3. Build and release pipelines

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

└ **3.6.3 Azure Deployment Environments (Self-Deployment)**

---

1. What are Azure Deployment Environments and what use case do they solve?
2. How do Azure Deployment Environments support developer self-service?
3. What are environment definitions and how are they created?
4. How do you configure projects and environments for team use?
5. How is access controlled in Azure Deployment Environments?
6. What IaC tools are supported in environment definitions?
7. How do environment templates integrate with source control?
8. How do you automate cleanup and lifecycle of deployment environments?
9. How are cost controls and quotas applied to environments?
10. What are best practices for using Azure Deployment Environments in enterprise DevOps?

---

1. **What are Azure Deployment Environments and what use case do they solve?**

They provide on-demand, consistent environments (dev/test/staging) for developers using predefined IaC templates, reducing wait times and manual provisioning.

---

2. **How do Azure Deployment Environments support developer self-service?**

Developers can deploy approved templates directly from a portal or CLI without needing contributor rights on subscriptions or waiting for central IT provisioning.

---

3. **What are environment definitions and how are they created?**

They are IaC templates (*Bicep* or *ARM*) stored in Git repositories, defining the infrastructure for an environment. Created and managed via *Azure* portal or CLI.

---

4. **How do you configure projects and environments for team use?**

- Admins create projects, link *GitHub* repos with definitions, and assign users or groups.
- Developers then select and deploy environments under that project.

---

5. **How is access controlled in Azure Deployment Environments?**

Using *Azure* RBAC at the project level—roles like Environment User or Project Admin control who can deploy, manage, or view environments.

---

6. **What IaC tools are supported in environment definitions?**

Bicep and ARM templates are supported. Terraform is currently not supported directly within Azure Deployment Environments as of June 2025.

---

7. **How do environment templates integrate with source control?**

- Templates are stored in *GitHub* (public or private repos).
- The system pulls definitions directly from branches specified during environment setup.

---

8. **How do you automate cleanup and lifecycle of deployment environments?**

Use environment expiration policies and lifecycle hooks (e.g., cleanup scripts) defined in the template or enforced via policy-as-code in *Azure*.

9. **How are cost controls and quotas applied to environments?**

- Set resource quotas,
- tag policies,
- and auto-expiration on environments.
- *Azure Cost Management* can be used to track and limit spend per environment.

---

10. What are best practices for using Azure Deployment Environments in enterprise DevOps?

- Use centralized template governance,
- automate validation,
- enforce least-privilege RBAC,
- tag all resources,
- and integrate environment cleanup into CI/CD workflows.