3. Build and release pipelines

└ **3.5 Implement deployment solutions**

└ **3.5.2 Deploy containers, binaries, scripts**

---

1. What are the main deployment options for containers in Azure DevOps?
2. How do you deploy Docker containers to Azure Kubernetes Service (AKS) using a pipeline?
3. How do you publish and deploy application binaries using Azure Pipelines?
4. What are the typical steps to deploy a script-based application in Azure DevOps?
5. How can deployment scripts be securely managed and executed in pipelines?
6. What tasks are available for deploying to Azure Web Apps from Azure DevOps?
7. How do you automate image versioning and tagging during container deployment?
8. What is the difference between classic and YAML-based deployment pipelines for these scenarios?
9. How do you handle rollback for failed deployments of containers, binaries, or scripts?
10. What are best practices for monitoring and validating deployments?

---

1. **What are the main deployment options for containers in Azure DevOps?**

You can deploy containers to

- Azure Kubernetes Service (AKS),
- Azure Container Instances,
- Azure App Service for Containers,
- or custom environments using built-in *Azure DevOps* tasks and service connections.

---

2. **How do you deploy Docker containers to Azure Kubernetes Service (AKS) using a pipeline?**

Use *Azure Pipelines* tasks: build and push image with *Docker* or *Azure CLI*, then use the KubernetesManifest or kubectl task to apply *Kubernetes* manifests and deploy to AKS.

---

3. **How do you publish and deploy application binaries using Azure Pipelines?**

Build binaries using build tasks, publish them as pipeline artifacts, then deploy using tasks like AzureWebApp or AzureFileCopy or custom script tasks to the desired target.

---

4. **What are the typical steps to deploy a script-based application in Azure DevOps?**

Store scripts in source control, reference them in pipeline YAML or classic release, use tasks like Bash PowerShell or Command Line to execute scripts, and manage outputs or errors accordingly.

---

5. **How can deployment scripts be securely managed and executed in pipelines?**

Store scripts in versioned source control. Store secrets/credentials in secure pipeline variables or *Azure Key Vault*. Never hard-code secrets in scripts; access them via secure variables or *Key Vault* references.

---

6. **What tasks are available for deploying to Azure Web Apps from Azure DevOps?**

- AzureWebApp task for code or binaries
- AzureWebAppContainer for *Docker* images
- Azure App Service Deploy for deployment slots and multi-phase scenarios

7. **How do you automate image versioning and tagging during container deployment?**
In your pipeline,

- use build IDs,
- commit hashes,
- or semantic versioning in the *Docker* tag step.

Automate tagging in the Docker or Azure CLI tasks before pushing images to a registry.

---

8. **What is the difference between classic and YAML-based deployment pipelines for these scenarios?**

- YAML pipelines provide pipeline-as-code, versioning, templates, and better reuse.
- Classic pipelines are GUI-based, less portable, and harder to manage for complex deployments.

---

9. **How do you handle rollback for failed deployments of containers, binaries, or scripts?**
Use pipeline stages to deploy to slots or canary environments. For containers, roll back by redeploying a previous image tag. For scripts/binaries, restore prior artifacts or use automated pipeline rollback steps.

---

11. **What are best practices for monitoring and validating deployments?**

- Enable deployment logs,
- configure health probes,
- set up *Application Insights*,
- and use release gates or post-deployment tests to validate success before full rollout.