**3. Build and release pipelines**

**└ 3.6 Infrastructure as Code (IaC)**

**└ 3.6.1 Define IaC strategy: Source Control, Automation**

---

1. What is Infrastructure as Code (IaC) and why is it important in DevOps?
2. What are the key principles of a successful IaC strategy?
3. Which tools are commonly used for IaC in Azure and GitHub-based environments?
4. How should IaC code be managed in source control?
5. What are best practices for organizing IaC repositories?
6. How can you automate testing and validation of IaC before deployment?
7. What are the main automation options for deploying IaC in Azure DevOps and GitHub Actions?
8. How do you manage secrets and sensitive configuration in IaC automation?
9. What are recommended practices for versioning and tracking changes in IaC?
10. How can you ensure repeatable and idempotent deployments with IaC?

---

**1. What is Infrastructure as Code (IaC) and why is it important in DevOps?**

- IaC is the practice of managing and provisioning infrastructure using code files, enabling automation, repeatability, and consistency.
- It eliminates manual configuration drift and supports rapid, reliable delivery.

---

2. **What are the key principles of a successful IaC strategy?**

Key principles:

- version everything in source control,
- automate validation/deployment,
- use modular and reusable code,
- ensure idempotency,
- and follow DRY (Don't Repeat Yourself) practices.

---

3. **Which tools are commonly used for IaC in Azure and GitHub-based environments?**

Popular IaC tools:

- *ARM* Templates,
- *Bicep*,
- *Terraform*,
- *Azure CLI*,
- *Azure PowerShell*,
- and *Ansible*.

For automation:

- *Azure DevOps* Pipelines
- and *GitHub Actions*.

---

4. **How should IaC code be managed in source control?**

- Store all IaC code in a versioned repository (e.g., Git).
- Use branching strategies,
- enforce code reviews,
- enable traceability,
- and restrict direct changes to main branches.

5. **What are best practices for organizing IaC repositories?**
   - Structure repositories by environment, service, or module.
   - Use directories for dev, test, prod.
   - Keep sensitive values out of code and use templates/modules for reusability.

---

6. **How can you automate testing and validation of IaC before deployment?**
   - Set up pipeline stages to lint, syntax check, and validate templates.
   - Use test deployments in isolated environments before production rollout.

---

7. **What are the main automation options for deploying IaC in Azure DevOps and GitHub Actions?**
   - Use *Azure Pipelines* or *GitHub Actions* to trigger IaC deployments on push, PR, or schedule.
   - Integrate approvals, environment checks, and rollback mechanisms.

---

8. **How do you manage secrets and sensitive configuration in IaC automation?**
   - Store secrets in *Azure Key Vault* or *GitHub Secrets*.
   - Reference them securely in pipeline definitions.
   - Never hard-code secrets in source or IaC files.

---

9. **What are recommended practices for versioning and tracking changes in IaC?**
   - Tag releases,
   - use PRs for change review,
   - link commits to work items,
   - and document change history.
   - Use versioned modules
   - and maintain a clear changelog.

---

10. **How can you ensure repeatable and idempotent deployments with IaC?**
Write templates and scripts to be idempotent, meaning running them multiple times produces the same result. Use declarative syntax (e.g., *Bicep*, *Terraform*) and avoid manual post-deployment steps.