

## Develop Azure Compute Solutions

### └ 1.1 Implement Containerized Solutions

#### └ 1.1.4 Create solutions by using Azure Container Apps

1. What is Azure Container Apps and when should it be used over AKS or App Services?
2. What components define an Azure Container App?
3. How do revisions work in Azure Container Apps?
4. What are the prerequisites for deploying a container to Azure Container Apps?
5. How do you deploy a container from Azure Container Registry using Azure CLI?
6. How is a YAML manifest used to deploy a container app?
7. How do you configure ingress and expose ports in Azure Container Apps?
8. How do you configure authentication for private container registries?
9. How are environment variables added to a container app?
10. How are secrets stored and injected into container apps?
11. What is KEDA and how does it apply to Azure Container Apps?
12. What scaling rules are supported in Azure Container Apps?
13. How do you configure HTTP-based autoscaling?
14. How do you configure scaling based on Azure Service Bus or Azure Queue Storage?
15. What are minReplicas and maxReplicas and how are they configured?
16. How does revision mode affect app behavior in Azure Container Apps?
17. How is traffic splitting configured across revisions?
18. How do you perform A/B testing using revisions?
19. How do you roll back to a previous revision?
20. What is Dapr and how is it used with Azure Container Apps?
21. What Dapr capabilities are supported in Azure Container Apps?
22. How do you enable and configure the Dapr sidecar?
23. How do container apps communicate using Dapr?
24. How is state management and pub/sub handled with Dapr?
25. What monitoring and logging features are built into Azure Container Apps?
26. How do you enable and access diagnostics logs?
27. How do you view application logs and container output?
28. What tools can be used to troubleshoot container app issues?
29. How does integration with Azure Monitor and Log Analytics work?
30. How do you integrate Azure Container Apps with Event Grid or Service Bus?
31. How do you connect a container app to Azure Storage queues?
32. How do you securely access Azure services from a container app?
33. How do you use managed identities in Azure Container Apps?
34. What role does VNET integration play in accessing private resources?

---

### 1. What is Azure Container Apps and when should it be used over AKS or App Services?

Azure Container Apps is a fully managed serverless container service for microservices, APIs, and background processing. Use it when:

- You need event-driven or HTTP-based workloads
- You prefer serverless scaling (including scale to zero)
- You want built-in Dapr and KEDA support without managing orchestration

Prefer AKS for orchestration or App Services for traditional web apps or minimal container needs.

---

## 2. What components define an Azure Container App?

- Container App: The deployed app instance
  - Environment: A shared context for apps (networking, logging)
  - Revision: An immutable version of the app
  - Ingress: Controls public/private HTTP access
  - Scaling Rules: Define autoscaling behavior (HTTP, KEDA, etc.)
- 

## 3. How do revisions work in Azure Container Apps?

- Each deployment creates a new immutable revision
  - Revisions can run concurrently
  - Traffic can be split between revisions
  - You can pin a revision or roll back
  - Revision mode can be single (default) or multiple
- 

## 4. What are the prerequisites for deploying a container to Azure Container Apps?

- A container image in ACR or public registry
  - A Container Apps environment
  - Azure CLI with the containerapp extension installed
  - App image must expose the correct HTTP port.
- 

## 5. How do you deploy a container from Azure Container Registry using Azure CLI?

```
az containerapp create \  
  --name myapp \  
  --resource-group myrg \  
  --environment myenv \  
  --image myacr.azurecr.io/myimage:tag \  
  --target-port 80 \  
  --ingress external \  
  --registry-server myacr.azurecr.io \  
  --registry-username <username> \  
  --registry-password <password>
```

---

## 6. How is a YAML manifest used to deploy a container app?

Define app configuration in a .yaml file (image, ports, scaling, secrets, etc.)

- Deploy using:  
az containerapp create --resource-group myrg --name myapp --yaml app.yaml
- 

## 7. How do you configure ingress and expose ports in Azure Container Apps?

- Use --ingress external or internal in CLI or ingress: block in YAML
  - Set targetPort to match container's exposed port
  - Public ingress automatically provisions HTTPS endpoint
- 

## 8. How do you configure authentication for private container registries?

- Use --registry-username and --registry-password in CLI
- In YAML:  
registryCredentials:  
 - server: myacr.azurecr.io  
 username: <username>  
 passwordSecretRef: acr-password
- Store password as a secret and reference it

### 9. How are environment variables added to a container app?

- In CLI:  
`--env-vars VAR1=value1 VAR2=value2`
  - In YAML:  
env:
    - name: VAR1  
value: value1
    - name: VAR2  
value: value2
- 

### 10. How are secrets stored and injected into container apps?

- Define secrets in CLI  
`--secrets key1=value1 key2=value2`
  - or YAML:  
secrets:
    - name: key1  
value: value1
  - Reference in env vars:  
env:
    - name: SEC\_VAR  
secretRef: key1
- 

### 12. What is KEDA and how does it apply to Azure Container Apps?

KEDA (Kubernetes Event-driven Autoscaler) enables event-based scaling. In Azure Container Apps, it's integrated to scale apps based on metrics like:

- HTTP traffic
  - Queue length (e.g., Service Bus, Storage Queues)
  - Custom metrics
- 

### 13. What scaling rules are supported in Azure Container Apps?

- HTTP request concurrency
  - CPU utilization
  - KEDA-based triggers (e.g., Azure Service Bus, RabbitMQ, Redis, Kafka)
  - Cron-based schedules
- 

### 14. How do you configure HTTP-based autoscaling?

In YAML:

```
scale:
  rules:
    - name: http-scaler
      http:
        concurrentRequests: 50
```

App will scale based on the number of concurrent HTTP requests.

---

### 15. What are minReplicas and maxReplicas and how are they configured?

- minReplicas: minimum number of app instances
  - maxReplicas: cap on autoscaling
- In YAML:
- ```
scale:
  minReplicas: 1
  maxReplicas: 10
```
-

## 16. How do you configure scaling based on Azure Service Bus or Azure Queue Storage?

Define a KEDA trigger in YAML:

```
scale:
  rules:
    - name: sb-scaler
      azureServiceBus:
        queueName: myqueue
        connection: sb-connection
        messageCount: 100
```

\*"connection" references a secret holding the Service Bus connection string

---

## 17. How does revision mode affect app behavior in Azure Container Apps?

- Single revision mode (default): only one revision is active; new deployments kill the previous
  - Multiple revision mode: multiple revisions can run concurrently; useful for traffic splitting
  - Set via CLI or YAML:  
revisionMode: multiple
- 

## 18. How is traffic splitting configured across revisions?

Assign percentage of traffic to each revision

- In CLI:  
az containerapp revision set-traffic-split \  
--name myapp \  
--resource-group myrg \  
--revision-weight revisionA=80 revisionB=20
  - In YAML:  
traffic:  
- revisionName: revisionA  
weight: 80  
- revisionName: revisionB  
weight: 20
- 

## 19. How do you perform A/B testing using revisions?

- Deploy a new revision in multiple revision mode
  - Split traffic between revisions (e.g., 90/10)
  - Monitor metrics and logs for both
  - Adjust traffic weights or rollback based on results
- 

## 20. How do you roll back to a previous revision?

- Set traffic weight to 100% for the target revision
  - Optionally disable the newer revision
  - CLI:  
az containerapp revision set-traffic-split --name myapp --revision-weight oldrev=100
- 

## 21. What is Dapr and how is it used with Azure Container Apps?

Dapr (Distributed Application Runtime) provides building blocks for microservices (e.g., service discovery, state management). Azure Container Apps has built-in Dapr support. Enable by setting daprEnabled: true. No additional setup is required for the Dapr sidecar.

---

---

**22. What Dapr capabilities are supported in Azure Container Apps?**

- Service invocation over HTTP/gRPC
- State management (e.g., Redis, Cosmos DB)
- Pub/sub messaging
- Secrets integration
- Middleware and observability tools

Note: Components are defined via Dapr-compatible configuration files.

---

**23. How do you enable and configure the Dapr sidecar?**

In YAML:

```
dapr:
  enabled: true
  appld: myapp
  appPort: 80
```

---

**24. How do container apps communicate using Dapr?**

- Service A calls Service B via `http://<appld>.dapr`
  - Dapr handles service discovery and routing
  - Requires both apps to have `dapr.enabled: true` and unique `appld`
- 

**25. What monitoring and logging features are built into Azure Container Apps?**

- Integrated Log Streaming via Azure CLI
  - Application logs, revision logs, and system logs
  - Container stdout/stderr collection
  - Azure Monitor and Log Analytics integration for metrics and centralized logging
- 

**26. How do you enable and access diagnostics logs?**

- Enable diagnostics when creating the Container App Environment
- Logs are sent to Azure Monitor (Log Analytics workspace)
- Use Azure CLI:

```
az containerapp logs show --name myapp --resource-group myrg
```

---

**27. How do you view application logs and container output?**

Via Azure CLI:

```
az containerapp logs show --name myapp --follow
```

- Logs include stdout and stderr from the container
  - For historical logs, query via Log Analytics using Kusto Query Language (KQL)
- 

**28. What tools can be used to troubleshoot container app issues?**

- `az containerapp logs show` for live logs
  - Log Analytics queries for historical data
  - Metrics in Azure Monitor (CPU, memory, HTTP throughput)
  - Azure CLI/Portal for revision status and health
  - Re-deploy with `--debug` flag to get CLI diagnostics
- 

**29. How does integration with Azure Monitor and Log Analytics work?**

- When enabled, diagnostics are sent to a specified Log Analytics workspace
- Use KQL to query logs (e.g., `ContainerAppConsoleLogs_CL`)
- Metrics surface in Azure Monitor for alerting and dashboarding

### 30. How do Azure Container Apps integrate with Event Grid or Service Bus?

- Use KEDA triggers to scale based on Event Grid or Service Bus messages
  - Event Grid: typically triggers external logic that posts to app HTTP endpoint
  - Service Bus: KEDA listens and scales app based on queue/topic message count
  - Connection strings are passed as secrets and referenced in scaling rules
- 

### 31. How do you connect a container app to Azure Storage queues?

- Use KEDA with azureQueue scaler
  - Define queueName, connection, and queueLength threshold
  - Store Storage Account connection string as a secret and reference it in scaling config
  - App logic must poll the queue if not using an event trigger
- 

### 32. How do you securely access Azure services from a container app?

- Use **Managed Identity** to authenticate to Azure services like Key Vault, Storage, or SQL
  - Avoid hardcoding credentials
  - Access tokens are obtained via Azure SDK or HTTP call to IMDS endpoint
- 

### 33. How do you use managed identities in Azure Container Apps?

- Enable system-assigned or user-assigned identity at app level
  - Assign proper RBAC role to the identity
  - Access Azure services using Azure SDKs with default credential chain
  - Example (Azure SDK):  
from azure.identity import DefaultAzureCredential  
from azure.keyvault.secrets import SecretClient
- 

### 35. What role does VNET integration play in accessing private resources

Enables access to private endpoints, databases, or internal APIs

- Configure internal ingress and associate the Container Apps environment with a VNET
- Required for scenarios needing outbound traffic restrictions or private-only dependencies