between GitHub and Azure Pipelines?

- Review pipeline run logs and error messages in *Azure DevOps*
- Check *GitHub* repository “Actions” and “Security & analysis” tabs
- Validate webhooks and service connection health in *Azure DevOps* project settings
- Check permission scopes and reauthorize the *Azure Pipelines GitHub App* if needed