

Trabajo Práctico 8 - Implementación de Contenedores en Azure y Automatización con Azure CLI

- Azure CLI instalado:

```
C:\Users\Manu>az

Welcome to Azure CLI!
-----
Use `az -h` to see available commands or go to https://aka.ms/cli.

Telemetry
-----
The Azure CLI collects usage data in order to improve your experience.
The data is anonymous and does not include commandline argument values.
The data is collected by Microsoft.

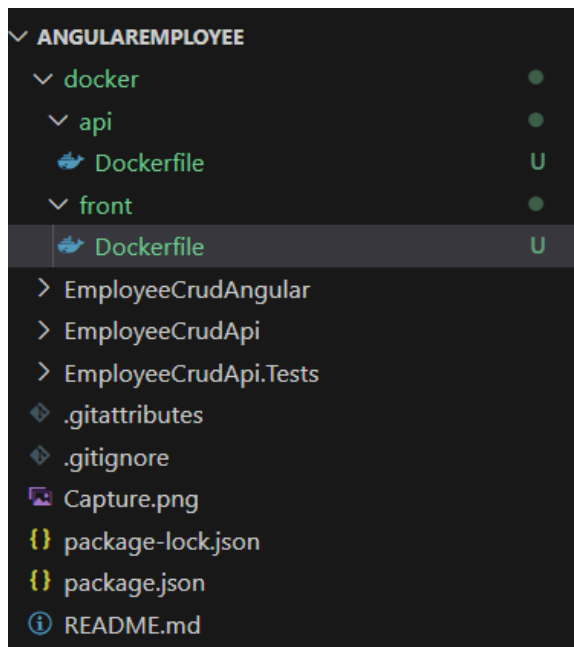
You can change your telemetry settings with `az configure`.

  /\
 /\  /\  /\  /\  /\  /\  /\  /\  /\  /\  /\  /\  /\  /\  /\  /\
/\  /\  /\  /\  /\  /\  /\  /\  /\  /\  /\  /\  /\  /\  /\  /\

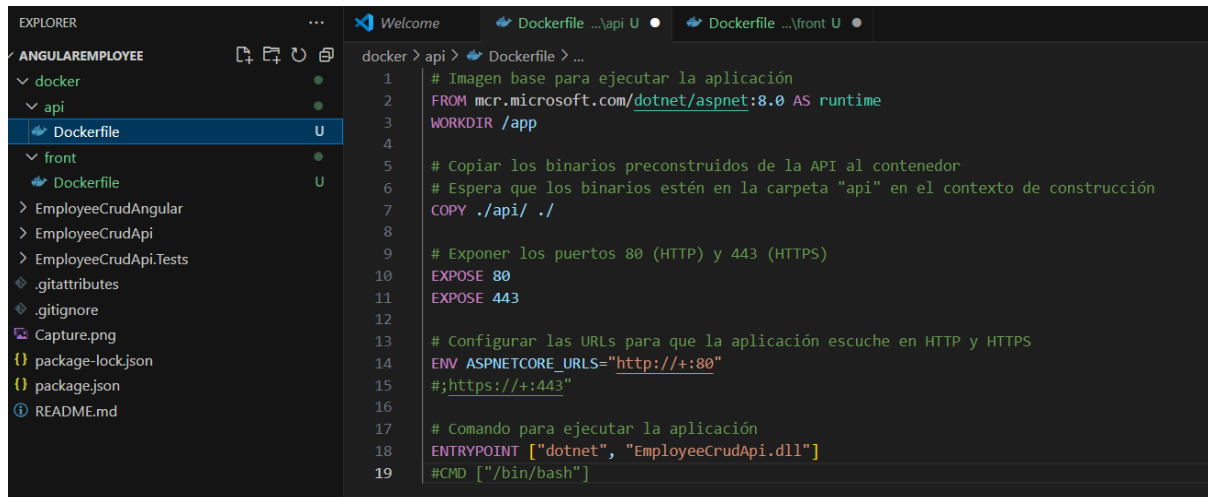
Welcome to the cool new Azure CLI!

Use `az --version` to display the current version.
Here are the base commands:
```

8.1 Se crea un directorio "docker" dentro del directorio raíz del proyecto que contiene 2 carpetas (api y front), y dentro de cada una de ellas se crean los correspondientes Dockerfile como se muestra a continuación.




Dockerfile Back:



The screenshot shows the Visual Studio Code interface with the Explorer on the left and the Dockerfile editor on the right. The Explorer shows a project named 'ANGULAREMPLOYEE' with a 'docker' folder containing 'api' and 'front' subfolders. The 'api' folder is selected, and its 'Dockerfile' is open in the editor. The Dockerfile content is as follows:

```
1 # Imagen base para ejecutar la aplicación
2 FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS runtime
3 WORKDIR /app
4
5 # Copiar los binarios preconstruídos de la API al contenedor
6 # Espera que los binarios estén en la carpeta "api" en el contexto de construcción
7 COPY ./api/ ./
8
9 # Exponer los puertos 80 (HTTP) y 443 (HTTPS)
10 EXPOSE 80
11 EXPOSE 443
12
13 # Configurar las URLs para que la aplicación escuche en HTTP y HTTPS
14 ENV ASPNETCORE_URLS="http://+:80"
15 #;https://+:443"
16
17 # Comando para ejecutar la aplicación
18 ENTRYPOINT ["dotnet", "EmployeeCrudApi.dll"]
19 #CMD ["/bin/bash"]
```

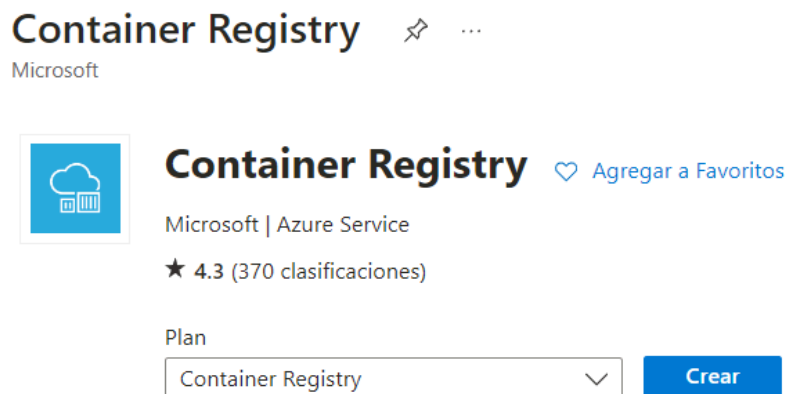
Dockerfile Front:



The screenshot shows the Visual Studio Code interface with the Explorer on the left and the Dockerfile editor on the right. The Explorer shows the same project structure as the previous screenshot. The 'front' folder is selected, and its 'Dockerfile' is open in the editor. The Dockerfile content is as follows:

```
1 # Utilizar la imagen base de nginx
2 FROM nginx:alpine
3
4 # Establecer el directorio de trabajo en el contenedor
5 WORKDIR /usr/share/nginx/html
6
7 # Eliminar los archivos existentes de la imagen base de nginx
8 RUN rm -rf /*
9
10 # Copiar los archivos compilados de Angular al directorio de nginx
11 COPY ./ .
12
13 # Exponer el puerto 80 para servir la aplicación Angular
14 EXPOSE 80
15 CMD sh -c 'echo "window[\"env\"] = { apiUrl: \"'${API_URL}'\" };\" > /usr/share/nginx/html/assets/env.js && nginx -g "daemon off;"'
```

- Crear un recurso ACR en Azure Portal: *(Siguiendo el instructivo)*



Detalles del proyecto

Suscripción *

Azure subscription 1

Grupo de recursos *

TP5IngSoft3UCC2024

[Crear nuevo](#)

Detalles de instancia

Nombre del Registro *

MRMIngSoft3UCCACR

✓

.azurecr.io

Ubicación *

Brazil South

Uso de zonas de disponibilidad ⓘ

ⓘ

 Las zonas de disponibilidad se activan en registros premium y en regiones que admiten zonas de disponibilidad. [Más información](#)

Plan de precios * ⓘ

Básico



Crear Registro de contenedor ...

ⓘ

 Ejecutando la validación final

- Datos básicos
- Redes
- Cifrado
- Etiquetas
- Revisar y crear

Detalles del registro

Datos básicos

Nombre del Registro	MRMIngSoft3UCCACR
Suscripción	Azure subscription 1
Grupo de recursos	TP5IngSoft3UCC2024
Ubicación	Brazil South
Zonas de disponibilidad	Deshabilitado
Plan de precios	Basic

Redes

Acceso de red pública	Sí
-----------------------	----

Cifrado

Clave administrada por el cliente	Deshabilitado
-----------------------------------	---------------

Inicio > Microsoft.ContainerRegistry | Información general ✕ ...

Implementación

Buscar ✕ « Eliminar Cancelar Volver a implementar Descargar Actualizar

Información general

- Entradas
- Salidas
- Plantilla

Se completó la implementación

Nombre de implementación : Microsoft.ContainerRegistry Hora de inicio : 14/10/2024, 14:53:21
 Suscripción : Azure subscription 1 Id. de correlación : 037d4d04-b0bc-40e4-9644-de7128656913
 Grupo de recursos : TPSIngSoft3UCC2024

> Detalles de implementación

▼ Pasos siguientes

[Ir al recurso](#)

Enviar comentarios

🗨 Cuéntenos su experiencia con la implementación

Administración de costos
 Obtenga una notificación para permanecer dentro del presupuesto y evitar cargos inesperados en su factura.
[Configurar alertas de costo >](#)

Microsoft Defender for Cloud
 Proteja sus aplicaciones e infraestructura.
[Ir a Microsoft Defender for Cloud >](#)

Tutoriales gratuitos de Microsoft
[Comience a aprender hoy >](#)

Trabajar con un experto

Copiar la url de nuestro recurso ACR: ***mrmingssoft3uccacr.azurecr.io***

- Modificar nuestro pipeline en la etapa de Build y Test
 - Luego de la tarea de publicación de los artefactos de **Back** agregar la tarea de publicación de nuestro dockerfile de back para que esté disponible en etapas posteriores:

```

87
88 - task: PublishBuildArtifacts@1
89   displayName: 'Publicar artefactos de compilación'
90   inputs:
91     PathToPublish: '$(Build.ArtifactStagingDirectory)'
92     ArtifactName: 'api-drop'
93     publishLocation: 'Container'
94
95
96 - task: PublishPipelineArtifact@1
97   displayName: 'Publicar Dockerfile de Back'
98   inputs:
99     targetPath: '$(Build.SourcesDirectory)/docker/api/dockerfile'
100    artifact: 'dockerfile-back'
101
102 - job: BuildAngular
103   displayName: "Build and Test Angular"
104   pool:
105     vmImage: 'ubuntu-latest'
106   steps:
107     - task: NodeTool@0
108       displayName: 'Instalar Node.js'
109       inputs:
110         versionSpec: '22.x'
111

```

- Luego de la tarea de publicación de los artefactos de **Front** agregar la tarea de publicación de nuestro dockerfile de front para que esté disponible en etapas posteriores:

```

- task: PublishBuildArtifacts@1
  displayName: 'Publicar artefactos Angular'
  inputs:
    PathToPublish: '$(frontPath)/dist'
    ArtifactName: 'front-drop'

- task: PublishPipelineArtifact@1
  displayName: 'Publicar Dockerfile de Back'
  inputs:
    targetPath: '$(Build.SourcesDirectory)/docker/front/dockerfile'
    artifact: 'dockerfile-front'

```

En caso de no contar en nuestro proyecto con una ServiceConnection a Azure Portal para el manejo de recursos, agregar una service connection a Azure Resource Manager como se indica en instructivo 5.2

New Azure service connection

Azure Resource Manager using Workload Identity federation with OpenID Connect (automatic)

Scope level

☒ Subscription
☐ Management Group
☐ Machine Learning Workspace

Subscription

Azure subscription 1 (5a496d64-56ee-4be3-9f0d-75685209f7a...)

Resource group

TP5IngSoft3UCC2024

Details

Service connection name

ServiceConnectionARM

Service Management Reference (optional)

Description (optional)

Security

☒ Grant access permission to all pipelines

[Learn more](#)
[Troubleshoot](#)

[Back](#) [Save](#)

ServiceConnectionARM

[Edit](#)

Overview Usage history Approvals and checks

Details

Service connection type

Azure Resource Manager
using workload identity federation with openid connect
[Manage service connection roles](#)

Creator

Manuel Romero Medina
manuromedina12345@gmail.com

Agregar a nuestro pipeline variables

```
trigger:
- main

pool:
- vmImage: 'windows-latest'

variables:
- solution: '**/*.sln'
- buildPlatform: 'Any CPU'
- buildConfiguration: 'Release'
- frontPath: './EmployeeCrudAngular'
- backPath: './EmployeeCrudApi'
- ConnectedServiceName: 'ServiceConnectionARM' #Por ejemplo 'ServiceConnectionARM'
- acrLoginServer: 'mrmingsoft3uccacr.azurecr.io' #Por ejemplo 'ascontainerregistry.azurecr.io'
- backImageName: 'employee-crud-api' #Por ejemplo 'employee-crud-api'
```

Agregar a nuestro pipeline una nueva etapa que dependa de nuestra etapa de Build y Test

- Agregar tareas para generar imagen Docker de Back:

```
155 # #
156 # ### STAGE: BUILD DOCKER IMAGES Y PUSH A AZURE CONTAINER REGISTRY
157 # #
158
159 --stage: DockerBuildAndPush
160 --displayName: 'Construir y Subir Imágenes Docker a ACR'
161 --dependsOn: BuildAndTest #NOMBRE DE NUESTRA ETAPA DE BUILD Y TEST
162 --jobs:
163 --  - job: docker_build_and_push
164 --    --displayName: 'Construir y Subir Imágenes Docker a ACR'
165 --    --pool:
166 --      --vmImage: 'ubuntu-latest'
167 --    --steps:
168 --      -- - checkout: self
169 --      --
170 --      -- #
171 --      -- # BUILD DOCKER BACK IMAGE Y PUSH A AZURE CONTAINER REGISTRY
172 --      -- #
173 --      --
174
175 --      -- Settings
176 --      -- - task: DownloadPipelineArtifact@2
177 --      --      -- displayName: 'Descargar Artefactos de Back'
178 --      --      -- inputs:
179 --      --      --   buildType: 'current'
180 --      --      --   artifactName: 'drop-back'
181 --      --      --   targetPath: '$(Pipeline.Workspace)/drop-back'
182
183 --      -- Settings
184 --      -- - task: DownloadPipelineArtifact@2
185 --      --      -- displayName: 'Descargar Dockerfile de Back'
186 --      --      -- inputs:
187 --      --      --   buildType: 'current'
188 --      --      --   artifactName: 'dockerfile-back'
189 --      --      --   targetPath: '$(Pipeline.Workspace)/dockerfile-back'
```

Ejecutar el pipeline y en Azure Portal acceder a la opción Repositorios de nuestro recurso Azure Container Registry. Verificar que exista una imagen con el nombre especificado en la variable backImageName asignada en nuestro pipeline.

Stages

Jobs

✓ Build and Test API an...

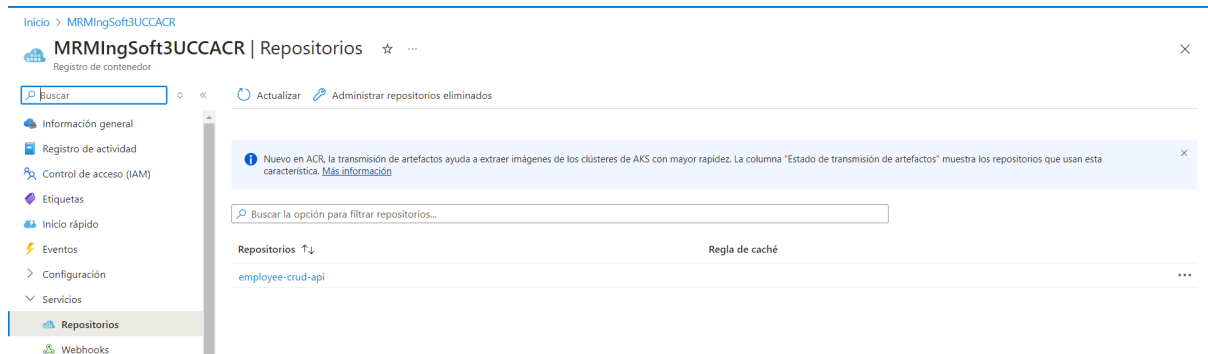
2 jobs completed4m 59s

100% tests passed

4 artifacts

✓ Construir y Subir Imá...

1 job completed50s



Agregar a nuestro pipeline una nueva etapa que dependa de nuestra etapa de Construcción de Imágenes Docker y subida a ACR

- Agregar variables a nuestro pipeline:

ResourceGroupName: 'NOMBRE_GRUPO_RECURSOS' #Por ejemplo 'TPS_INGSOFT3_UCC'
backContainerInstanceNameQA: 'NOMBRE_CONTAINER_BACK_QA' #Por ejemplo 'as-crud-api-qa'
backImageTag: 'latest'

container-cpu-api-qa: 1 #CPUS de nuestro container de QA

container-memory-api-qa: 1.5 #RAM de nuestro container de QA

← New variable

Name

cnn_string_qa

Value

.....

☒ Keep this value secret

☐ Let users override this value when running this pipeline

To reference a variable in YAML, prefix it with a dollar sign and enclose it in parentheses. For example: `$(cnn_string_qa)`

To use a variable in a script, use environment variable syntax.

Replace `.` and space with `_`, capitalize the letters, and then use your platform's syntax for referencing an environment variable. Examples:

Batch script: `%CNN_STRING_QA%`

PowerShell script: `${env:CNN_STRING_QA}`

Bash script: `$(CNN_STRING_QA)`

To use a secret variable in a script, you must explicitly map it as an environment variable.

Server=tcp:mrmsql.database.windows.net,1433;Initial Catalog=mrmsql;Persist

Security Info=False;User

ID=sqladmin;Password=Manuomero1;MultipleActiveResultSets=False;Encrypt=Tru

e;TrustServerCertificate=False;Connection Timeout=30;

```

21     - frontImageName: 'employee-crud-front'
22     - ResourceGroupName: 'TP5IngSoft3UCC2024' #Por ejemplo 'TPS_INGSOFT3_UCC'
23     - backContainerInstanceNameQA: 'mrm-crud-api-qa' #Por ejemplo 'as-crud-api-qa'
24     - backImageTag: 'latest'
25     - container-cpu-api-qa: 1 #CPUS de nuestro container de QA
26     - container-memory-api-qa: 1.5 #RAM de nuestro container de QA
27

```

Tenemos que modificar program.cs para que lea la variable de entorno y use esa cadena de conexión

```

// Obtener la cadena de conexión de una variable de entorno o de la configuración
var connectionString = Environment.GetEnvironmentVariable("cnn_string_qa")
| | | | | ?? builder.Configuration.GetConnectionString("DefaultConnection");

if (string.IsNullOrEmpty(connectionString))
{
    throw new InvalidOperationException("La cadena de conexión no está configurada.");
}

builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(connectionString));

```

- Agregar tareas para crear un recurso Azure Container Instances que levante un contenedor con nuestra imagen de back

```

#####
### STAGE DEPLOY TO ACI QA
#####
- stage: DeployToACIQA
- displayName: 'Desplegar en Azure Container Instances (ACI) QA'
- dependsOn: DockerBuildAndPush
- jobs:
  - job: deploy_to_aci_qa
    displayName: 'Desplegar en Azure Container Instances (ACI) QA'
    pool:
      vmImage: 'ubuntu-latest'

    steps:
      - # -----
        # DEPLOY DOCKER BACK IMAGE A AZURE CONTAINER INSTANCES QA
        # -----

        Settings
        - task: AzureCLI@2
          displayName: 'Desplegar Imagen Docker de Back en ACI QA'
          inputs:
            azureSubscription: '$(ConnectedServiceName)'
            scriptType: bash
            scriptLocation: inlineScript
            inlineScript: |
              echo "Resource Group: $(ResourceGroupName)"
              echo "Container Instance Name: $(backContainerInstanceNameQA)"
              echo "ACR Login Server: $(acrLoginServer)"
              echo "Image Name: $(backImageName)"
              echo "Image Tag: $(backImageTag)"
              echo "Connection String: $(cnn-string-qa)"

              az container delete --resource-group $(ResourceGroupName) --name $(backContainerInstanceNameQA) --yes

              az container create --resource-group $(ResourceGroupName) \
                --name $(backContainerInstanceNameQA) \

```

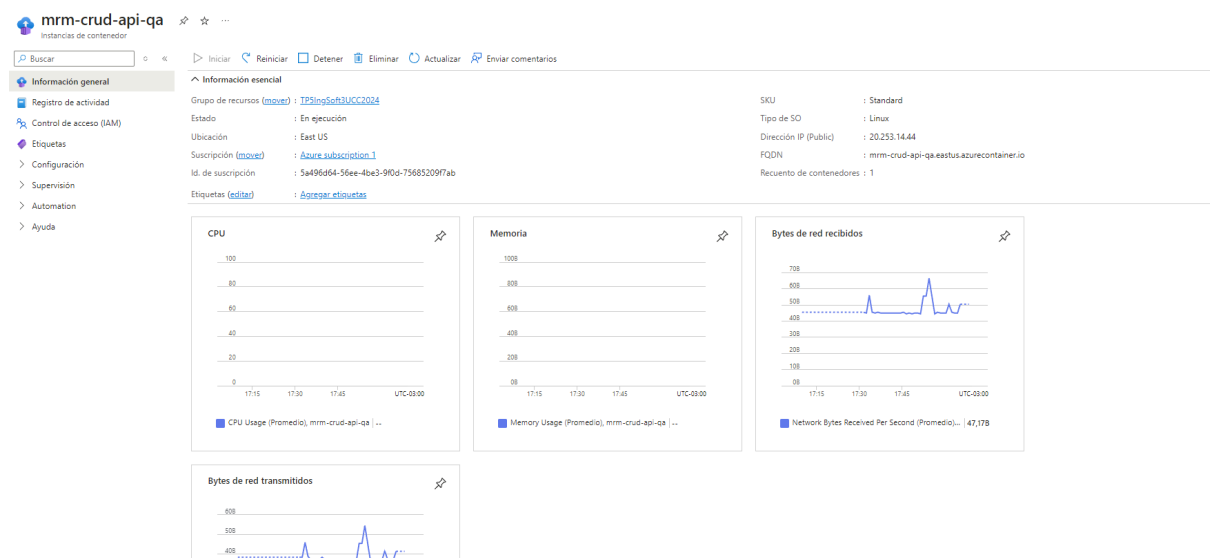
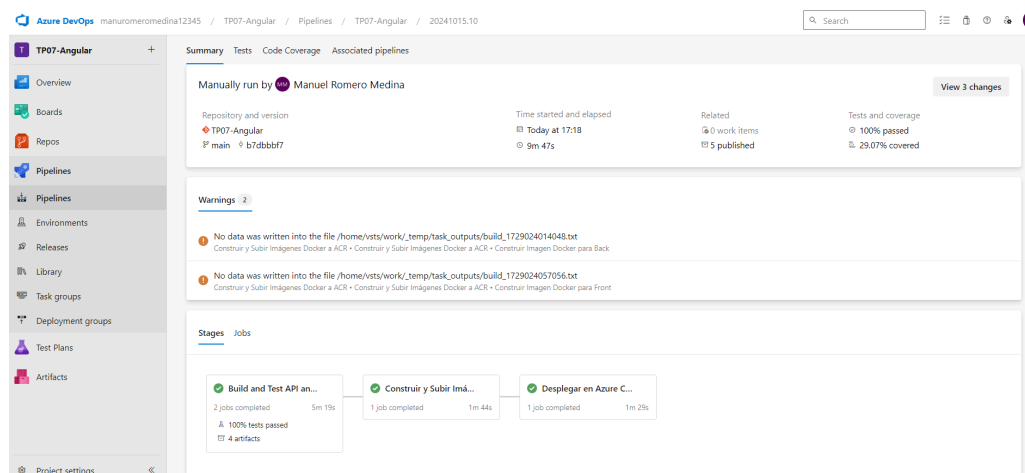

Para que funcione, hay que habilitar el acceso administrativo de ACR desde azure CLI

```
PS C:\Users\Manu> az acr update -n MRMIingSoft3UCCACR --admin-enabled true
>>
{
  "adminUserEnabled": true,
  "anonymousPullEnabled": false,
  "creationDate": "2024-10-14T17:53:22.687314+00:00",
  "dataEndpointEnabled": false,
  "dataEndpointHostNames": [],
  "encryption": {
    "keyVaultProperties": null,
    "status": "disabled"
  },
  "id": "/subscriptions/5a496d64-56ee-4be3-9f0d-75685209f7ab/resourceGroups/TP5IngSoft3UCC2024/providers/Microsoft.ContainerRegistry/registries/MRMIingSoft3UCCACR",
  "identity": null,
  "location": "brazilsouth",
  "loginServer": "mrmingsoft3uccacr.azurecr.io",
  "metadataSearch": "Disabled",
  "name": "MRMIingSoft3UCCACR",
  "networkRuleBypassOptions": "AzureServices",
  "networkRuleSet": null,
}
```

Además, debemos registrar el proveedor de recursos Microsoft.ContainerInstance, que es necesario para crear instancias de contenedores en Azure.

```
PS C:\Users\Manu> az provider register --namespace Microsoft.ContainerInstance
>>
Registering is still on-going. You can monitor using 'az provider show -n Microsoft.ContainerInstance'
```

8.10 - Ejecutar el pipeline y en Azure Portal acceder al recurso de Azure Container Instances creado. Copiar la url del contenedor y navegarlo desde browser. Verificar que traiga datos.



▼ Create or update Container Group	Correcto	hace 9 minu...	Tue Oct 15 ...	Azure subscription 1	manuromeromedina1234...
Create or update Container Group	Iniciado	hace 10 min...	Tue Oct 15 ...	Azure subscription 1	manuromeromedina1234...
Create or update Container Group	Accepted	hace 10 min...	Tue Oct 15 ...	Azure subscription 1	manuromeromedina1234...
Get Container Groups	Running	hace 10 min...	Tue Oct 15 ...	Azure subscription 1	manuromeromedina1234...
Get Container Groups	Running	hace 9 minu...	Tue Oct 15 ...	Azure subscription 1	manuromeromedina1234...

Pretty-print

```
[
  {
    "id": 1,
    "name": "Juan Perez",
    "createdDate": "2024-09-05T00:00:00"
  },
  {
    "id": 2,
    "name": "Carla Ruiz",
    "createdDate": "2024-09-05T22:22:47.440572"
  },
  {
    "id": 3,
    "name": "Carlos Gomez",
    "createdDate": "2024-09-06T09:16:50.070943"
  },
  {
    "id": 4,
    "name": "Joaquin Zarate",
    "createdDate": "2024-09-06T09:19:37.898716"
  },
  {
    "id": 5,
    "name": "Luis Rodriguez",
    "createdDate": "2024-09-06T09:23:31.324434"
  }
]
```

DESAFÍOS

1. Agregar tareas para generar imagen Docker de Front

Agregar tareas para generar imagen Docker de Front (DESAFIO)

- A la etapa creada en 4.1.6 Agregar tareas para generar imagen Docker de Front

```
- backImageName: 'employee-crud-api' #Por  
- frontImageName: 'employee-crud-front'
```

```
#-----  
# BUILD DOCKER FRONT IMAGE Y PUSH A AZURE CONTAINER REGISTRY  
#-----  
  
Settings  
- task: DownloadPipelineArtifact@2  
  displayName: 'Descargar Artefactos de Front'  
  inputs:  
    buildType: 'current'  
    artifactName: 'front-drop'  
    targetPath: '$(Pipeline.Workspace)/drop-front'  
  
Settings  
- task: DownloadPipelineArtifact@2  
  displayName: 'Descargar Dockerfile de Front'  
  inputs:  
    buildType: 'current'  
    artifactName: 'dockerfile-front'  
    targetPath: '$(Pipeline.Workspace)/dockerfile-front'  
  
Settings  
- task: Docker@2  
  displayName: 'Construir Imagen Docker para Front'  
  inputs:  
    command: build  
    repository: $(acrLoginServer)/employee-crud-front # Cambia el nombre de la imagen según sea necesario  
    dockerfile: $(Pipeline.Workspace)/dockerfile-front/Dockerfile  
    buildContext: $(Pipeline.Workspace)/drop-front  
    tags: 'latest'  
  
Settings  
- task: Docker@2  
  displayName: 'Subir Imagen Docker de Front a ACR'  
  inputs:  
    command: push  
    repository: $(acrLoginServer)/employee-crud-front # Cambia el nombre de la imagen según sea necesario  
    tags: 'latest'
```

← Jobs in run #20241014.24

> ✓ Build and Test API2m 32s

> ✓ Build and Test Ang...2m 9s

Construir y Subir Imágenes Docker a A...

▼ ✓ Construir y Subir I...1m 31s

✓ Initialize job8s

✓ Checkout TP07-Ang...4s

✓ Descargar Artefactos...6s

✓ Descargar Dockerfile...4s

✓ Iniciar Sesión en Az...15s

✓ Construir Imagen Do...6s

✓ Subir Imagen Dock...13s

✓ Descargar Artefactos...5s

✓ Descargar Dockerfile...5s

✓ Construir Imagen Do...5s

✓ Subir Imagen Dock...15s

✓ Post-job: Checkout...<1s

✓ Finalize Job<1s

✓ Construir Imagen Docker para Front

137 #2 DONE 0.0s
138
139 #3 [internal] load .dockerignore
140 #3 transferring context: 2B done
141 #3 DONE 0.0s
142
143 #4 [internal] load build context
144 #4 transferring context: 1.65MB 0.0s done
145 #4 DONE 0.0s
146
147 #5 [1/4] FROM docker.io/library/nginx:alpine
148 #5 DONE 0.1s
149
150 #6 [2/4] WORKDIR /usr/share/nginx/html
151 #6 DONE 0.0s
152
153 #7 [3/4] RUN rm -rf /*
154 #7 DONE 0.2s
155
156 #8 [4/4] COPY ./ .
157 #8 DONE 0.0s
158
159 #9 exporting to image
160 #9 exporting layers
161 #9 exporting layers 1.0s done
162 #9 writing image sha256:108267c243b578344250feba2f969e0f2f28b585c7b4392b291c87feb6aea52d done
163 #9 naming to mrmingsoft3uccacr.azurecr.io/employee-crud-front:latest done
164 #9 DONE 1.0s
165 ##[warning]No data was written into the file /home/vsts/work/_temp/task_outputs/build_1728944885856.txt
166 Finishing: Construir Imagen Docker para Front

Inicio > MRMIngSoft3UCCACR

MRMIngSoft3UCCACR | Repositorios

☆ ...

Registro de contenedor

🔍 Buscar

⌵ <<

🔄 Actualizar

🔗 Administrar repositorios eliminados

📁 Información general

📅 Registro de actividad

🔑 Control de acceso (IAM)

🏷️ Etiquetas

🚀 Inicio rápido

⚡ Eventos

> Configuración

▼ Servicios

📦 Repositorios

🔗 Webhooks

🔄 Replicaciones

📘 Nuevo en ACR, la transmisión de artefactos ayuda a extraer imágenes de los clústeres de AKS con mayor rapidez. La columna "Estado de transmisión de artefactos" es una característica. [Más información](#)

🔍 Buscar la opción para filtrar repositorios...

Repositorios ↑↓	Regla de caché
employee-crud-api	
employee-crud-front	

2. Agregar tareas para generar en Azure Container Instances un contenedor de imagen Docker de Front. (Complemento con 4.1.11)

- Creamos la variable

← Update variable

Name

API_URL

Value

mrm-crud-api-qa.eastus.azurecontainer.io/api/employee/getall

☐ Keep this value secret

☐ Let users override this value when running this pipeline

To reference a variable in YAML, prefix it with a dollar sign and enclose it in parentheses. For example: `$(API_URL)`

To use a variable in a script, use environment variable syntax.

Replace `.` and space with `_`, capitalize the letters, and then use your platform's syntax for referencing an environment variable. Examples:

Batch script: `%API_URL%`

PowerShell script: `${env:API_URL}`

Bash script: `$(API_URL)`

mrm-crud-api-qa.eastus.azurecontainer.io/api/employee/getall

- Instalamos lo necesario

```
C:\Users\Wan\Desktop\INGENIERIA EN SISTEMAS UC\CUARTO AÑO\INGENIERIA EN SISTEMAS UC\TP8\AngularEmployee\EmployeeCrudAngular>npm install @angular/cli --save-dev
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerful.
npm warn deprecated read-package-json@7.0.1: This package is no longer supported. Please use @npmcli/package-json instead.
npm warn deprecated rimraf@3.0.2: Rimraf versions prior to v4 are no longer supported
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported

added 1058 packages, and audited 1051 packages in 52s

132 packages are looking for funding
  run `npm fund` for details

4 low severity vulnerabilities
To address all issues, run:
  npm audit fix
```

- Modificamos el código del FrontEnd

```
TS environment.ts M X TS environment.prod.ts ●
EmployeeCrudAngular > src > environments > TS environment.ts > [e] environment > apiUrl
1 // This file can be replaced during build by using the `fileReplacements` array.
2 // `ng build --prod` replaces `environment.ts` with `environment.prod.ts`.
3 // The list of file replacements can be found in `angular.json`.
4
5 export const environment = {
6   production: false,
7   apiUrl: 'mrm-crud-api-qa.eastus.azurecontainer.io/api/employee' // URL de la API para desarrollo
8 };
9
10
11 /*
12  * For easier debugging in development mode, you can import the following file
13  * to ignore zone related error stack frames such as `zone.run`, `zoneDelegate.invokeTask`.
14  *
15  * This import should be commented out in production mode because it will have a negative impact
16  * on performance if an error is thrown.
17  */
18 // import 'zone.js/dist/zone-error'; // Included with Angular CLI.
19
20

TS environment.ts M TS environment.prod.ts M X
EmployeeCrudAngular > src > environments > TS environment.prod.ts > ...
1 export const environment = {
2   production: true,
3   apiUrl: 'mrm-crud-api-qa.eastus.azurecontainer.io/api/employee' // URL de la API para prod
4 };
5
```

test.ts

[Edit](#)[Contents](#) [History](#) [Compare](#) [Blame](#)

```
1 // Este archivo es requerido por karma.conf.js y carga recursivamente todos los archivos .spec y del marco.
2 (window as any).__env = (window as any).__env || {};
3 (window as any).__env['apiUrl'] = window['__env'] && window['__env']['apiUrl'] || 'mrm-crud-api-qa.eastus.azurecontainer.io/api/employee'; // Valor pr
4
5 import 'zone.js/dist/zone-testing';
6 import { getTestBed } from '@angular/core/testing';
7 import { BrowserDynamicTestingModule } from '@angular/platform-browser-dynamic/testing';
8
9 // Primero, inicializa el entorno de pruebas de Angular.
10 getTestBed().initTestingModule({
11   BrowserDynamicTestingModule,
12   platformBrowserDynamicTesting(),
13 });
14
15 // Luego, encontraremos todas las pruebas.
16 const context = require.context('./', true, /\.spec\.ts$/);
17 context.keys().forEach(context);
18
```

index.html

[Contents](#) [Preview](#) [Highlight changes](#)

Committed 9c5b50d4: Updated test.ts

```
1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <title>EmployeeCrudAngular</title>
6     <base href="/">
7     <meta name="viewport" content="width=device-width, initial-scale=1">
8     <link rel="icon" type="image/x-icon" href="favicon.ico">
9   </head>
10  <body>
11    <script src="assets/env.js"></script>
12    <script>
13      console.log('Valor de API_URLs:', window['__env'] && window['__env'].apiUrl);
14    </script>
15    <app-root></app-root>
16  </body>
17 </html>
18
```

[TS environment.ts](#) M[TS environment.prod.ts](#) M[TS global.d.ts](#) U X

EmployeeCrudAngular > src > environments > TS global.d.ts > Window

```
1 interface Window {
2   |   env : {
3     |     apiUrl: string
4   |   }
5 }
```

Agregamos las siguientes variables de entorno

```
frontContainerInstanceNameQA: 'mrm-crud-front-qa' # Por ejemplo, cambiar según tu entorno
container-cpu-front-qa: 1
container-memory-front-qa: 1.5
frontImageTag: 'latest'
```

Agregamos las tareas para que levante el contenedor de front

```
#####
# DEPLOY DOCKER FRONT IMAGE A AZURE CONTAINER INSTANCES QA
#####

Settings
- task: AzureCLI@2
  displayName: 'Desplegar Imagen Docker de Front en ACI QA'
  inputs:
    azureSubscription: '$(ConnectedServiceName)'
    scriptType: bash
    scriptLocation: inlineScript
    inlineScript: |
      echo "Resource Group: $(ResourceGroupName)"
      echo "Container Instance Name: $(frontContainerInstanceNameQA)" # Debes agregar la variable para el front
      echo "ACR Login Server: $(acrLoginServer)"
      echo "Image Name: $(frontImageName)"
      echo "Image Tag: $(frontImageTag)"
      echo "Api Url: $(API_URL)"

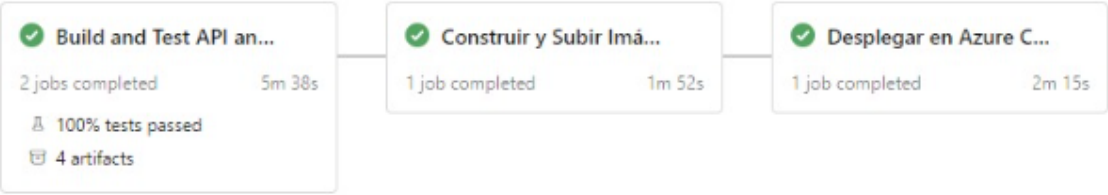
      az container delete --resource-group $(ResourceGroupName) --name $(frontContainerInstanceNameQA) --yes

      az container create --resource-group $(ResourceGroupName) \
        --name $(frontContainerInstanceNameQA) \
        --image $(acrLoginServer)/$(frontImageName):$(frontImageTag) \
        --registry-login-server $(acrLoginServer) \
        --registry-username $(acrName) \
        --registry-password $(az acr credential show --name $(acrName) --query "passwords[0].value" --o tsv) \
        --dns-name-label $(frontContainerInstanceNameQA) \
        --ports 80 \
        --environment-variables API_URL="$(API_URL)" \
        --restart-policy Always \
        --cpu $(container-cpu-front-qa) \
        --memory $(container-memory-front-qa)
```

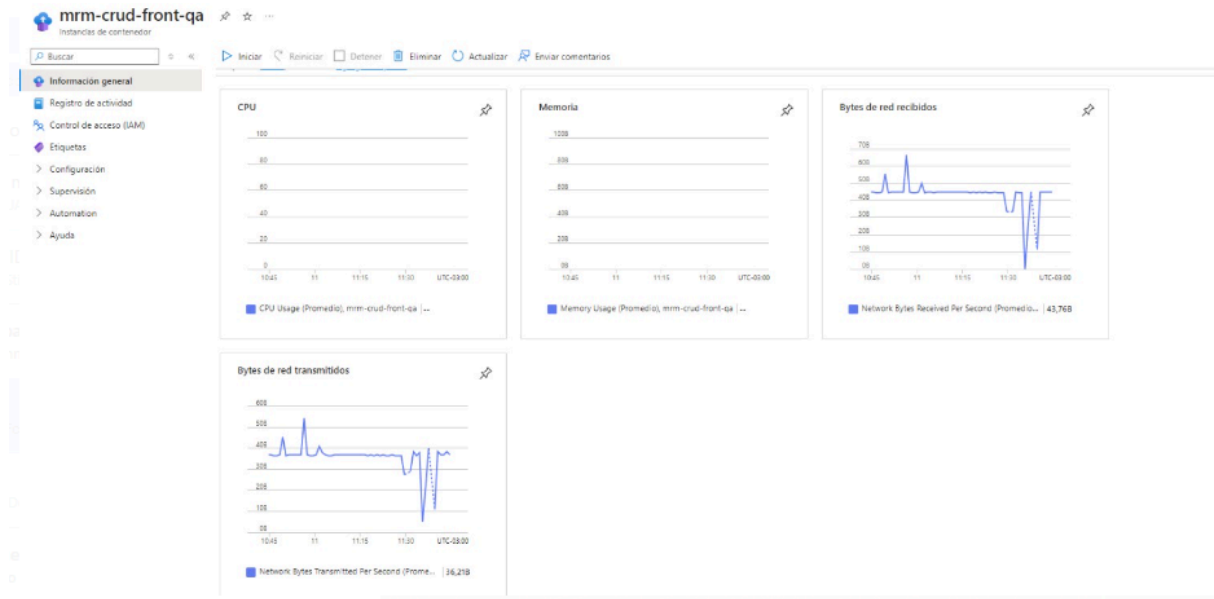
- Corremos el pipeline

The screenshot displays the Azure DevOps web interface. On the left, a sidebar shows navigation options: Overview, Boards, Repos, Pipelines (selected), Environments, Releases, Library, Task groups, and Deployment groups. The main area is titled 'Jobs in run #20241016.15' and lists several pipeline jobs with their status and duration. The job 'Desplegar en Azure Container Instances (ACI) QA' is highlighted, showing a status of 'Succeeded' and a duration of '2m 9s'. To the right, a detailed view of this job is shown, including a 'View raw log' button and a log of the execution steps. The log shows the job starting at 8:42, the agent being hosted, and the container being deployed successfully.

Job Name	Status	Duration
Build and Test API	Succeeded	3m 7s
Build and Test Ang...	Succeeded	2m 8s
Construir y Subir Imágenes Docker a ACR	Succeeded	
Construir y Subir I...	Succeeded	1m 45s
Desplegar en Azure Container Instance...	Succeeded	
Desplegar en Azure...	Succeeded	2m 9s
Initialize job	Succeeded	1s
Checkout TP07-Ang...	Succeeded	2s
Desplegar Imag...	Succeeded	1m 20s
Desplegar Imagen ...	Succeeded	43s
Post-job: Checkout ...	Succeeded	<1s
Finalize Job	Succeeded	<1s



• Comprobamos



3. Agregar tareas para correr pruebas de integración en el entorno de QA de Back y Front creado en ACI. (Complemento con 4.1.12)

- Modificamos el pipeline con un “Stage Cypress” (VER PORQUE NO ME EJECUTAN LAS PRUEBAS)

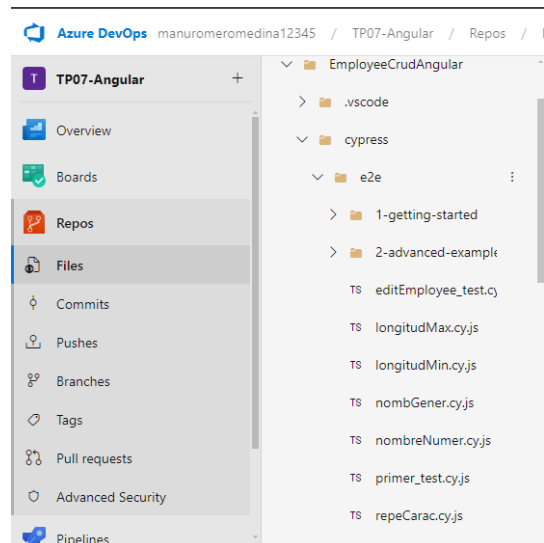
```
# STAGE: CYPRESS TESTING
- stage: CypressTesting
  displayName: 'Pruebas E2E con Cypress'
  jobs:
  - job: IntegrationTesting
    displayName: 'Cypress'
    variables:
    - baseUrl: '${(frontContainerInstanceNameQA)}.eastus.azurecontainer.io'
    steps:
    - script: |
        cd $(Build.SourcesDirectory)/EmployeeCrudAngular
        npm install typescript ts-node
        displayName: 'Instalar TypeScript'

    - script: |
        if not exist $(Build.SourcesDirectory)/EmployeeCrudAngular/cypress/results mkdir "$(Build.SourcesDirectory)/EmployeeCrudAngular/cypress/results"
        displayName: 'Crear el directorio de resultados'

    - script: |
        cd $(Build.SourcesDirectory)/EmployeeCrudAngular
        npx cypress run --config-file cypress.config.ts --env baseUrl=$(baseUrl)
        displayName: 'Ejecutar Pruebas E2E con Cypress'

Settings
- task: PublishTestResults@2
  inputs:
  - testResultsFiles: '$(Build.SourcesDirectory)/EmployeeCrudAngular/cypress/results/*.xml'
  - testRunTitle: 'Pruebas E2E con Cypress (QA)'
  - displayName: 'Publicar Resultados de las Pruebas de Cypress'
```

- Creamos pruebas



- Longitud máxima de caracteres:

```
EmployeeCrudAngular > cypress > e2e > JS longitudMax.cy.js > describe('Validación: longitud máxima de caracteres') callback > it('Debe mostrar un mensaje de error cuando el nombre excede los 100 caracteres', () => {
1 describe('Validación: longitud máxima de caracteres', () => {
2   it('Debe mostrar un mensaje de error cuando el nombre excede los 100 caracteres', () => {
3     const apiUrl = Cypress.env('baseUrl'); // Obtener la URL desde las variables de entorno
4
5     // Verificar si apiUrl está definido
6     if (!apiUrl) {
7       throw new Error('Error: apiUrl is not defined. Please check your environment variables.');
```

- Longitud mínima de caracteres:

```
EmployeeCrudAngular > cypress > e2e > JS longitudMin.cy.js > ...
1 describe('Validación: Longitud mínima del nombre', () => {
2   it('Debe mostrar un mensaje de error cuando el nombre tiene menos de 2 caracteres', () => {
3     const apiUrl = Cypress.env('baseUrl'); // Obtener la URL desde las variables de entorno
4
5     // Verificar si apiUrl está definido
6     if (!apiUrl) {
7       throw new Error('Error: apiUrl is not defined. Please check your environment variables.');
```

- Nombre con números:

```
EmployeeCrudAngular > cypress > e2e > JS nombreNumer.cy.js > ...
1 describe('Validación: Nombre contiene números', () => {
2   it('Debe mostrar un mensaje de error cuando el nombre contiene números', () => {
3     const baseUrl = Cypress.env('baseUrl'); // Obtener la URL desde las variables de entorno
4
5     // Verificar si baseUrl está definido
6     if (!baseUrl) {
7       throw new Error('Error: baseUrl is not defined. Please check your environment variables.');
```

- Repeticiones de caracteres:

```
EmployeeCrudAngular > cypress > e2e > JS repeCarac.cy.js > ...
1 describe('Validación: Repetición de caracteres del nombre', () => {
2   it('Debe mostrar un mensaje de error cuando el nombre contiene caracteres repetidos excesivamente', () => {
3     const apiUrl = Cypress.env('baseUrl'); // Obtener la URL desde las variables de entorno
4
5     // Verificar si apiUrl está definido
6     if (!apiUrl) {
7       throw new Error('Error: apiUrl is not defined. Please check your environment variables.');
```

- Nombre trivial o genérico:

```
EmployeeCrudAngular > cypress > e2e > JS nombGener.cy.js > ...
1 describe('Validación: Nombre genérico o trivial', () => {
2   it('Debe mostrar un mensaje de error cuando el nombre es genérico o trivial', () => {
3     const apiUrl = Cypress.env('baseUrl'); // Obtener la URL desde las variables de entorno
4
5     // Verificar si apiUrl está definido
6     if (!apiUrl) {
7       throw new Error('Error: apiUrl is not defined. Please check your environment variables.');
```

4. Agregar etapa que dependa de la etapa de Deploy en ACI QA y genere contenedores en ACI para entorno de PROD.

```
- backContainerInstanceNameProd: 'mrm-container-back-prod'
- frontContainerInstanceNameProd: 'mrm-container-front-prod'
```

BACK PROD:

```
297 #STAGE BACK Y FRONT PROD
298
299 --stage: DeployToACIPROD
300 --displayName: 'Desplegar PROD'
301 --dependson:
302 -- --DeployToACIQA
303 --jobs:
304 -- --deployment: DeployToProd
305 -- --displayName: 'Desplegar PROD'
306 -- --environment:
307 -- -- --name: 'Production'
308 -- --strategy:
309 -- -- --runOnce:
310 -- -- -- --deploy:
311 -- -- -- --steps:
312 -- -- -- --Settings
313 -- -- -- -- --task: AzureCLI@2
314 -- -- -- -- --displayName: 'Desplegar Imagen Docker de Back en ACI (Prod)'
315 -- -- -- -- --inputs:
316 -- -- -- -- -- --azureSubscription: '${ConnectedServiceName}'
317 -- -- -- -- -- --scriptType: bash
318 -- -- -- -- -- --scriptLocation: inlineScript
319 -- -- -- -- -- --inlineScript: |
320 -- -- -- -- -- --echo "Resource Group: $(ResourceGroupName)"
321 -- -- -- -- -- --echo "Container Instance Name: $(backContainerInstanceNameProd)"
322 -- -- -- -- -- --echo "ACR Login Server: $(acrLoginServer)"
323 -- -- -- -- -- --echo "Image Name: $(backImageName)"
324 -- -- -- -- -- --echo "Image Tag: $(backImageTag)"
325 -- -- -- -- -- --echo "Connection String: $(cnn-string-qa)"
326 -- -- -- -- -- --az container delete --resource-group $(ResourceGroupName) --name $(backContainerInstanceNameProd) --yes
327 -- -- -- -- -- --az container create --resource-group $(ResourceGroupName) \
328 -- -- -- -- -- -- --name $(backContainerInstanceNameProd) \
329 -- -- -- -- -- -- --image $(acrLoginServer)/$(backImageName):$(backImageTag) \
330 -- -- -- -- -- -- --registry-login-server $(acrLoginServer) \
331 -- -- -- -- -- -- --registry-username $(acrName) \
332 -- -- -- -- -- -- --registry-password $(az acr credential show --name $(acrName) --query "passwords[0].value" --o tsv) \
333 -- -- -- -- -- -- --dns-name-label $(backContainerInstanceNameProd) \
334 -- -- -- -- -- -- --ports 80 \
335 -- -- -- -- -- -- --environment-variables ConnectionStrings__DefaultConnection="$(cnn-string-qa)" \
336 -- -- -- -- -- -- --restart-policy Always \
337 -- -- -- -- -- -- --cpu $(container-cpu-api-qa) \
338 -- -- -- -- -- -- --memory $(container-memory-api-qa)
```

FRONT PROD:

```
Settings
340 .....task: AzureCLI@2
341 .....displayName: 'Desplegar Imagen Docker de Front en ACI (PROD)'
342 .....inputs:
343 .....  azureSubscription: '$(ConnectedServiceName)'
344 .....  scriptType: bash
345 .....  scriptLocation: inlineScript
346 .....  inlineScript: |-
347 .....    echo "Resource Group: $(ResourceGroupName)"
348 .....    echo "Container Instance Name: $(frontContainerInstanceNameProd)"
349 .....    echo "ACR Login Server: $(acrLoginServer)"
350 .....    echo "Image Name: $(frontImageName)"
351 .....    echo "Image Tag: $(frontImageTag)"
352 .....    echo "API URL: $(PROD)"
353 .....
354 .....    az container delete --resource-group $(ResourceGroupName) --name $(frontContainerInstanceNameProd) --yes
355 .....
356 .....    az container create --resource-group $(ResourceGroupName) \
357 .....      --name $(frontContainerInstanceNameProd) \
358 .....      --image $(acrLoginServer)/$(frontImageName):$(frontImageTag) \
359 .....      --registry-login-server $(acrLoginServer) \
360 .....      --registry-username $(acrName) \
361 .....      --registry-password $(az acr credential show --name $(acrName) --query "passwords[0].value" --o tsv) \
362 .....      --dns-name-label $(frontContainerInstanceNameProd) \
363 .....      --ports 80 \
364 .....      --environment-variables API_URL="$(PROD)" \
365 .....      --restart-policy Always \
366 .....      --cpu $(container-cpu-front-qa) \
367 .....      --memory $(container-memory-front-qa)
```

Checks and manual validations for Desplegar PROD



Permission Environment **Production**

Permission needed

Permit

✓ Build and Test API an...

2 jobs completed 5m 51s

100% tests passed

4 artifacts

✓ Construir y Subir Imá...

1 job completed 1m 40s

✓ Desplegar en Azure C...

1 job completed 1m 57s

✓ Desplegar PROD

1 job completed 3m 7s



mrm-container-back-prod



mrm-container-front-prod



mrm-crud-api-qa



mrm-crud-front-qa