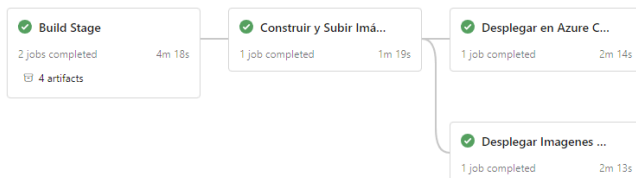


4.1 Modificar nuestro pipeline para incluir el deploy en QA y PROD de Imágenes Docker en Servicio Azure App Services con Soporte para Contenedores

4.1.1 - Agregar a nuestro pipeline una nueva etapa que dependa de nuestra etapa de Construcción y Pruebas y de la etapa de Construcción de Imágenes Docker y subida a ACR realizada en el TP08

```
##### STAGE DEPLOY TO AZURE APP SERVICE QA
#####
- stage: DeployImagesToAppServiceQA
  displayName: 'Desplegar Imágenes en Azure App Service (QA)'
  dependsOn:
    - BuildAndTestBackAndFront
    - DockerBuildAndPush
  condition: succeeded()
  jobs:
    - job: DeployImagesToAppServiceQA
      displayName: 'Desplegar Imágenes de API y Front en Azure App Service (QA)'
      pool:
        vmImage: 'ubuntu-latest'
      steps:
        - #-----
          # DEPLOY DOCKER API IMAGE TO AZURE APP SERVICE (QA)
          #-----
          Settings
          - task: AzureCLI@2
            displayName: 'Verificar y crear el recurso Azure App Service para API (QA) si no existe'
            inputs:
              azureSubscription: '$(ConnectedServiceName)'
              scriptType: 'bash'
              scriptLocation: 'inlineScript'
              inlineScript: |
                # Verificar si el App Service para la API ya existe
                if ! az webapp list --query "[?name=='$(WebAppApiNameContainersQA)' && resourceGroup=='$(ResourceGroupName)']" | jq -r '.length' | grep -q 0
                then
                  echo "El App Service para API QA no existe. Creando..."
                  # Crear el App Service sin especificar la imagen del contenedor
                  az webapp create --resource-group $(ResourceGroupName) --plan $(AppServicePlanLinux) --name $(WebAppApiNameContainersQA) --deployment-container-registry-url https://$(acrLoginServer)/$(backImageName):$(backImageTag)
                else
                  echo "El App Service para API QA ya existe. Actualizando la imagen..."
                fi
            # Configurar el App Service para usar Azure Container Registry (ACR)
            az webapp config container set --name $(WebAppApiNameContainersQA) --resource-group $(ResourceGroupName) \
              --container-image-name $(acrLoginServer)/$(backImageName):$(backImageTag) \
              --container-registry-url https://$(acrLoginServer)/\
```

Stages Jobs



Activate Windows
Go to Settings to activate Windows.

Desplegar Imagenes en Azure App Service (QA)

▼	✓	Desplegar Imagenes de API y Fr...	2m 0s
	✓	Initialize job	1s
	✓	Checkout angularcustom-tp7.git...	2s
	✓	Verificar y crear el recurso Az...	1m 55s
	✓	Post-job: Checkout angularcusto...	<1s
	✓	Finalize Job	<1s

- 4.2.1 Agregar tareas para generar Front en Azure App Service con Soporte para Contenedores

```
.....#-----
.....# DEPLOY DOCKER FRONT IMAGE TO AZURE APP SERVICE (QA)
.....#-----
.....Settings
.....- task: AzureCLI@2
.....  displayName: 'Verificar y crear el recurso Azure App Service para Front (QA) si no existe'
.....  inputs:
.....    azureSubscription: '$(ConnectedServiceName)'
.....    scriptType: 'bash'
.....    scriptLocation: 'inlineScript'
.....    inlineScript: |
.....      # Verificar si el App Service para el Front ya existe
.....      if ! az webapp list --query "[?name=='$(WebAppFrontNameContainersQA)' && resourceGroup=='$(ResourceGroupName)']" --o tsv | grep
.....      echo "El App Service para Front QA no existe. Creando..."
.....      # Crear el App Service sin especificar la imagen del contenedor
.....      az webapp create --resource-group $(ResourceGroupName) --plan $(AppServicePlanLinux) --name $(WebAppFrontNameContainersQA) --deployment-c
.....      else
.....      echo "El App Service para Front QA ya existe. Actualizando la imagen..."
.....      fi
.....
.....      # Configurar el App Service para usar Azure Container Registry (ACR)
.....      az webapp config container set --name $(WebAppFrontNameContainersQA) --resource-group $(ResourceGroupName) \
.....      --container-image-name $(acrLoginServer)/$(frontImageName):$(frontImageTag) \
.....      --container-registry-url https://$(acrLoginServer) \
.....      --container-registry-user $(acrName) \
.....      --container-registry-password $(az acr credential show --name $(acrName) --query "passwords[0].value" --o tsv)
.....
.....      # Establecer variables de entorno para el front-end
.....      az webapp config appsettings set --name $(WebAppFrontNameContainersQA) --resource-group $(ResourceGroupName) \
.....      --settings API_URL="http://$(WebAppApiNameContainersQA).azurewebsites.net"
```

```

41 | | | | |
42 | | | | | # Opcional: Reiniciar las aplicaciones web para aplicar cambios
    | | | | | Settings
43 | | | | | --task: AzureCLI@2
44 | | | | | displayName: 'Reiniciar App Service de API (QA)'
45 | | | | | inputs:
46 | | | | |   azureSubscription: '${ConnectedServiceName}'
47 | | | | |   scriptType: 'bash'
48 | | | | |   scriptLocation: 'inlineScript'
49 | | | | |   inlineScript: |
50 | | | | |     az webapp restart --name $(WebAppApiNameContainersQA) --resource-group $(ResourceGroupName)
51 | | | | |
    | | | | | Settings
52 | | | | | --task: AzureCLI@2
53 | | | | | displayName: 'Reiniciar App Service de Front (QA)'
54 | | | | | inputs:
55 | | | | |   azureSubscription: '${ConnectedServiceName}'
56 | | | | |   scriptType: 'bash'
57 | | | | |   scriptLocation: 'inlineScript'
58 | | | | |   inlineScript: |
59 | | | | |     az webapp restart --name $(WebAppFrontNameContainersQA) --resource-group $(ResourceGroupName)
60 | | | | |

```

Verificar y crear el recurso Azure App Service para Front (QA) si no existe
View raw log

```

1 Starting: Verificar y crear el recurso Azure App Service para Front (QA) si no existe
2 =====
3 Task      : Azure CLI
4 Description: Run Azure CLI commands against an Azure subscription in a PowerShell Core/Shell script when running on Linux agent or PowerShell/PowerShell Core/Batch script when
5 Version   : 2.247.1
6 Author    : Microsoft Corporation
7 Help      : https://docs.microsoft.com/azure/devops/pipelines/tasks/deploy/azure-cli
8 =====
9 /usr/bin/az --version
10 azure-cli      2.65.0
11
12 core           2.65.0
13 telemetry      1.1.0
14
15 Extensions:
16 azure-devops   1.0.1
17
18 Dependencies:
19 msal            1.31.0
20 azure-mgmt-resource 23.1.1
21
22 Python location '/opt/az/bin/python3'
23 Extensions directory '/opt/az/azcliextensions'
24
25 Python (Linux) 3.11.8 (main, Sep 25 2024, 11:33:44) [GCC 11.4.0]
26
27 Legal docs and information: aka.ms/AzureCliLegal
28
29

```

- 4.2.2 Agregar variables necesarias para el funcionamiento de la nueva etapa considerando que debe haber 2 entornos QA y PROD para Back y Front.

```

# Variables para QA
backContainerInstanceNameQA: 'as-crud-api-qa-sm'
frontContainerInstanceNameQA: 'as-crud-front-qa-sm'
backImageTagQA: 'qa'
frontImageTagQA: 'qa'
container-cpu-api-qa: 1 # CPUs de nuestro container de QA
container-memory-api-qa: 1.5 # RAM de nuestro container de QA
WebAppApiNameContainersQA: 'my-webapp-api-qa'
WebAppFrontNameContainersQA: 'my-webapp-front-qa'
container-url-api-qa: 'as-crud-api-qa-sm.westeurope.azurecontainer.io'

# Variables para PROD
backContainerInstanceNamePROD: 'as-crud-api-prod-sm'
frontContainerInstanceNamePROD: 'as-crud-front-prod-sm'
backImageTagPROD: 'prod'
frontImageTagPROD: 'prod'
container-cpu-api-prod: 1 # Ajusta según sea necesario
container-memory-api-prod: 1.5 # Ajusta según sea necesario
WebAppApiNameContainersPROD: 'my-webapp-api-prod'
WebAppFrontNameContainersPROD: 'my-webapp-front-prod'
container-url-api-prod: 'as-crud-api-prod-sm.westeurope.azurecontainer.io'

# Variables comunes
AppServicePlanLinux: 'MiWebApp02'
cnn_string_qaa: '$(cnn_string_qa)' # Usamos la misma cadena para QA y PROD

```

- 4.2.3 Agregar tareas para correr pruebas de integración en el entorno de QA de Back y Front creado en Azure App Services con Soporte para Contenedores.

```

194 #-----
195 # Ejecutar pruebas E2E del Front-End en Azure App Service (QA)
196 #-----
197 |. . . . .| script: |
198 |. . . . .| echo "Ejecutando pruebas E2E del Front-End en Azure App Service (QA)..."
199 |. . . . .| npx cypress run --config baseUrl=https://$(WebAppFrontNameContainersQA).azurewebsites.net
200 |. . . . .| displayName: 'Ejecutar Pruebas E2E de Front-End en Azure App Service (QA) con Cypress'
201 |. . . . .|
202 |. . . . .| # Publicar resultados de pruebas E2E del Front-End
203 |. . . . .| Settings
204 |. . . . .| task: PublishTestResults@2
205 |. . . . .| displayName: 'Publicar Resultados de Pruebas E2E de Front-End en Azure App Service (QA)'
206 |. . . . .| inputs:
207 |. . . . .| testResultsFormat: 'JUnit'
208 |. . . . .| testResultsFiles: 'cypress/results/*.xml'
209 |. . . . .| failTaskOnFailedTests: true

```

- 4.2.4 Agregar etapa que dependa de la etapa de Deploy en QA que genere un entorno de PROD.

```

#-----
### STAGE DEPLOY TO AZURE APP SERVICE PROD
#-----

- stage: DeployImagesToAppServicePROD
  displayName: 'Desplegar Imágenes en Azure App Service (PROD)'
  dependsOn:
    - DeployImagesToAppServiceQA
    - IntegrationTestsQA
  condition: succeeded()
  jobs:
    - job: DeployImagesToAppServicePROD
      displayName: 'Desplegar Imágenes de API y Front en Azure App Service (PROD)'
      pool:
        vmImage: 'ubuntu-latest'
      steps:
        - # Fetch ACR credentials
          Settings
          - task: AzureCLI@2
            displayName: 'Fetch ACR credentials'
            inputs:
              azureSubscription: '$(ConnectedServiceName)'
              scriptType: bash
              scriptLocation: inlineScript
              inlineScript: |
                ACR_PASSWORD=$(az acr credential show --name $(acrName) --query "passwords[0].value" --o tsv)
                echo "##vso[task.setvariable variable=ACR_PASSWORD]$ACR_PASSWORD"

450 #-----
451 # DEPLOY DOCKER API IMAGE TO AZURE APP SERVICE (PROD)
452 #-----
      Settings
      - task: AzureCLI@2
        displayName: 'Verificar y crear el recurso Azure App Service para API (PROD) si no existe'
        inputs:
          azureSubscription: '$(ConnectedServiceName)'
          scriptType: 'bash'
          scriptLocation: 'inlineScript'
          inlineScript: |
            # Verificar si el App Service para la API ya existe
            if ! az webapp list --query "[?name=='$(WebAppApiNameContainersPROD)' && resourceGroup=='$(ResourceGroupName)'] | .length(@)" --o tsv | grep
            echo "El App Service para API PROD no existe. Creando..."
            # Crear el App Service sin especificar la imagen del contenedor
            az webapp create --resource-group $(ResourceGroupName) --plan $(AppServicePlanLinux) --name $(WebAppApiNameContainersPROD) --deployment-c
            else
            echo "El App Service para API PROD ya existe. Actualizando la imagen..."
            fi
        # Configurar el App Service para usar Azure Container Registry (ACR)
        az webapp config container set --name $(WebAppApiNameContainersPROD) --resource-group $(ResourceGroupName) \
        --container-image-name $(acrLoginServer)/$(backImageName):$(backImageTagPROD) \
        --container-registry-url https://$(acrLoginServer) \
        --container-registry-user $(acrName) \
        --container-registry-password $(ACR_PASSWORD)
        # Establecer variables de entorno para el back-end
        az webapp config appsettings set --name $(WebAppApiNameContainersPROD) --resource-group $(ResourceGroupName) \
        --settings ConnectionStrings_DefaultConnection="$(cnn_string_qaa)" # Usamos la misma cadena

```

```

480 .....#-----
481 .....# DEPLOY DOCKER FRONT IMAGE TO AZURE APP SERVICE (PROD)
482 .....#-----
      Settings
483 .....- task: AzureCLI@2
484 .....  displayName: 'Verificar y crear el recurso Azure App Service para Front (PROD) si no existe'
485 .....  inputs:
486 .....    azureSubscription: '$(ConnectedServiceName)'
487 .....    scriptType: 'bash'
488 .....    scriptLocation: 'inlineScript'
489 .....    inlineScript: |
490 .....      # Verificar si el App Service para el Front ya existe
491 .....      if ! az webapp list --query "[?name=='$(WebAppFrontNameContainersPROD)' && resourceGroup=='$(ResourceGroupName)']" | length(@) --o tsv | grep
492 .....      echo "El App Service para Front PROD no existe. Creando..."
493 .....      # Crear el App Service sin especificar la imagen del contenedor
494 .....      az webapp create --resource-group $(ResourceGroupName) --plan $(AppServicePlanLinux) --name $(WebAppFrontNameContainersPROD) --deployment-c
495 .....      else
496 .....      echo "El App Service para Front PROD ya existe. Actualizando la imagen..."
497 .....      fi
498 .....
499 .....      # Configurar el App Service para usar Azure Container Registry (ACR)
500 .....      az webapp config container set --name $(WebAppFrontNameContainersPROD) --resource-group $(ResourceGroupName) \
501 .....      --container-image-name $(acrLoginServer)/$(frontImageName):$(frontImageTagPROD) \
502 .....      --container-registry-url https://$(acrLoginServer) \
503 .....      --container-registry-user $(acrName) \
504 .....      --container-registry-password $(ACR_PASSWORD)
505 .....
506 .....      # Establecer variables de entorno para el front-end
507 .....      az webapp config appsettings set --name $(WebAppFrontNameContainersPROD) --resource-group $(ResourceGroupName) \
508 .....      --settings API_URL="http://$(WebAppApiNameContainersPROD).azurewebsites.net"
509 .....
.....
.....# Opcional: Reiniciar las aplicaciones web para aplicar cambios
      Settings
.....- task: AzureCLI@2
.....  displayName: 'Reiniciar App Service de API (PROD)'
.....  inputs:
.....    azureSubscription: '$(ConnectedServiceName)'
.....    scriptType: 'bash'
.....    scriptLocation: 'inlineScript'
.....    inlineScript: |
.....      az webapp restart --name $(WebAppApiNameContainers) --resource-group $(ResourceGroupName)
.....
      Settings
.....- task: AzureCLI@2
.....  displayName: 'Reiniciar App Service de Front (PROD)'
.....  inputs:
.....    azureSubscription: '$(ConnectedServiceName)'
.....    scriptType: 'bash'
.....    scriptLocation: 'inlineScript'
.....    inlineScript: |
.....      az webapp restart --name $(WebAppFrontNameContainers) --resource-group $(ResourceGroupName)
.....

```

- 4.2.5 Entregar un pipeline que incluya:

- A) Etapa Construcción y Pruebas Unitarias y Code Coverage Back y Front

```
46 stages:
47 - stage: Build
48   . displayName: 'Build Stage'
49   . jobs:
50     . # Job para el Back-End
51     . - job: Backend
52       . displayName: 'Build y Análisis del Back-End'
53       . steps:
54         . - checkout: self
55         . - fetchDepth: 0
56
57       . # Restaurar paquetes NuGet
58       . Settings
59       . - task: DotNetCoreCLI@2
60         . displayName: 'Restore NuGet Packages'
61         . inputs:
62           . command: 'restore'
63           . projects: '$(solution)'
64
65       . # Compilar la solución del back-end y pruebas
66       . Settings
67       . - task: DotNetCoreCLI@2
68         . displayName: 'Build Backend y Pruebas Unitarias'
69         . inputs:
70           . command: 'build'
71           . projects: '**/*.sln' . # Esto compila tanto EmployeeCrudApi como EmployeeCrudApi.Tests
72           . arguments: '--configuration $(configuration)'
73
74       . # Publicar Back-End
75       . Settings
76       . - task: DotNetCoreCLI@2
77         . displayName: 'Publicar Back-End'
78         . inputs:
79           . command: publish
80           . publishWebProjects: true
81           . arguments: '--configuration $(configuration) --output $(Build.ArtifactStagingDirectory)'
82           . zipAfterPublish: false
83
84 ..
```

```

81 | | | | | # Publicar Artefactos del Back-End
    | | | | | Settings
82 | | | | | - task: PublishBuildArtifacts@1
83 | | | | | displayName: 'Publicar Artefactos de Back'
84 | | | | | inputs:
85 | | | | | pathToPublish: '$(buildOutput)'
86 | | | | | artifactName: 'drop-back'
87 | | | | | publishLocation: 'Container'
88 | | | | |
    | | | | | Settings
89 | | | | | - task: PublishPipelineArtifact@1
90 | | | | | displayName: 'Publicar Dockerfile de Back'
91 | | | | | inputs:
92 | | | | | targetPath: '$(Build.SourcesDirectory)/api/Dockerfile'
93 | | | | | artifact: 'dockerfile-back'
94 | | | | |
95 | | | | | # Job para el Front-End
96 | | | | | job: Frontend
97 | | | | | displayName: 'Build y Análisis del Front-End'
98 | | | | | steps:
99 | | | | | - checkout: self
100 | | | | | fetchDepth: 0
101 | | | | |
102 | | | | | # Instalar Node.js
    | | | | | Settings
103 | | | | | - task: NodeTool@0
104 | | | | | inputs:
105 | | | | | versionSpec: '18.x'
106 | | | | | displayName: 'Instalar Node.js'
107 | | | | |
108 | | | | | # Instalar dependencias del front-end
109 | | | | | script: npm install
110 | | | | | displayName: 'Instalar Dependencias de Front'
111 | | | | | workingDirectory: 'EmployeeCrudAngular'
112 | | | | |
113 | | | | | # Generar el archivo env.js
114 | | | | | script: node src/generate-env.js
115 | | | | | displayName: 'Generar archivo env.js'
116 | | | | | workingDirectory: './EmployeeCrudAngular'
117 | | | | |
118 | | | | | # Compilar el front-end
119 | | | | | script: npx ng build --configuration production
120 | | | | | displayName: 'Build de Front'
121 | | | | | workingDirectory: './EmployeeCrudAngular'
122 | | | | | condition: succeeded()
123 | | | | |
124 | | | | | # Publicar artefactos del front-end
    | | | | | Settings
125 | | | | | - task: PublishBuildArtifacts@1
126 | | | | | displayName: 'Publicar Artefactos de Front'
127 | | | | | inputs:
128 | | | | | pathToPublish: './EmployeeCrudAngular/dist'
129 | | | | | artifactName: 'drop-front'
130 | | | | | publishLocation: 'Container'
131 | | | | |
    | | | | | Settings
132 | | | | | - task: PublishPipelineArtifact@1
133 | | | | | displayName: 'Publicar Dockerfile de Front'
134 | | | | | inputs:
135 | | | | | targetPath: '$(Build.SourcesDirectory)/front/Dockerfile'
136 | | | | | artifact: 'dockerfile-front'
137 | | | | |

```


- B) Construcción de Imágenes Docker y subida a ACR

```
#-----
### STAGE BUILD DOCKER IMAGES Y PUSH A AZURE CONTAINER REGISTRY
#-----

-- stage: DockerBuildAndPush
-- displayName: 'Construir y Subir Imágenes Docker a ACR'
-- jobs:
-- -- job: docker_build_and_push
-- -- -- displayName: 'Construir y Subir Imágenes Docker a ACR'
-- -- -- pool:
-- -- -- vmImage: 'ubuntu-latest'

-- -- steps:
-- -- -- checkout: self
```

```
153 | | | | #-----
154 | | | | # BUILD DOCKER BACK IMAGE Y PUSH A AZURE CONTAINER REGISTRY
155 | | | | #-----
156 | | | |
157 | | | | Settings
158 | | | | -- task: DownloadPipelineArtifact@2
159 | | | | -- displayName: 'Descargar Artefactos de Back'
160 | | | | -- inputs:
161 | | | | -- buildType: 'current'
162 | | | | -- artifactName: 'drop-back'
163 | | | | -- targetPath: '$(Pipeline.Workspace)/drop-back'
164 | | | |
165 | | | | Settings
166 | | | | -- task: DownloadPipelineArtifact@2
167 | | | | -- displayName: 'Descargar Dockerfile de Back'
168 | | | | -- inputs:
169 | | | | -- buildType: 'current'
170 | | | | -- artifactName: 'dockerfile-back'
171 | | | | -- targetPath: '$(Pipeline.Workspace)/dockerfile-back'
172 | | | |
173 | | | | Settings
174 | | | | -- task: AzureCLI@2
175 | | | | -- displayName: 'Iniciar Sesión en Azure Container Registry (ACR)'
176 | | | | -- inputs:
177 | | | | -- azureSubscription: '$(ConnectedServiceName)'
178 | | | | -- scriptType: bash
179 | | | | -- scriptLocation: inlineScript
180 | | | | -- inlineScript: |
181 | | | | -- az acr login --name $(acrName)
182 | | | |
183 | | | | Settings
184 | | | | -- task: Docker@2
185 | | | | -- displayName: 'Construir Imagen Docker para Back'
186 | | | | -- inputs:
187 | | | | -- command: build
188 | | | | -- repository: $(acrLoginServer)/$(backImageName)
189 | | | | -- dockerfile: $(Pipeline.Workspace)/dockerfile-back/Dockerfile
190 | | | | -- buildContext: $(Pipeline.Workspace)/drop-back
191 | | | | -- tags: '$(backImageTagQA),$(backImageTagPROD)'
192 | | | |
193 | | | | Settings
194 | | | | -- task: Docker@2
195 | | | | -- displayName: 'Subir Imagen Docker de Back a ACR'
196 | | | | -- inputs:
197 | | | | -- command: push
198 | | | | -- repository: $(acrLoginServer)/$(backImageName)
199 | | | | -- tags: '$(backImageTagQA),$(backImageTagPROD)'
```

```

196 | | | | #-----
197 | | | | # BUILD DOCKER FRONT IMAGE Y PUSH A AZURE CONTAINER REGISTRY
198 | | | | #-----
199 | | | |
    | | | | Settings
200 | | | | - task: DownloadPipelineArtifact@2
201 | | | |   displayName: 'Descargar Artefactos de Front'
202 | | | |   inputs:
203 | | | |     buildType: 'current'
204 | | | |     artifactName: 'drop-front'
205 | | | |     targetPath: '$(Pipeline.Workspace)/drop-front'
206 | | | |
    | | | | Settings
207 | | | | - task: DownloadPipelineArtifact@2
208 | | | |   displayName: 'Descargar Dockerfile de Front'
209 | | | |   inputs:
210 | | | |     buildType: 'current'
211 | | | |     artifactName: 'dockerfile-front'
212 | | | |     targetPath: '$(Pipeline.Workspace)/dockerfile-front'
213 | | | |
    | | | | Settings
214 | | | | - task: Docker@2
215 | | | |   displayName: 'Construir Imagen Docker para Front'
216 | | | |   inputs:
217 | | | |     command: build
218 | | | |     repository: '$(acrLoginServer)/$(frontImageName)'
219 | | | |     dockerfile: '$(Pipeline.Workspace)/dockerfile-front/Dockerfile'
220 | | | |     buildContext: '$(Pipeline.Workspace)/drop-front'
221 | | | |     tags: '$(frontImageTagQA),$(frontImageTagPROD)'
222 | | | |
    | | | | Settings
223 | | | | - task: Docker@2
224 | | | |   displayName: 'Subir Imagen Docker de Front a ACR'
225 | | | |   inputs:
226 | | | |     command: push
227 | | | |     repository: '$(acrLoginServer)/$(frontImageName)'
228 | | | |     tags: '$(frontImageTagQA),$(frontImageTagPROD)'
229 | | | |

```

- C) Deploy Back y Front en QA con pruebas de integración para Azure Web Apps

```

231 #-----
232 ### STAGE: Deploy to Azure Web Apps (QA)
233 #-----
234 - stage: DeployToAzureWebAppsQA
235 |   displayName: 'Deploy to Azure Web Apps (QA)'
236 |   dependsOn: DockerBuildAndPush
237 |   condition: succeeded()
238 |   jobs:
239 |     # Job para el Back-End
240 |     - job: DeployQABackend
241 |       displayName: 'Deploy to QA -- Back-End'
242 |       steps:
243 |         Settings
244 |         - task: DownloadPipelineArtifact@2
245 |           displayName: 'Download Build Artifacts'
246 |           inputs:
247 |             buildType: 'current'
248 |             artifactName: 'drop-back'
249 |             targetPath: '$(Pipeline.Workspace)/drop-back'
250 |         Settings
251 |         - task: AzureWebApp@1
252 |           displayName: 'Deploy to QA -- Back-End'
253 |           inputs:
254 |             azureSubscription: '$(ConnectedServiceName)'
255 |             appType: 'webApp'
256 |             appName: 'EmployeeCrudBack-QA'
257 |             package: '$(Pipeline.Workspace)/drop-back'
258 |     # Job para el Front-End
259 |     - job: DeployQAFrontend
260 |       displayName: 'Deploy to QA -- Front-End'
261 |       steps:
262 |         Settings
263 |         - task: DownloadPipelineArtifact@2
264 |           displayName: 'Download Build Artifacts'
265 |           inputs:
266 |             buildType: 'current'
267 |             artifactName: 'drop-front'
268 |             targetPath: '$(Pipeline.Workspace)/drop-front'
269 |         Settings
270 |         - task: AzureWebApp@1
271 |           displayName: 'Deploy to QA -- Front-End'
272 |           inputs:
273 |             azureSubscription: '$(ConnectedServiceName)'
274 |             appType: 'webApp'
275 |             appName: 'EmployeeCrudFront-QA'
276 |             package: '$(Pipeline.Workspace)/drop-front/employee-crud-angular/browser'
277 ---

```

```

277 |...# Job para los Test de Integración (ejecutado después del despliegue)
278 |...- job: IntegrationTestsAzureWebAppsQA
279 |...  displayName: 'Tests de Integracion en Azure Web Apps (QA)'
280 |...  dependsOn:
281 |...    - DeployQABackend
282 |...    - DeployQAFrontend
283 |...  steps:
284 |...    - script: npm install
285 |...      workingDirectory: ../EmployeeCrudAngular
286 |...      displayName: 'Instalar Dependencias para Pruebas'
287 |...
288 |...    - script: npx cypress run --config baseUrl=https://EmployeeCRUDFront.azurewebsites.net
289 |...      workingDirectory: ../EmployeeCrudAngular
290 |...      displayName: 'Correr Tests en Cypress E2E'
291 |...
292 |...  Settings
293 |...    - task: PublishTestResults@2
294 |...      displayName: 'Publicar Resultados de Cypress'
295 |...      inputs:
296 |...        testResultsFormat: 'JUnit'
297 |...        testResultsFiles: 'EmployeeCrudAngular/cypress/results/*.xml'
298 |...        testRunTitle: 'Cypress E2E Tests - QA'
299 |...        failTaskOnFailedTests: true

```

- D) Deploy Back y Front en QA con pruebas de integración para ACI

```

302
303 #-----
304 ### STAGE DEPLOY TO ACI QA
305 #-----
306 - -
307 - stage: DeployToACIQA
308 - displayName: 'Desplegar en Azure Container Instances (ACI) QA'
309 - dependsOn: DockerBuildAndPush
310 - jobs:
311 - - job: deploy_to_aci_qa
312 - - displayName: 'Desplegar en Azure Container Instances (ACI) QA'
313 - - pool:
314 - - vmImage: 'ubuntu-latest'
315 - - steps:
316 - - -
317 - - - # Fetch ACR credentials
318 - - - Settings
319 - - - - task: AzureCLI@2
320 - - - - displayName: 'Fetch ACR credentials'
321 - - - - inputs:
322 - - - - - azureSubscription: '${ConnectedServiceName}'
323 - - - - - scriptType: bash
324 - - - - - scriptLocation: inlineScript
325 - - - - - inlineScript: |
326 - - - - - ACR_PASSWORD=$(az acr credential show --name $(acrName) --query "passwords[0].value" -o tsv)
327 - - - - - echo "##vso[task.setvariable variable=ACR_PASSWORD]$ACR_PASSWORD"
328 - - - -
329 - - - - # DEPLOY DOCKER BACK IMAGE TO ACI QA
330 - - - - Settings
331 - - - - - task: AzureCLI@2
332 - - - - - displayName: 'Desplegar Imagen Docker de Back en ACI QA'
333 - - - - - inputs:
334 - - - - - - azureSubscription: '${ConnectedServiceName}'
335 - - - - - - scriptType: bash
336 - - - - - - scriptLocation: inlineScript
337 - - - - - - inlineScript: |
338 - - - - - - echo "Resource Group: $(ResourceGroupName)"
339 - - - - - - echo "Container Instance Name: $(backContainerInstanceNameQA)"
340 - - - - - - echo "ACR Login Server: $(acrLoginServer)"
341 - - - - - - echo "Image Name: $(backImageName)"
342 - - - - - - echo "Image Tag: $(backImageTagQA)"
343 - - - - - - echo "Connection String: $(cnn_string_qaa)"
344 - - - - - # Delete the existing container if present
345 - - - - - az container delete --resource-group $(ResourceGroupName) --name $(backContainerInstanceNameQA) --yes || true
346 - - - - - # Create the container
347 - - - - - az container create --resource-group $(ResourceGroupName) \
348 - - - - - --location westeurope \
349 - - - - - --name $(backContainerInstanceNameQA) \
350 - - - - - --image $(acrLoginServer)/$(backImageName):$(backImageTagQA) \
351 - - - - - --registry-login-server $(acrLoginServer) \
352 - - - - - --registry-username $(acrName) \
353 - - - - - --registry-password $(ACR_PASSWORD) \
354 - - - - - --dns-name-label $(backContainerInstanceNameQA) \
355 - - - - - --ports 80 \
356 - - - - - --environment-variables ConnectionStrings__DefaultConnection="$(cnn_string_qaa)" \
357 - - - - - --restart-policy Always \
358 - - - - - --cpu $(container-cpu-api-qa) \
359 - - - - - --memory $(container-memory-api-qa)
360

```

```

361 | | | | | # DEPLOY DOCKER FRONT IMAGE TO ACI QA
362 | | | | | Settings
363 | | | | | - task: AzureCLI@2
364 | | | | | displayName: 'Desplegar Imagen Docker de Front en ACI QA'
365 | | | | | inputs:
366 | | | | |   azureSubscription: '$(ConnectedServiceName)'
367 | | | | |   scriptType: bash
368 | | | | |   scriptLocation: inlineScript
369 | | | | |   inlineScript: |
370 | | | | |     echo "Resource Group: $(ResourceGroupName)"
371 | | | | |     echo "Container Instance Name: $(frontContainerInstanceNameQA)"
372 | | | | |     echo "ACR Login Server: $(acrLoginServer)"
373 | | | | |     echo "Image Name: $(frontImageName)"
374 | | | | |     echo "Image Tag: $(frontImageTagQA)"
375 | | | | |
376 | | | | |     # Delete the existing front container if present
377 | | | | |     az container delete --resource-group $(ResourceGroupName) --name $(frontContainerInstanceNameQA) --yes || true
378 | | | | |
379 | | | | |     # Create the front-end container
380 | | | | |     az container create --resource-group $(ResourceGroupName) \
381 | | | | |       --location westeurope \
382 | | | | |       --name $(frontContainerInstanceNameQA) \
383 | | | | |       --image $(acrLoginServer)/$(frontImageName):$(frontImageTagQA) \
384 | | | | |       --registry-login-server $(acrLoginServer) \
385 | | | | |       --registry-username $(acrName) \
386 | | | | |       --registry-password $(ACR_PASSWORD) \
387 | | | | |       --dns-name-label $(frontContainerInstanceNameQA) \
388 | | | | |       --ports 80 \
389 | | | | |       --environment-variables API_URL=$(container-url-api-qa) \
390 | | | | |       --restart-policy Always \
391 | | | | |       --cpu 1 \
392 | | | | |       --memory 1.5
393 | | | | |
394 | | | | | # Ejecutar pruebas E2E del Front-End en el contenedor de ACI QA
395 | | | | | - script: |
396 | | | | |   echo "Ejecutando pruebas E2E del Front-End en ACI QA..."
397 | | | | |   npx cypress run --config baseUrl=http://$(frontContainerInstanceNameQA).westeurope.azurecontainer.io
398 | | | | |   displayName: 'Ejecutar Pruebas E2E de Front-End en ACI QA con Cypress'
399 | | | | |
400 | | | | | # Publicar resultados de pruebas E2E del Front-End
401 | | | | | Settings
402 | | | | | - task: PublishTestResults@2
403 | | | | | displayName: 'Publicar Resultados de Pruebas E2E de Front-End en ACI QA'
404 | | | | | inputs:
405 | | | | |   testResultsFormat: 'JUnit'
406 | | | | |   testResultsFiles: 'cypress/results/*.xml'
407 | | | | |   failTaskOnFailedTests: true

```

- E) Deploy Back y Front en QA con pruebas de integración para Azure Web Apps con Soporte para contenedores

```

407 | | | | | #-----
408 | | | | | ### STAGE DEPLOY TO AZURE APP SERVICE QA
409 | | | | | #-----
410 | | | | |
411 | | | | | - stage: DeployImagesToAppServiceQA
412 | | | | |   displayName: 'Desplegar Imágenes en Azure App Service (QA)'
413 | | | | |   dependsOn:
414 | | | | |     - DockerBuildAndPush
415 | | | | |   condition: succeeded()
416 | | | | |   jobs:
417 | | | | |     - job: DeployImagesToAppServiceQA
418 | | | | |       displayName: 'Desplegar Imágenes de API y Front en Azure App Service (QA)'
419 | | | | |       pool:
420 | | | | |         vmImage: 'ubuntu-latest'
421 | | | | |       steps:
422 | | | | |         # Fetch ACR credentials
423 | | | | |         Settings
424 | | | | |         - task: AzureCLI@2
425 | | | | |           displayName: 'Fetch ACR credentials'
426 | | | | |           inputs:
427 | | | | |             azureSubscription: '$(ConnectedServiceName)'
428 | | | | |             scriptType: bash
429 | | | | |             scriptLocation: inlineScript
430 | | | | |             inlineScript: |
431 | | | | |               ACR_PASSWORD=$(az acr credential show --name $(acrName) --query "passwords[0].value" -o tsv)
432 | | | | |               echo "##vso[task.setvariable variable=ACR_PASSWORD]$ACR_PASSWORD"

```

```

432 | .....#-----
433 | .....# DEPLOY DOCKER API IMAGE TO AZURE APP SERVICE (QA)
434 | .....#-----
435 | .....Settings
436 | .....- task: AzureCLI@2
437 | .....  displayName: 'Verificar y crear el recurso Azure App Service para API (QA) si no existe'
438 | .....  inputs:
439 | .....    azureSubscription: '$(ConnectedServiceName)'
440 | .....    scriptType: 'bash'
441 | .....    scriptLocation: 'inlineScript'
442 | .....    inlineScript: |
443 | .....      # Verificar si el App Service para la API ya existe
444 | .....      if ! az webapp list --query "[?name=='$(WebAppApiNameContainersQA)' && resourceGroup=='$(ResourceGroupName)']" --o tsv | grep -q '
445 | .....      echo "El App Service para API QA no existe. Creando..."
446 | .....      # Crear el App Service sin especificar la imagen del contenedor
447 | .....      az webapp create --resource-group $(ResourceGroupName) --plan $(AppServicePlanLinux) --name $(WebAppApiNameContainersQA) --deployment-conta
448 | .....      else
449 | .....      echo "El App Service para API QA ya existe. Actualizando la imagen..."
450 | .....      fi
451 | .....
452 | .....      # Configurar el App Service para usar Azure Container Registry (ACR)
453 | .....      az webapp config container set --name $(WebAppApiNameContainersQA) --resource-group $(ResourceGroupName) \
454 | .....      --container-image-name $(acrLoginServer)/$(backImageName):$(backImageTagQA) \
455 | .....      --container-registry-url https://$(acrLoginServer) \
456 | .....      --container-registry-user $(acrName) \
457 | .....      --container-registry-password $(ACR_PASSWORD)
458 | .....
459 | .....      # Establecer variables de entorno para el back-end
460 | .....      az webapp config appsettings set --name $(WebAppApiNameContainersQA) --resource-group $(ResourceGroupName) \
461 | .....      --settings ConnectionStrings_DefaultConnection="$(conn_string_qaa)"
462 | .....
463 | .....#-----
464 | .....# DEPLOY DOCKER FRONT IMAGE TO AZURE APP SERVICE (QA)
465 | .....#-----
466 | .....Settings
467 | .....- task: AzureCLI@2
468 | .....  displayName: 'Verificar y crear el recurso Azure App Service para Front (QA) si no existe'
469 | .....  inputs:
470 | .....    azureSubscription: '$(ConnectedServiceName)'
471 | .....    scriptType: 'bash'
472 | .....    scriptLocation: 'inlineScript'
473 | .....    inlineScript: |
474 | .....      # Verificar si el App Service para el Front ya existe
475 | .....      if ! az webapp list --query "[?name=='$(WebAppFrontNameContainersQA)' && resourceGroup=='$(ResourceGroupName)']" --o tsv | grep -q '
476 | .....      echo "El App Service para Front QA no existe. Creando..."
477 | .....      # Crear el App Service sin especificar la imagen del contenedor
478 | .....      az webapp create --resource-group $(ResourceGroupName) --plan $(AppServicePlanLinux) --name $(WebAppFrontNameContainersQA) --deployment-con
479 | .....      else
480 | .....      echo "El App Service para Front QA ya existe. Actualizando la imagen..."
481 | .....      fi
482 | .....
483 | .....      # Configurar el App Service para usar Azure Container Registry (ACR)
484 | .....      az webapp config container set --name $(WebAppFrontNameContainersQA) --resource-group $(ResourceGroupName) \
485 | .....      --container-image-name $(acrLoginServer)/$(frontImageName):$(frontImageTagQA) \
486 | .....      --container-registry-url https://$(acrLoginServer) \
487 | .....      --container-registry-user $(acrName) \
488 | .....      --container-registry-password $(ACR_PASSWORD)
489 | .....
490 | .....      # Establecer variables de entorno para el front-end
491 | .....      az webapp config appsettings set --name $(WebAppFrontNameContainersQA) --resource-group $(ResourceGroupName) \
492 | .....      --settings API_URL="http://$(WebAppApiNameContainersQA).azurewebsites.net"
493 | .....
494 | .....#-----
495 | .....# Ejecutar pruebas E2E del Front-End en Azure App Service (QA)
496 | .....#-----
497 | .....  script: |
498 | .....    echo "Ejecutando pruebas E2E del Front-End en Azure App Service (QA)..."
499 | .....    npx cypress run --config baseUrl=https://$(WebAppFrontNameContainersQA).azurewebsites.net
500 | .....    displayName: 'Ejecutar Pruebas E2E de Front-End en Azure App Service (QA) con Cypress'
501 | .....
502 | .....  # Publicar resultados de pruebas E2E del Front-End
503 | .....  Settings
504 | .....  - task: PublishTestResults@2
505 | .....    displayName: 'Publicar Resultados de Pruebas E2E de Front-End en Azure App Service (QA)'
506 | .....    inputs:
507 | .....      testResultsFormat: 'JUnit'
508 | .....      testResultsFiles: 'cypress/results/*.xml'
509 | .....      failTaskOnFailedTests: true
510 | .....

```

- F) Aprobación manual de QA para los puntos C,D,E

- G) Deploy Back y Front en PROD para Azure Web Apps

```

529 #-----
530 ### STAGE: Deploy to Azure Web Apps (PROD)
531 #-----
532 - stage: DeployToAzureWebAppsProd
533   displayName: 'Deploy to Azure Web Apps (PROD)'
534   dependsOn: DeployToAzureWebAppsQA
535   condition: succeeded()
536   jobs:
537     - deployment: DeployToAzureWebAppsProd
538       displayName: 'Deploy to Azure Web Apps (PROD)'
539       environment: 'Production'
540       strategy:
541         runOnce:
542           deploy:
543             steps:
544               Settings
545               - task: DownloadPipelineArtifact@2
546                 displayName: 'Download Build Artifacts'
547                 inputs:
548                   buildType: 'current'
549                   artifactName: 'drop-back'
550                   targetPath: '$(Pipeline.Workspace)/drop-back'
551               Settings
552               - task: AzureWebApp@1
553                 displayName: 'Deploy to PROD - Back-End'
554                 inputs:
555                   azureSubscription: '$(ConnectedServiceName)'
556                   appType: 'webApp'
557                   appName: 'EmployeeCrudBack-Prod'
558                   package: '$(Pipeline.Workspace)/drop-back'

```



```
551 | | | | | Settings
552 | | | | | -- task: AzureWebApp@1
553 | | | | | displayName: 'Deploy to PROD -- Back-End'
554 | | | | | inputs:
555 | | | | |   azureSubscription: '$(ConnectedServiceName)'
556 | | | | |   appType: 'webApp'
557 | | | | |   appName: 'EmployeeCrudBack-Prod'
558 | | | | |   package: '$(Pipeline.Workspace)/drop-back'

559 | | | | | Settings
560 | | | | | -- task: DownloadPipelineArtifact@2
561 | | | | | displayName: 'Download Build Artifacts'
562 | | | | | inputs:
563 | | | | |   buildType: 'current'
564 | | | | |   artifactName: 'drop-front'
565 | | | | |   targetPath: '$(Pipeline.Workspace)/drop-front'

566 | | | | | Settings
567 | | | | | -- task: AzureWebApp@1
568 | | | | | displayName: 'Deploy to PROD -- Front-End'
569 | | | | | inputs:
570 | | | | |   azureSubscription: '$(ConnectedServiceName)'
571 | | | | |   appType: 'webApp'
572 | | | | |   appName: 'EmployeeCrudFront-Prod'
573 | | | | |   package: '$(Pipeline.Workspace)/drop-front/employee-crud-angular/browser'
```

ussalovesmuzska / sample02 / Pipelines / angularcustom-tp7.git / 20241022.36

+

Summary

Associated pipelines

Code Coverage

Warnings 2

No data was written into the file /home/vsts/work/_temp/task_outputs/build_1729636784284.txt

Construir y Subir Imágenes Docker a ACR • Construir y Subir Imágenes Docker a ACR • Construir Imagen Docker para Back

No data was written into the file /home/vsts/work/_temp/task_outputs/build_1729636811045.txt

Construir y Subir Imágenes Docker a ACR • Construir y Subir Imágenes Docker a ACR • Construir Imagen Docker para Front

This pipeline needs permission to access a resource before this run can continue to Deploy to Azure Web Apps (PROD)

Stages

Jobs

Build Stage

2 jobs completed 3m 21s

4 artifacts

Construir y Subir Imá...

1 job completed 1m 20s

Deploy to Azure Web...

2 jobs completed 7m 46s

Desplegar en Azure C...

0/1 completed 50s

Desplegar en Azure Contain... 5...

Cancel

Desplegar Imágenes ...

1 job completed 2m 46s

Deploy to Azure Web...

Waiting

Permission needed

Deploy to Azure Con...

Not started

Desplegar Imágenes ...

Not started

Checks for Deploy to Azure Web Apps (PROD)

Permission Environment Production

Permission needed

Permit

Activate Windows

Go to Settings to activate Windows.

- H) Deploy Back y Front en PROD para ACI

```

576 #-----
577 ### STAGE: Deploy to ACI (PROD)
578 #-----
579 - stage: DeployToACIProd
580 - displayName: 'Deploy to Azure Container Instances (ACI) PROD'
581 - dependsOn: DeployToACIQA
582 - condition: succeeded()
583 - jobs:
584 - - deployment: DeployToACIProd
585 - - displayName: 'Deploy to Azure Container Instances (ACI) PROD'
586 - - environment: 'Production'
587 - - strategy:
588 - - runOnce:
589 - - deploy:
590 - - steps:
591 - - # Fetch ACR credentials
592 - - Settings
593 - - - task: AzureCLI@2
594 - - - displayName: 'Fetch ACR credentials for PROD'
595 - - - inputs:
596 - - - azureSubscription: '$(ConnectedServiceName)'
597 - - - scriptType: bash
598 - - - scriptLocation: inlineScript
599 - - - inlineScript: |
600 - - - ACR_PASSWORD=$(az acr credential show --name $(acrName) --query "passwords[0].value" -o tsv)
601 - - - echo "##vso[task.setvariable variable=ACR_PASSWORD]$ACR_PASSWORD"
602 - - # DEPLOY DOCKER BACK IMAGE TO ACI PROD
603 - - Settings
604 - - - task: AzureCLI@2
605 - - - displayName: 'Deploy Back-End Docker Image to ACI (PROD)'
606 - - - inputs:
607 - - - azureSubscription: '$(ConnectedServiceName)'
608 - - - scriptType: bash
609 - - - scriptLocation: inlineScript
610 - - - inlineScript: |
611 - - - echo "Resource Group: $(ResourceGroupName)"
612 - - - echo "Container Instance Name: $(backContainerInstanceNamePROD)"
613 - - - echo "ACR Login Server: $(acrLoginServer)"
614 - - - echo "Image Name: $(backImageName)"
615 - - - echo "Image Tag: $(backImageTagPROD)"
616 - - - echo "Connection String: $(cnn_string_prod)" # Usamos cnn_string_qaa
617 - - - # Delete the existing container if present
618 - - - az container delete --resource-group $(ResourceGroupName) --name $(backContainerInstanceNamePROD) --yes || true
619 - - - # Create the container
620 - - - az container create --resource-group $(ResourceGroupName) \
621 - - - --location westeurope \
622 - - - --name $(backContainerInstanceNamePROD) \
623 - - - --image $(acrLoginServer)/$(backImageName):$(backImageTagPROD) \
624 - - - --registry-login-server $(acrLoginServer) \
625 - - - --registry-username $(acrName) \
626 - - - --registry-password $(ACR_PASSWORD) \
627 - - - --dns-name-label $(backContainerInstanceNamePROD) \
628 - - - --ports 80 \
629 - - - --environment-variables ConnectionStrings__DefaultConnection="$(cnn_string_qaa)" \
630 - - - --restart-policy Always \
631 - - - --cpu $(container-cpu-api-prod) \
632 - - - --memory $(container-memory-api-prod)
633 - - # DEPLOY DOCKER FRONT IMAGE TO ACI PROD
634 - - Settings
635 - - - task: AzureCLI@2
636 - - - displayName: 'Deploy Front-End Docker Image to ACI (PROD)'
637 - - - inputs:
638 - - - azureSubscription: '$(ConnectedServiceName)'
639 - - - scriptType: bash
640 - - - scriptLocation: inlineScript
641 - - - inlineScript: |
642 - - - echo "Resource Group: $(ResourceGroupName)"
643 - - - echo "Container Instance Name: $(frontContainerInstanceNamePROD)"
644 - - - echo "ACR Login Server: $(acrLoginServer)"
645 - - - echo "Image Name: $(frontImageName)"
646 - - - echo "Image Tag: $(frontImageTagPROD)"
647 - - - # Delete the existing container if present
648 - - - az container delete --resource-group $(ResourceGroupName) --name $(frontContainerInstanceNamePROD) --yes || true
649 - - - # Create the container
650 - - - az container create --resource-group $(ResourceGroupName) \
651 - - - --location westeurope \
652 - - - --name $(frontContainerInstanceNamePROD) \
653 - - - --image $(acrLoginServer)/$(frontImageName):$(frontImageTagPROD) \
654 - - - --registry-login-server $(acrLoginServer) \
655 - - - --registry-username $(acrName) \
656 - - - --registry-password $(ACR_PASSWORD) \
657 - - - --dns-name-label $(frontContainerInstanceNamePROD) \
658 - - - --ports 80 \
659 - - - --environment-variables API_URL=$(container-url-api-prod) \
660 - - - --restart-policy Always \
661 - - - --cpu 1 \
662 - - - --memory 1.5
663 - -

```

- I) Deploy Back y Front en PROD para Azure Web Apps con Soporte para contenedores

```

664 #-----
665 ### STAGE DEPLOY TO AZURE APP SERVICE PROD
666 #-----
667
668 --stage: DeployImagesToAppServicePROD
669 |--displayName: 'Desplegar Imágenes en Azure App Service (PROD)'
670 |--dependsOn:
671 |  |-- DeployImagesToAppServiceQA
672 |--condition: succeeded()
673 |--jobs:
674 |  |-- job: DeployImagesToAppServicePROD
675 |    |-- displayName: 'Desplegar Imágenes de API y Front en Azure App Service (PROD)'
676 |    |-- pool:
677 |      |-- vmImage: 'ubuntu-latest'
678 |      |-- steps:
679 |        |-- # Fetch ACR credentials
680 |          | Settings
681 |          | -- task: AzureCLI@2
682 |          | -- displayName: 'Fetch ACR credentials'
683 |          | -- inputs:
684 |            |-- azureSubscription: '${ConnectedServiceName}'
685 |            |-- scriptType: bash
686 |            |-- scriptLocation: inlineScript
687 |            |-- inlineScript: |
688 |              |-- ACR_PASSWORD=$(az acr credential show --name ${acrName} --query "passwords[0].value" --o tsv)
689 |              |-- echo "##vso[task.setvariable variable=ACR_PASSWORD]$ACR_PASSWORD"
690 |
691 |        |-- # DEPLOY DOCKER API IMAGE TO AZURE APP SERVICE (PROD)
692 |        | Settings
693 |        | -- task: AzureCLI@2
694 |        | -- displayName: 'Verificar y crear el recurso Azure App Service para API (PROD) si no existe'
695 |        | -- inputs:
696 |          |-- azureSubscription: '${ConnectedServiceName}'
697 |          |-- scriptType: 'bash'
698 |          |-- scriptLocation: 'inlineScript'
699 |          |-- inlineScript: |
700 |            |-- # Verificar si el App Service para la API ya existe
701 |            |-- if ! az webapp list --query "[?name=='$(WebAppApiNameContainersPROD)'] && resourceGroup=='$(ResourceGroupName)']" | length(@) --o tsv | grep
702 |              |-- echo "El App Service para API PROD no existe. Creando..."
703 |              |-- # Crear el App Service sin especificar la imagen del contenedor
704 |              |-- az webapp create --resource-group $(ResourceGroupName) --plan $(AppServicePlanLinux) --name $(WebAppApiNameContainersPROD) --deployment-c
705 |              |-- else
706 |                |-- echo "El App Service para API PROD ya existe. Actualizando la imagen..."
707 |                |-- fi
708 |
709 |            |-- # Configurar el App Service para usar Azure Container Registry (ACR)
710 |            |-- az webapp config container set --name $(WebAppApiNameContainersPROD) --resource-group $(ResourceGroupName) \
711 |              |-- --container-image-name $(acrLoginServer)/$(backImageName):$(backImageTagPROD) \
712 |              |-- --container-registry-url https://$(acrLoginServer) \
713 |              |-- --container-registry-user $(acrName) \
714 |              |-- --container-registry-password $(ACR_PASSWORD)
715 |
716 |            |-- # Establecer variables de entorno para el back-end
717 |            |-- az webapp config appsettings set --name $(WebAppApiNameContainersPROD) --resource-group $(ResourceGroupName) \
718 |              |-- --settings ConnectionStrings_DefaultConnection="$(cnn_string_qaa)" # Usamos la misma cadena
719 |
720 |
721 |        |-- # DEPLOY DOCKER FRONT IMAGE TO AZURE APP SERVICE (PROD)
722 |        | Settings
723 |        | -- task: AzureCLI@2
724 |        | -- displayName: 'Verificar y crear el recurso Azure App Service para Front (PROD) si no existe'
725 |        | -- inputs:
726 |          |-- azureSubscription: '${ConnectedServiceName}'
727 |          |-- scriptType: 'bash'
728 |          |-- scriptLocation: 'inlineScript'
729 |          |-- inlineScript: |
730 |            |-- # Verificar si el App Service para el Front ya existe
731 |            |-- if ! az webapp list --query "[?name=='$(WebAppFrontNameContainersPROD)'] && resourceGroup=='$(ResourceGroupName)']" | length(@) --o tsv | grep
732 |              |-- echo "El App Service para Front PROD no existe. Creando..."
733 |              |-- # Crear el App Service sin especificar la imagen del contenedor
734 |              |-- az webapp create --resource-group $(ResourceGroupName) --plan $(AppServicePlanLinux) --name $(WebAppFrontNameContainersPROD) --deployment-
735 |              |-- else
736 |                |-- echo "El App Service para Front PROD ya existe. Actualizando la imagen..."
737 |                |-- fi
738 |
739 |            |-- # Configurar el App Service para usar Azure Container Registry (ACR)
740 |            |-- az webapp config container set --name $(WebAppFrontNameContainersPROD) --resource-group $(ResourceGroupName) \
741 |              |-- --container-image-name $(acrLoginServer)/$(frontImageName):$(frontImageTagPROD) \
742 |              |-- --container-registry-url https://$(acrLoginServer) \
743 |              |-- --container-registry-user $(acrName) \
744 |              |-- --container-registry-password $(ACR_PASSWORD)
745 |
746 |            |-- # Establecer variables de entorno para el front-end
747 |            |-- az webapp config appsettings set --name $(WebAppFrontNameContainersPROD) --resource-group $(ResourceGroupName) \
748 |              |-- --settings API_URL="http://$(WebAppApiNameContainersPROD).azurewebsites.net"

```

Pipeline Completo:

Summary

Environments

Code Coverage

Associated pipelines

Triggered by Santiago Tomas Muscellini

View 6 changes

Repository and version

Time started and elapsed

Related

Tests and coverage

angularcustom-tp7.git

main cfa94665

Today at 8:14 PM

22m 48s

0 work items

4 published

Get started

Warnings 2

No data was written into the file /home/vsts/work/_temp/task_outputs/build_1729639191071.txt

Construir y Subir Imágenes Docker a ACR • Construir y Subir Imágenes Docker a ACR • Construir Imagen Docker para Back

No data was written into the file /home/vsts/work/_temp/task_outputs/build_1729639215044.txt

Construir y Subir Imágenes Docker a ACR • Construir y Subir Imágenes Docker a ACR • Construir Imagen Docker para Front

Stages

Jobs

Build Stage

2 jobs completed

4m 7s

4 artifacts

Construir y Subir Imá...

1 job completed

1m 12s

Deploy to Azure Web...

2 jobs completed

2m 0s

Deploy to Azure Web...

1 job completed

1m 26s

Desplegar en Azure C...

1 job completed

3m 12s

Desplegar Imágenes ...

1 job completed

2m 13s

Deploy to Azure Con...

1 job completed

3m 45s

Desplegar Imágenes ...

1 job completed

2m 53s

Activate Windows

Go to Settings to activate Windows.