AZ-204: Develop Azure Compute Solutions

→ Implement Containerized Solutions
→ Run Containers by Using Azure Container Instances (ACI)

- What is Azure Container Instances (ACI) and when should you use it?
- How do you create and deploy a container using ACI via Azure CLI?
- What are the key configuration parameters for az container create?
- How do you pull images from Azure Container Registry to ACI?
- How does managed identity authentication work with ACI?
- How do you assign and verify roles for ACI to access ACR?
- What are the options for exposing containers to the internet or VNETs in ACI?
- How do you mount Azure Files or secrets (Key Vault) into containers?
- How do you monitor logs, metrics, and container status in ACI?
- What are the restart policies and lifecycle options in ACI?
- How do you use YAML to define ACI deployments?
- What are common use cases and limitations of ACI?

What is Azure Container Instances (ACI) and when should you use it?
ACI is a serverless container platform allowing fast, isolated container runs without VM management.
Use cases:

- Short-lived jobs or batch processing
- Event-driven container execution
- Lightweight API hosting without orchestration overhead

How do you create and deploy a container using ACI via Azure CLI?

```
az container create \
        --resource-group <rg> \
        --name <container-name> \
        --image <image-name> \
        --cpu 1 \
        --memory 1 \
        --restart-policy OnFailure \
        --dns-name-label <unique-label> \
        --ports 80
```

This deploys a public-facing container running on port 80.

What are the key configuration parameters for az container create?
--image: Container image to run (e.g., from ACR or Docker Hub)
--cpu / --memory: Resource limits
--environment-variables: Inject app settings
--ports: Exposed ports
--dns-name-label: For public IP
--restart-policy: Options: Always, OnFailure, Never
--vnet and --subnet: Attach to virtual network
--secrets and --secrets-mount-path: Mount secrets

How do you pull images from Azure Container Registry to ACI?
ACI can pull private images from ACR by granting ACI's managed identity the **AcrPull** role on the ACR.
Ensure image format is:
```
<acr-name>.azurecr.io/<repository>:<tag>
```
Example ACI deployment with an image from ACR:
```
az container create \
  --name <container-name> \
  --resource-group <rg> \
  --image <acr-name>.azurecr.io/app:latest \
  --registry-login-server <acr-name>.azurecr.io \
  --assign-identity \
  --cpu 1 --memory 1
```

How does managed identity authentication work with ACI?
ACI supports **user-assigned** and **system-assigned managed identities**.
Steps:
1. Enable managed identity with --assign-identity.
2. Assign AcrPull role to the identity at ACR scope.
3. ACI uses this identity to authenticate and pull private images—no credentials needed.

How do you assign and verify roles for ACI to access ACR?
Use Azure CLI to assign roles:
```
az role assignment create \
  --assignee <principal-id> \
  --role AcrPull \
  --scope /subscriptions/<sub-id>/resourceGroups/<rg>/providers/Microsoft.ContainerRegistry/registries/<acr-name>
```
Verify with:
```
az role assignment list --assignee <principal-id> --scope <acr-resource-id>
```

What are the options for exposing containers to the internet or VNETs in ACI?
- **Public IP (default)**: Use --dns-name-label and --ports to expose over the internet.
- **Private IP (VNET)**: Use --vnet & --subnet to deploy into a virtual network for internal-only access.

ACI supports:
- Inbound public access
- Private IP in VNET (for secure inter-service traffic)
- No ingress (headless jobs)

How do you mount Azure Files or secrets (Key Vault) into containers?
Mount Azure Files:
```
az container create \
  --azure-file-volume-share-name <share> \
  --azure-file-volume-account-name <storage-account> \
  --azure-file-volume-account-key <key> \
  --azure-file-volume-mount-path /mnt/data
```
Mount Key Vault secrets:
Use --secrets and --secrets-mount-path:
```
az container create \
  --secrets key1=value1 key2=value2 \
  --secrets-mount-path /mnt/secrets
```

How do you monitor logs, metrics, and container status in ACI?

View logs:

```
az container logs --name <container-name> --resource-group <rg>
```

Get status:

```
az container show --name <container-name> --resource-group <rg> --query instanceView.state
```

What are the restart policies and lifecycle options in ACI?

Available values for --restart-policy:

- Always: Container restarts on exit (default).
- OnFailure: Restarts only on non-zero exit code.
- Never: One-time execution, used for jobs.

Lifecycle:

- No native job scheduling—combine with Logic Apps, Functions, or Event Grid for automation.
- ACI auto-deletes after manual az container delete or TTL implementation logic.

How do you use YAML to define ACI deployments?

Example aci.yaml:

```
apiVersion: 2018-10-01
location: eastus
name: mycontainer
properties:
  containers:
    - name: myapp
      properties:
        image: myacr.azurecr.io/myapp:latest
        resources:
          requests:
            cpu: 1
            memoryInGb: 1.5
        ports:
          - port: 80
  osType: Linux
  restartPolicy: OnFailure
  ipAddress:
    type: Public
    dnsNameLabel: mycontainerdemo
    ports:
      - port: 80
type: Microsoft.ContainerInstance/containerGroups
```

Deploy with:

```
az deployment group create \
  --resource-group <rg> \
  --template-file aci.yaml
```

What are common use cases and limitations of ACI?

Use Cases:

- Lightweight API/backend services
- Batch jobs
- Temporary compute (build/test)
- Event-driven processing

Limitations:

- No built-in autoscaling
- No service mesh or ingress controller
- Not suited for complex orchestration—use AKS instead