**Step 1: Define Variables in PowerShell**

# Unique suffix for names

$SUFFIX = [System.Guid]::NewGuid().ToString("N").Substring(0,8)

# Variables

$LOCATION = "eastus"

$RESOURCE\_GROUP = "devui-devsecops"

$STORAGE\_ACCOUNT = "storagedevui$SUFFIX"

$FUNCTION\_APP = "devuifunc$SUFFIX"

$PLAN = "DevuiLinuxPlan"

$CONTAINER\_INPUT = "sonarinput"

$CONTAINER\_OUTPUT = "summaries"

$APP\_INSIGHTS = "devuiappinsights"

$OPENAI\_RESOURCE = "openai-devui"

$DEPLOYMENT\_NAME = "gpt-4.1"

Step 2: Create Resource Group

az group create --name $RESOURCE\_GROUP --location $LOCATION

Step 3: Create Storage Account and Blob Containers

az storage account create `

--name $STORAGE\_ACCOUNT `

--resource-group $RESOURCE\_GROUP `

--location $LOCATION `

--sku Standard\_LRS

# Get connection string for future use (not strictly required for CLI)

$CONN\_STRING = az storage account show-connection-string `

--name $STORAGE\_ACCOUNT `

--resource-group $RESOURCE\_GROUP `

--query connectionString -o tsv

# Create blob containers

az storage container create --name $CONTAINER\_INPUT --account-name $STORAGE\_ACCOUNT --public-access off

az storage container create --name $CONTAINER\_OUTPUT --account-name $STORAGE\_ACCOUNT --public-access off

Step 4: Create Application Insights

az monitor app-insights component create `

--app $APP\_INSIGHTS `

--location $LOCATION `

--resource-group $RESOURCE\_GROUP `

--application-type web

Step 5: Create Function App and Configure Settings

# Create app service plan

az functionapp plan create `

--resource-group $RESOURCE\_GROUP `

--name $PLAN `

--location $LOCATION `

--number-of-workers 1 `

--sku B1 `

--is-linux

# Create function app

az functionapp create `

--name $FUNCTION\_APP `

--storage-account $STORAGE\_ACCOUNT `

--resource-group $RESOURCE\_GROUP `

--plan $PLAN `

--runtime python `

--runtime-version 3.11 `

--functions-version 4 `

--os-type Linux `

--app-insights $APP\_INSIGHTS

# Create openAI resource

az cognitiveservices account create `

--name $OPENAI\_RESOURCE `

--resource-group $RESOURCE\_GROUP `

--kind OpenAI `

--sku S0 `

--location $LOCATION `

--custom-domain $OPENAI\_RESOURCE `

--yes

After creating the OpenAI resource, deploy the GPT-4.1 model before proceeding to the next step.

# Set Function App environment variables

az functionapp config appsettings set `

--name $FUNCTION\_APP `

--resource-group $RESOURCE\_GROUP `

--settings `

OPENAI\_API\_BASE="https://openai-devui.openai.azure.com/" `

OPENAI\_API\_KEY="3a5273777ae14692b323ec8fb22b2613" `

OPENAI\_DEPLOYMENT\_NAME="gpt-4.1" `

OPENAI\_API\_VERSION="2025-04-14"

Step 6: Generate SAS Token for Blob Access

$SAS\_TOKEN = az storage container generate-sas `

--account-name $STORAGE\_ACCOUNT `

--name $CONTAINER\_INPUT `

--permissions rwl `

--expiry 2025-12-31T23:59Z `

--output tsv

Write-Host "SAS Token: $SAS\_TOKEN"

Step 7: Use Azure Container Registry (ACR) for SonarQube Deployment

# Create Azure Container Registry

$ACR = "devuicr$SUFFIX"

az acr create --name $ACR --resource-group $RESOURCE\_GROUP --sku Basic --location $LOCATION

# Import SonarQube Image into Your ACR

az acr import `

--name $ACR `

--source docker.io/library/sonarqube:10.5-community `

--image sonarqube:10.5-community

# Deploy SonarQube Container from ACR

$ACR\_LOGIN\_SERVER = az acr show --name $ACR --query "loginServer" -o tsv

# Get ACR Credentials

az acr update -n devuicre97bd7fd --admin-enabled true

$ACR\_USERNAME = az acr credential show --name $ACR --query "username" -o tsv

$ACR\_PASSWORD = az acr credential show --name $ACR --query "passwords[0].value" -o tsv

# deploy to Azure Container Instances

az container create `

--resource-group $RESOURCE\_GROUP `

--name "sonarqube" `

--image "$ACR\_LOGIN\_SERVER/sonarqube:10.5-community" `

--cpu 2 --memory 4 `

--ports 9000 `

--dns-name-label "sonarqube-devui$SUFFIX" `

--environment-variables SONARQUBE\_JDBC\_USERNAME=sonar SONARQUBE\_JDBC\_PASSWORD=sonar `

--ip-address public `

--os-type Linux `

--registry-login-server $ACR\_LOGIN\_SERVER `

--registry-username $ACR\_USERNAME `

--registry-password $ACR\_PASSWORD

Step 8: Access SonarQube

az container show --resource-group $RESOURCE\_GROUP --name "sonarqube" --query "ipAddress.fqdn" -o tsv

eg. Output: sonarqube-devuie97bd7fd.eastus.azurecontainer.io

Then access through:

<http://sonarqube-devuie97bd7fd.eastus.azurecontainer.io:9000>

**Default login:**

* Username: admin
* Password: admin

Update SonarQube

Step 1: Import the Latest Image to Your ACR

az acr import `

--name $ACR `

--source docker.io/library/sonarqube:community `

--image sonarqube:community

Step 2: Stop and Delete the Old Container

az container delete --name "sonarqube" --resource-group $RESOURCE\_GROUP –yes

Step 3: Deploy the New Version

az container create `

--resource-group $RESOURCE\_GROUP `

--name "sonarqube" `

--image "$ACR\_LOGIN\_SERVER/sonarqube:community" `

--cpu 2 --memory 4 `

--ports 9000 `

--dns-name-label "sonarqube-devui$SUFFIX" `

--environment-variables SONARQUBE\_JDBC\_USERNAME=sonar SONARQUBE\_JDBC\_PASSWORD=sonar `

--ip-address public `

--os-type Linux `

--registry-login-server $ACR\_LOGIN\_SERVER `

--registry-username $ACR\_USERNAME `

--registry-password $ACR\_PASSWORD

Step 4: Get the new public FQDN

az container show --resource-group $RESOURCE\_GROUP --name "sonarqube" --query "ipAddress.fqdn" -o tsv