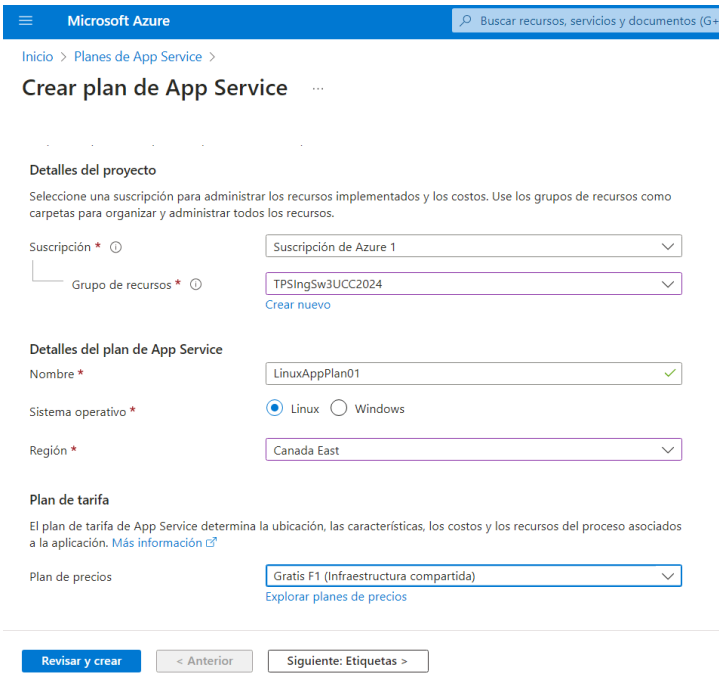
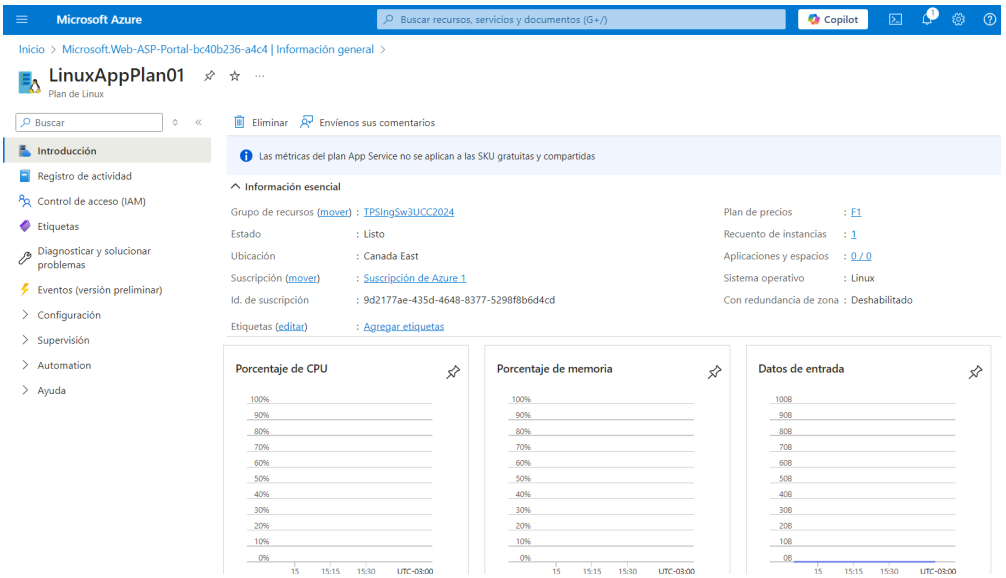
Ingeniería en Software 3
TP9
Santiago Agüero

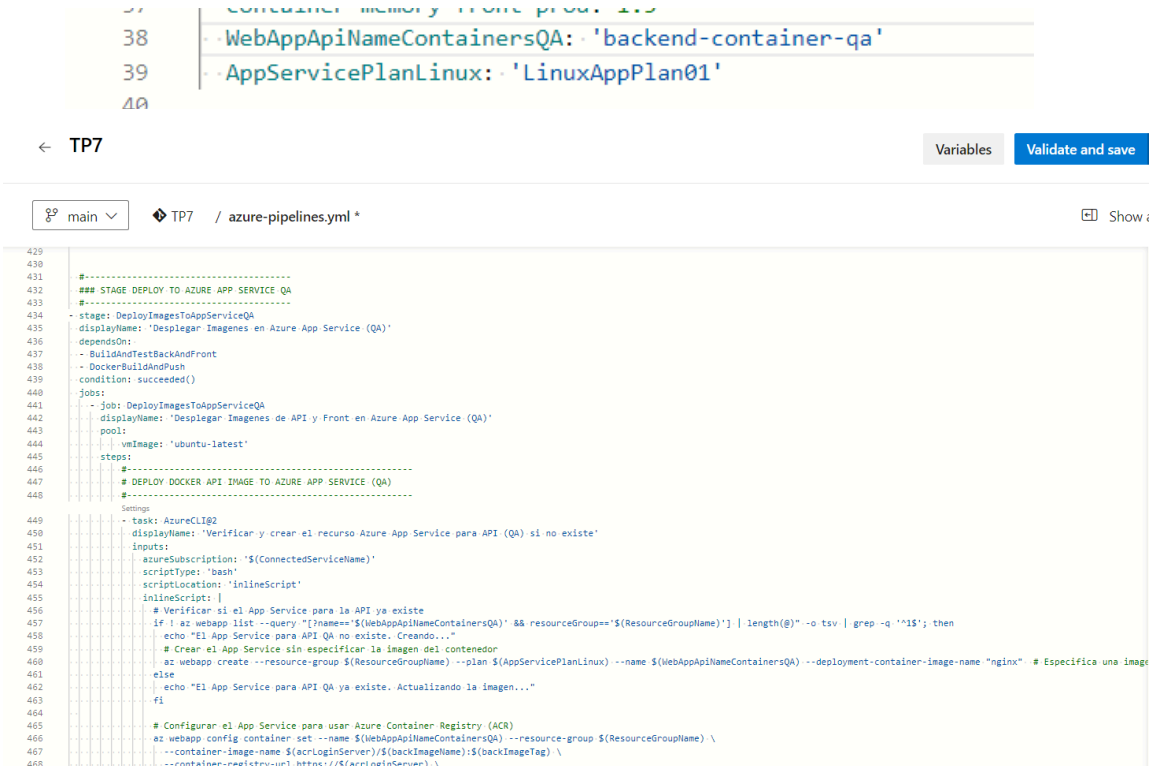
Actividad 4.1

- Agregar a nuestro pipeline una nueva etapa que dependa de nuestra etapa de Construcción y Pruebas y de la etapa de Construcción de Imagenes Docker y subida a ACR realizada en el TP08
- Agregar tareas para crear un recurso Azure Container Instances que levante un contenedor con nuestra imagen de back utilizando un AppServicePlan en Linux
- Creamos el recurso:

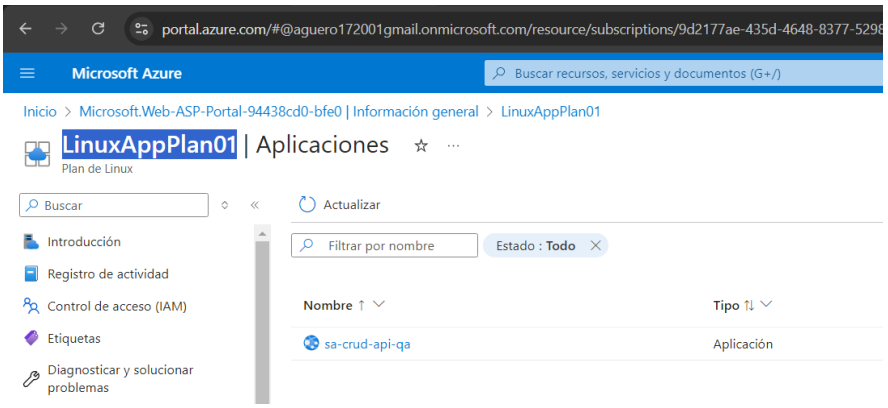




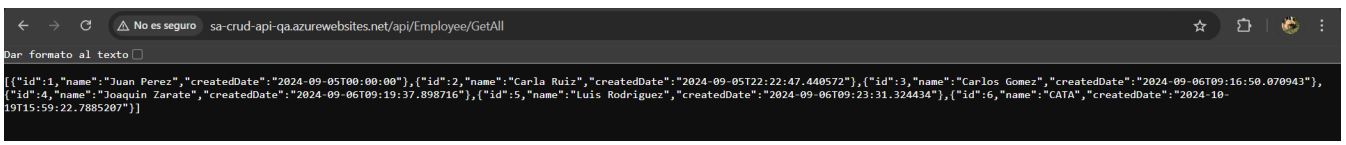
- Actualizamos variables y agregamos al pipeline las tareas para hacer el deploy a App Service para contenedores.



- Ahora corremos el pipeline:

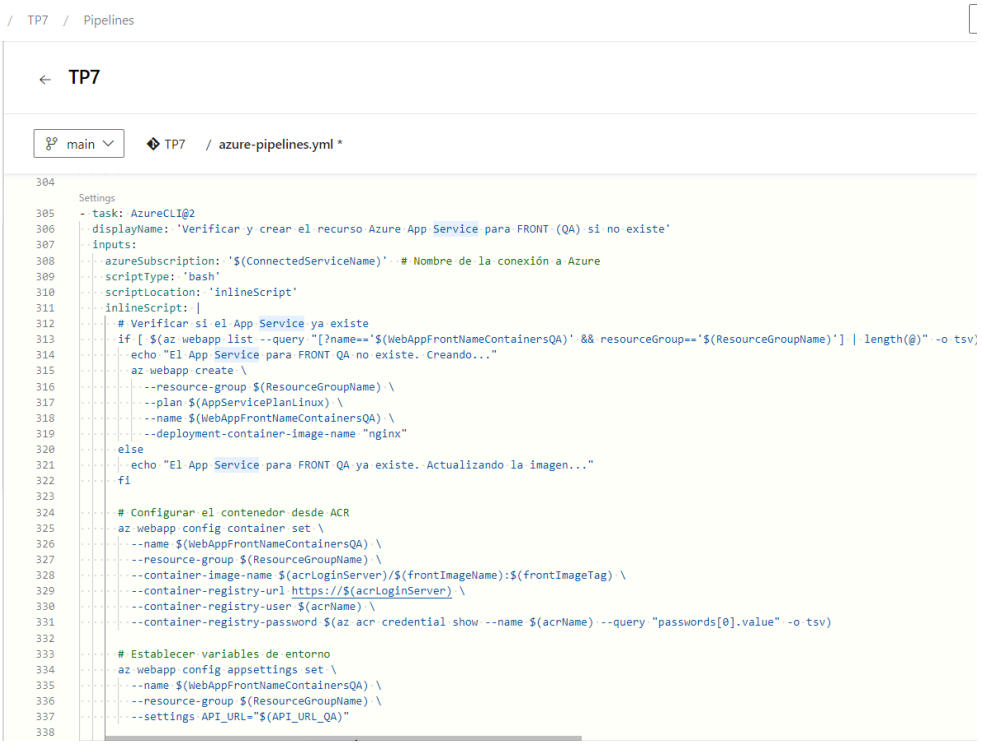


- Chequeamos que este todo funcionando

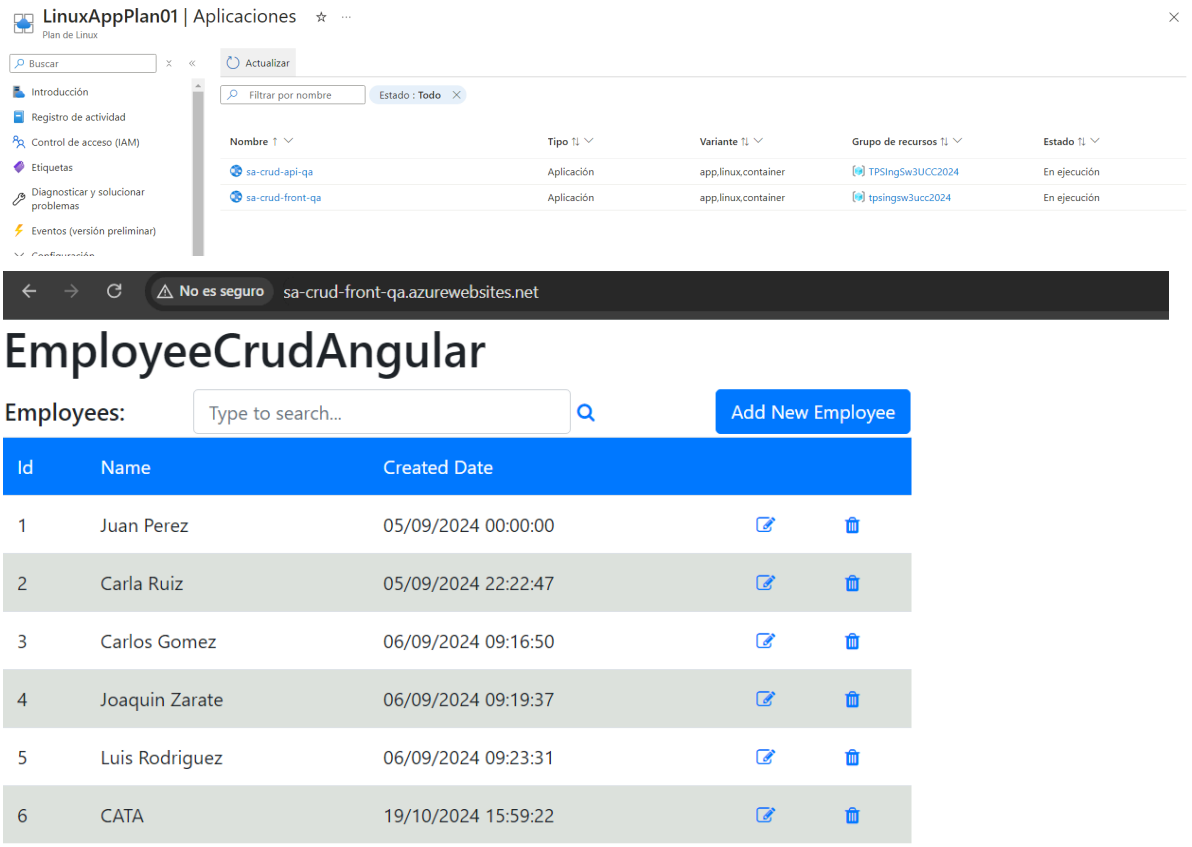


Actividad 4.2 (DESAFÍOS)

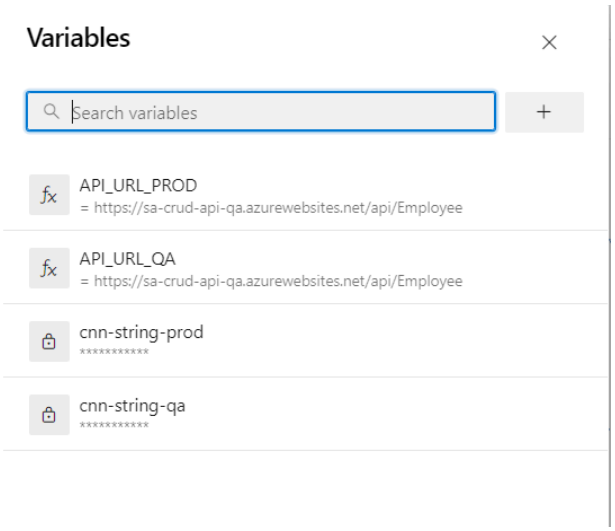
1. Agregar tareas para generar Front en Azure App Service con Soporte para Contenedores



- Y vemos que ahora se creó el container de front.



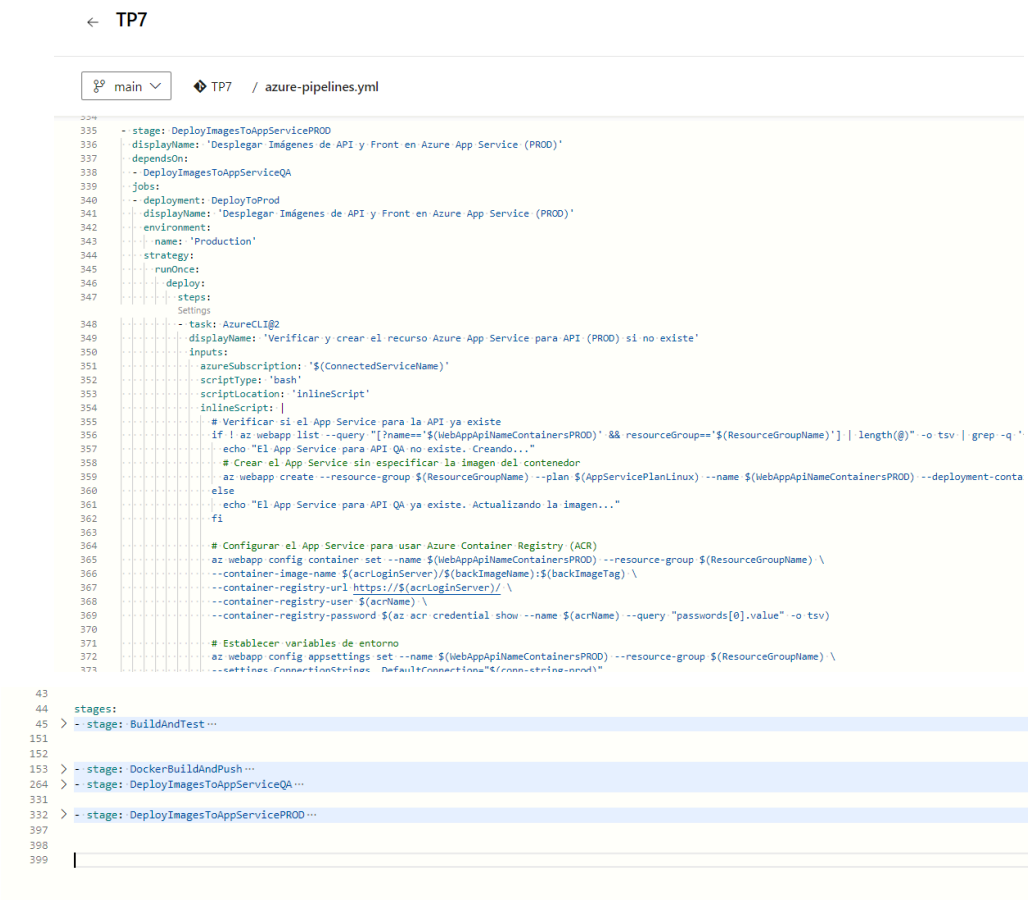
2. Agregar variables necesarias para el funcionamiento de la nueva etapa considerando que debe haber 2 entornos QA y PROD para Back y Front.



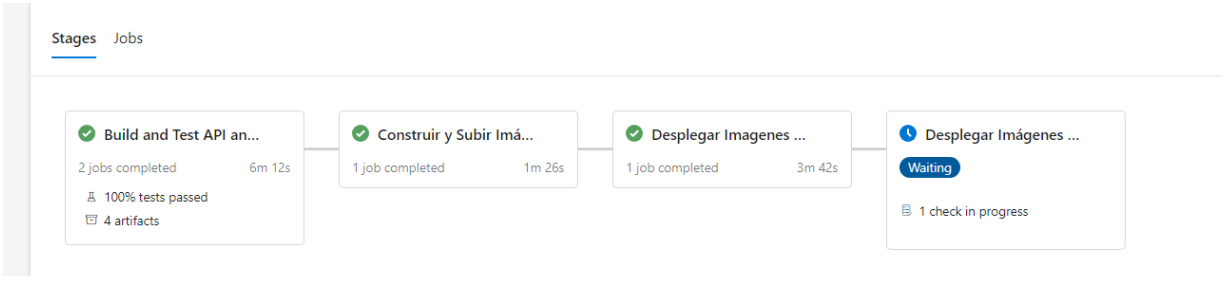

```
baseUrlBackEnd: 'http://$(backContainerInstanceNameQA).brazilsouth.azurecontainer.io'
baseUrlFrontEnd: 'http://$(frontContainerInstanceNameQA).brazilsouth.azurecontainer.io'
```

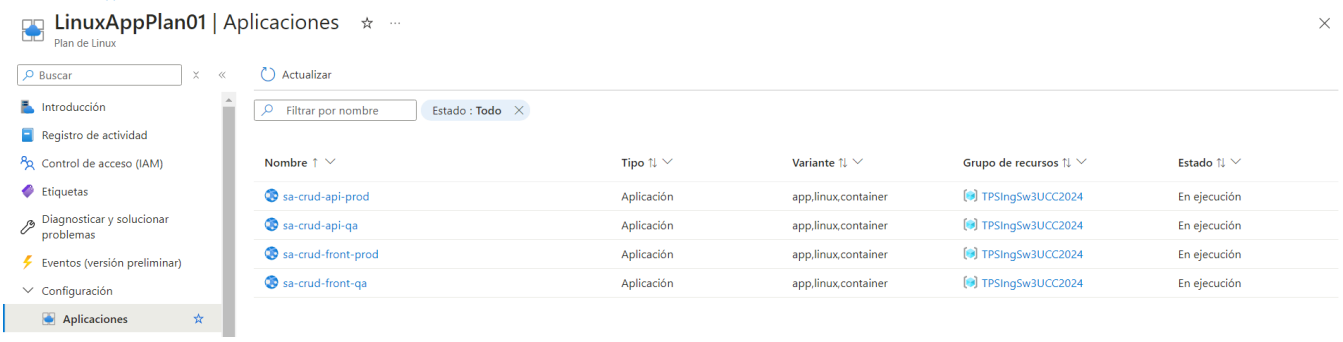
3. Agregar tareas para correr pruebas de integración en el entorno de QA de Back y Front creado en Azure App Services con Soporte para Contenedores.

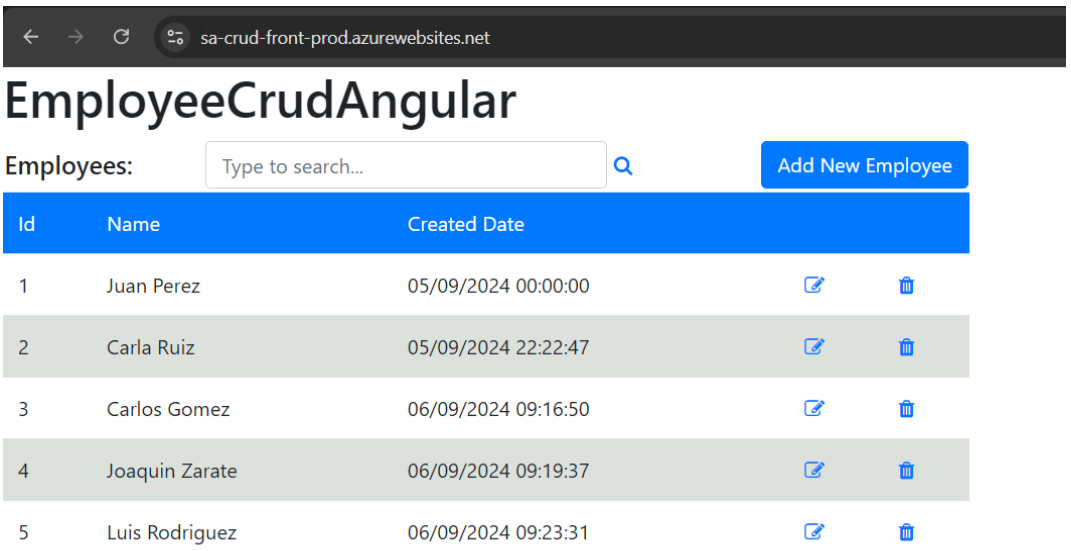
4. Agregar etapa que dependa de la etapa de Deploy en QA que genere un entorno de PROD.



- Corremos el pipeline y vemos que cuando pasa la etapa de QA nos pide aprobación.







5. Entregar un pipeline que incluya:

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

B) Construcción de Imágenes Docker y subida a ACR

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

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

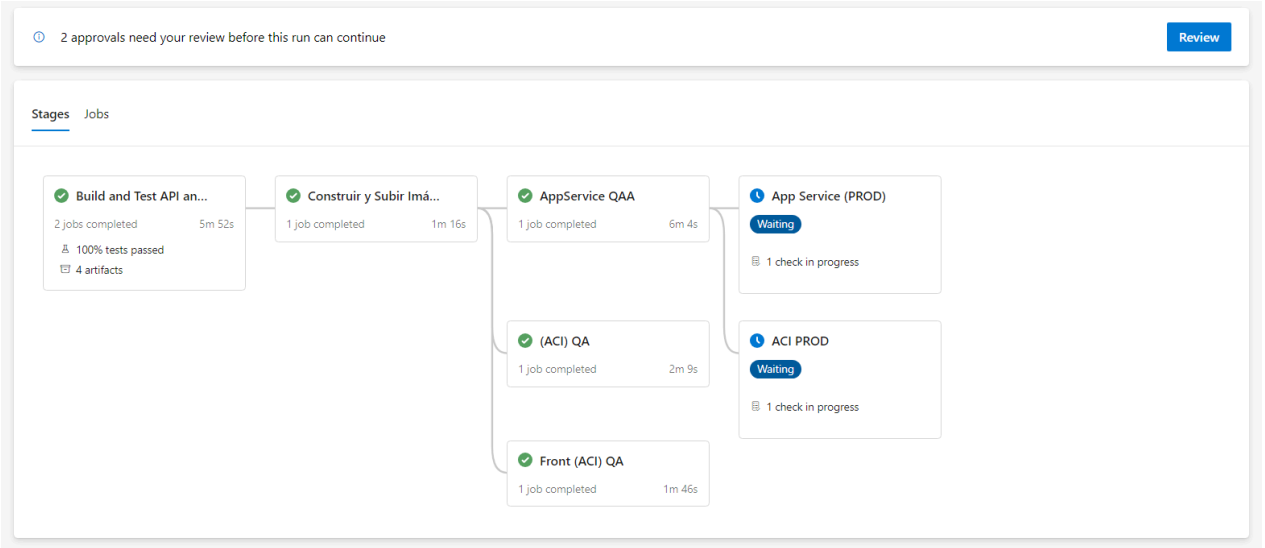
E) Deploy Back y Front en QA con pruebas de integración para Azure Web Apps con Soporte para contenedores

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

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

H) Deploy Back y Front en PROD para ACI

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



PIPELINE FINAL:

```
trigger:
- main

pool:
  vmImage: 'windows-latest'

variables:
 # AZURE VARIABLES
  ConnectedServiceName: 'ServiceConnectionARM' #Por ejemplo
'ServiceConnectionARM'
  acrLoginServer: 'saingsw3uccacr.azurecr.io' #Por ejemplo
'ascontainerregistry.azurecr.io'
  acrName: 'SAIngSw3UCCACR'
  backImageName: 'santiago-employee-crud-api' #Por ejemplo
'employee-crud-api'
  frontImageName: 'santiago-employee-crud-front'
  solution: '**/*.sln'
  buildPlatform: 'Any CPU'
  buildConfiguration: 'Release'
  Configuration: 'Release'
  frontPath: './EmployeeCrudAngular'
  ResourceGroupName: 'TPSIngSw3UCC2024' #Por ejemplo 'TPS_INGSOFT3_UCC'
  backContainerInstanceNameQA: 'santi-crud-api-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
  cnn-string-qa:
'Server=tcp:sa-sql-server-1.database.windows.net,1433;Initial
Catalog=sa-sql-bd-01;Persist Security Info=False;User
```

```
ID=sqladmin;Password=4223551Santiago;MultipleActiveResultSets=False;Encrypt=T
rue;TrustServerCertificate=False;Connection Timeout=30'
  cnn-string-prod:
'Server=tcp:sa-sql-server-1.database.windows.net,1433;Initial
Catalog=sa-sql-bd-01;Persist Security Info=False;User
ID=sqladmin;Password=4223551Santiago;MultipleActiveResultSets=False;Encrypt=T
rue;TrustServerCertificate=False;Connection Timeout=30'
  frontContainerInstanceNameQA: 'santi-crud-front-qa'
  frontImageTag: 'latest'
  container-cpu-front-qa: 1
  container-memory-front-qa: 1.5
  baseUrlBackEnd:
'http://$(backContainerInstanceNameQA).canadaeast.azurecontainer.io'
  baseUrlFrontEnd:
'http://$(frontContainerInstanceNameQA).canadaeast.azurecontainer.io'
  backContainerInstanceNamePROD: 'santi-crud-api-prod'
  frontContainerInstanceNamePROD: 'santi-crud-front-prod'
  container-cpu-api-prod: 1
  container-memory-api-prod: 1.5
  container-cpu-front-prod: 1
  container-memory-front-prod: 1.5
  AppServicePlanLinux: 'LinuxAppPlan01'
  WebAppApiNameContainersQA: 'sa-crud-api-qa'
  WebAppFrontNameContainersQA: 'sa-crud-front-qa'
  baseUrlBackEndQA: 'https://$(WebAppApiNameContainersQA).azurewebsites.net'
  baseUrlFrontEndQA:
'https://$(WebAppFrontNameContainersQA).azurewebsites.net'
  WebAppApiNameContainersPROD: 'sa-crud-api-prod'
  WebAppFrontNameContainersPROD: 'sa-crud-front-prod'


stages:
- stage: BuildAndTest
  displayName: "Build and Test API and Front"
  jobs:
  - job: BuildDotnet
    displayName: "Build and Test API"
    pool:
      vmImage: 'windows-latest'
    steps:
    - checkout: self
      fetchDepth: 0

    - task: DotNetCoreCLI@2
      displayName: 'Restaurar paquetes NuGet'
      inputs:
        command: restore
        projects: '$(solution)'
```

```
    - task: DotNetCoreCLI@2
      displayName: 'Ejecutar pruebas de la API'
      inputs:
        command: 'test'
        projects: '**/*.Tests.csproj'
        arguments: '--collect:"XPlat Code Coverage"'

    - task: PublishCodeCoverageResults@2
      displayName: 'Publicar resultados de code coverage del back-end'
      inputs:
        summaryFileLocation: '$(Agent.TempDirectory)/**/*.cobertura.xml'
        failIfCoverageEmpty: false

    - task: DotNetCoreCLI@2
      displayName: 'Compilar la API'
      inputs:
        command: build
        projects: '$(solution)'
        arguments: '--configuration $(configuration) --self-contained false'

    - task: DotNetCoreCLI@2
      displayName: 'Publicar aplicación'
      inputs:
        command: publish
        publishWebProjects: True
        arguments: '--configuration $(buildConfiguration) --output
$(Build.ArtifactStagingDirectory)'
        zipAfterPublish: true

    - task: PublishBuildArtifacts@1
      displayName: 'Publicar artefactos de compilación'
      inputs:
        PathtoPublish: '$(Build.ArtifactStagingDirectory)'
        ArtifactName: 'drop-back'
        publishLocation: 'Container'


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

  - job: BuildAngular
    displayName: "Build and Test Angular"
    pool:
      vmImage: 'ubuntu-latest'
```

```yaml
    steps:
    - task: NodeTool@0
      displayName: 'Instalar Node.js'
      inputs:
        versionSpec: '22.x'

    - script: npm install
      displayName: 'Instalar dependencias'
      workingDirectory: $(frontPath)

    - script: npx ng test --karma-config=karma.conf.js --watch=false
--browsers ChromeHeadless --code-coverage
      displayName: 'Ejecutar pruebas del front'
      workingDirectory: $(frontPath)
      continueOnError: true  # Continue on test failure

    - task: PublishCodeCoverageResults@2
      displayName: 'Publicar resultados de code coverage del front'
      inputs:
        summaryFileLocation: '$(frontPath)/coverage/lcov.info'
        failIfCoverageEmpty: false
      condition: always()
    - task: PublishTestResults@2
      displayName: 'Publicar resultados de pruebas unitarias del front'
      inputs:
        testResultsFormat: 'JUnit'
        testResultsFiles: '$(frontPath)/test-results/test-results.xml'
        failTaskOnFailedTests: true
      condition: always()

    - script: npm run build
      displayName: 'Compilar el proyecto Angular'
      workingDirectory: $(frontPath)

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

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


# #-------------------------------------------------------------
```

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

- stage: DockerBuildAndPush
  displayName: 'Construir y Subir Imágenes Docker a ACR'
  dependsOn: BuildAndTest #NOMBRE DE NUESTRA ETAPA DE BUILD Y TEST
  jobs:
    - job: docker_build_and_push
      displayName: 'Construir y Subir Imágenes Docker a ACR'
      pool:
        vmImage: 'ubuntu-latest'

      steps:
        - checkout: self

        #-------------------------------------------------------------
        # BUILD DOCKER BACK IMAGE Y PUSH A AZURE CONTAINER REGISTRY
        #-------------------------------------------------------------

        - task: DownloadPipelineArtifact@2
          displayName: 'Descargar Artefactos de Back'
          inputs:
            buildType: 'current'
            artifactName: 'drop-back'
            targetPath: '$(Pipeline.Workspace)/drop-back'

        - task: DownloadPipelineArtifact@2
          displayName: 'Descargar Dockerfile de Back'
          inputs:
            buildType: 'current'
            artifactName: 'dockerfile-back'
            targetPath: '$(Pipeline.Workspace)/dockerfile-back'

        - task: AzureCLI@2
          displayName: 'Iniciar Sesión en Azure Container Registry (ACR)'
          inputs:
            azureSubscription: '$(ConnectedServiceName)'
            scriptType: bash
            scriptLocation: inlineScript
            inlineScript: |
              az acr login --name $(acrLoginServer)

        - task: ExtractFiles@1
          displayName: 'Descomprimir API en carpeta api'
          inputs:
            archiveFilePatterns:
'$(Pipeline.Workspace)/drop-back/EmployeeCrudApi.zip'
            destinationFolder: '$(Pipeline.Workspace)/drop-back/api'
```

```yaml
- script: ls -la $(Pipeline.Workspace)/drop-back
  displayName: 'Verificar archivos publicados'


- task: Docker@2
  displayName: 'Construir Imagen Docker para Back'
  inputs:
    command: build
    repository: $(acrLoginServer)/$(backImageName)
    dockerfile: $(Pipeline.Workspace)/dockerfile-back/dockerfile
    buildContext: $(Pipeline.Workspace)/drop-back
    tags: 'latest'

- task: Docker@2
  displayName: 'Subir Imagen Docker de Back a ACR'
  inputs:
    command: push
    repository: $(acrLoginServer)/$(backImageName)
    tags: 'latest'

 #-------------------------------------------------------------
# CONSTRUIR Y SUBIR IMAGEN DEL FRONTEND
#-------------------------------------------------------------

- task: DownloadPipelineArtifact@2
  displayName: 'Descargar artefactos de Frontend'
  inputs:
    buildType: 'current'
    artifactName: 'drop-front'
    targetPath: '$(Pipeline.Workspace)/drop-front'


- task: DownloadPipelineArtifact@2
  displayName: 'Descargar Dockerfile de Frontend'
  inputs:
    buildType: 'current'
    artifactName: 'dockerfile-front'
    targetPath: '$(Pipeline.Workspace)/dockerfile-front'


- script: ls -la $(Pipeline.Workspace)/drop-front
  displayName: 'Verificar archivos del frontend'

- task: Docker@2
  displayName: 'Construir Imagen Docker para Front'
  inputs:
```

```
            command: build
            repository: $(acrLoginServer)/$(frontImageName)
            dockerfile: $(Pipeline.Workspace)/dockerfile-front/dockerfile
            buildContext:
$(Pipeline.Workspace)/drop-front/employee-crud-angular/browser
            tags: 'latest'

        - task: Docker@2
          displayName: 'Subir Imagen Docker de Front a ACR'
          inputs:
            command: push
            repository: $(acrLoginServer)/$(frontImageName)
            tags: 'latest'


#-------------------------------------------------------------
### STAGE DEPLOY TO ACI QA
#-------------------------------------------------------------

- stage: DeployToACIQA
  displayName: '(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
      #-------------------------------------------------------

      - 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) \
            --image $(acrLoginServer)/$(backImageName):$(backImageTag) \
            --registry-login-server $(acrLoginServer) \
            --registry-username $(acrName) \
            --registry-password $(az acr credential show --name $(acrName)
--query "passwords[0].value" -o tsv) \
            --dns-name-label $(backContainerInstanceNameQA) \
            --ports 80 \
            --environment-variables
ConnectionStrings__DefaultConnection="$(cnn-string-qa)" \
            --restart-policy Always \
            --cpu $(container-cpu-api-qa) \
            --memory $(container-memory-api-qa)

#-------------------------------------------------------------
### STAGE DEPLOY TO ACI QA FOR FRONT
#-------------------------------------------------------------

- stage: DeployToACIQAFront
  displayName: 'Front (ACI) QA'
  dependsOn: DockerBuildAndPush  # El contenedor del front depende de que las
imágenes estén creadas
  jobs:
    - job: deploy_to_aci_qa_front
      displayName: 'Desplegar Front ACI QA'
      pool:
        vmImage: 'ubuntu-latest'

      steps:
      #-------------------------------------------------------
      # DEPLOY DOCKER FRONT IMAGE A AZURE CONTAINER INSTANCES QA
      #-------------------------------------------------------
      - 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)"
            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)


#--------------------------------------
### STAGE DEPLOY TO AZURE APP SERVICE QA
#--------------------------------------
- stage: DeployImagesToAppServiceQA
  displayName: 'AppService QAA'
  dependsOn:
  - BuildAndTest
  - DockerBuildAndPush
  condition: succeeded()
  jobs:
    - job: DeployImagesToAppServiceQA
      displayName: 'Desplegar Imagenes de API y Front en Azure App Service
(QA)'
      pool:
        vmImage: 'ubuntu-latest'
      steps:
          #---------------------------------------------------------
          # DEPLOY DOCKER API IMAGE TO AZURE APP SERVICE (QA)
          #---------------------------------------------------------
          - 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)'] | length(@)" -o tsv | grep -q '^1$';
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-image-name "nginx"  # Especifica una imagen temporal
para permitir la creación
                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) \
                  --container-registry-user $(acrName) \
                  --container-registry-password $(az acr credential show
--name $(acrName) --query "passwords[0].value" -o tsv)
                # Establecer variables de entorno
                az webapp config appsettings set --name
$(WebAppApiNameContainersQA) --resource-group $(ResourceGroupName) \
                  --settings
ConnectionStrings__DefaultConnection="$(cnn-string-qa)" \


          - task: AzureCLI@2
            displayName: 'Verificar y crear el recurso Azure App Service para
fron (QA) si no existe'
            inputs:
              azureSubscription: '$(ConnectedServiceName)'
              scriptType: 'bash'
              scriptLocation: 'inlineScript'
              inlineScript: |
                # Verificar si el App Service para la API ya existe
                if ! az webapp list --query
"[?name=='$(WebAppFrontNameContainersQA)' &&
resourceGroup=='$(ResourceGroupName)'] | length(@) > `0`"; then
                  echo "El App Service para API QA no existe. Creando..."
                  # Crear el App Service sin especificar la imagen del
contenedor
```

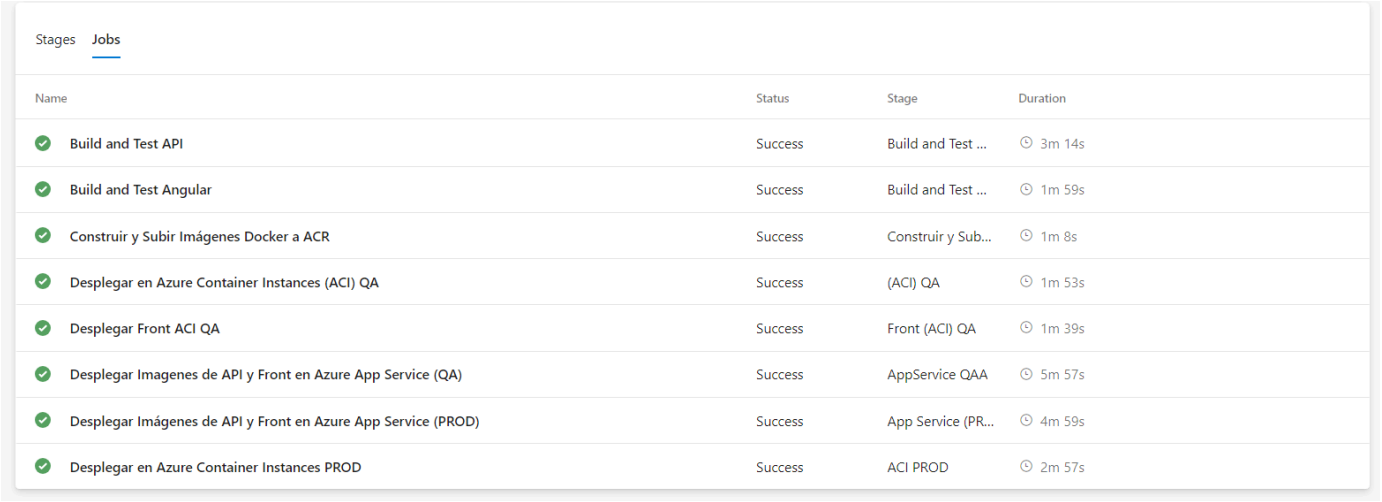
```
                  az webapp create --resource-group $(ResourceGroupName)
--plan $(AppServicePlanLinux) --name $(WebAppFrontNameContainersQA)
--deployment-container-image-name "nginx"
                else
                  echo "El App Service para API QA ya existe. Actualizando la
imagen..."
                fi

                # Configurar el App Service para usar Azure Container
Registry (ACR)
                az webapp config container set --name
$(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
                az webapp config appsettings set --name
$(WebAppFrontNameContainersQA) --resource-group $(ResourceGroupName) \
                --settings API_URL="$(API_URL_QA)"


#-------------------------------------------------------------
#DEPLOY A PROD
#-------------------------------------------------------------

- stage: DeployImagesToAppServicePROD
  displayName: 'App Service (PROD)'
  dependsOn: DeployImagesToAppServiceQA
  jobs:
    - deployment: DeployToProd
      displayName: 'Desplegar Imágenes de API y Front en Azure App Service
(PROD)'
      environment: 'Production'
      strategy:
        runOnce:
          deploy:
            steps:

#-------------------------------------------------------------
# DEPLOY DOCKER BACK IMAGE A AZURE APP SERVICE PROD
#-------------------------------------------------------------

            - task: AzureCLI@2
```

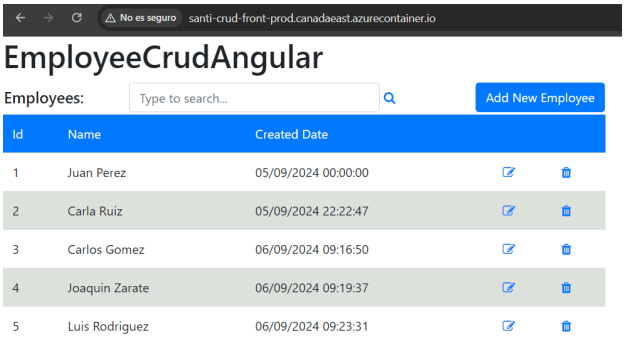
```
              displayName: 'Verificar y crear el recurso Azure App Service
para API (PROD) si no existe'
              inputs:
                azureSubscription: '$(ConnectedServiceName)'
                scriptType: 'bash'
                scriptLocation: 'inlineScript'
                inlineScript: |
                  # Verificar si el App Service para la API ya existe
                  if ! az webapp list --query
"[?name=='$(WebAppApiNameContainersPROD)' &&
resourceGroup=='$(ResourceGroupName)'] | length(@)" -o tsv | grep -q '^1$';
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 $(WebAppApiNameContainersPROD)
--deployment-container-image-name "nginx"
                  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
$(WebAppApiNameContainersPROD) --resource-group $(ResourceGroupName) \
                  --container-image-name
$(acrLoginServer)/$(backImageName):$(backImageTag) \
                  --container-registry-url https://$(acrLoginServer)/ \
                  --container-registry-user $(acrName) \
                  --container-registry-password $(az acr credential show
--name $(acrName) --query "passwords[0].value" -o tsv)

                  # Establecer variables de entorno
                  az webapp config appsettings set --name
$(WebAppApiNameContainersPROD) --resource-group $(ResourceGroupName) \
                  --settings
ConnectionStrings__DefaultConnection="$(conn-string-prod)"

            - task: AzureCLI@2
              displayName: 'Verificar y crear el recurso Azure App Service
para FRONT (PROD) si no existe'
              inputs:
                azureSubscription: '$(ConnectedServiceName)'
                scriptType: 'bash'
                scriptLocation: 'inlineScript'
                inlineScript: |
```

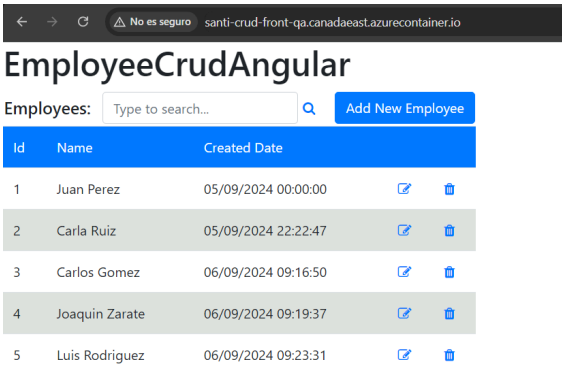
```
                  # Verificar si el App Service para la API ya existe
                  if ! az webapp list --query
"[?name=='$(WebAppFrontNameContainersPROD)' &&
resourceGroup=='$(ResourceGroupName)'] | length(@)" -o tsv | grep -q '^1$';
then
                    echo "El App Service para API QA no existe. Creando..."
                    # Crear el App Service sin especificar la imagen del
contenedor
                    az webapp create --resource-group $(ResourceGroupName)
--plan $(AppServicePlanLinux) --name $(WebAppFrontNameContainersPROD)
--deployment-container-image-name "nginx"
                  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
$(WebAppFrontNameContainersPROD) --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
                  az webapp config appsettings set --name
$(WebAppFrontNameContainersPROD) --resource-group $(ResourceGroupName) \
                  --settings API_URL="$(API_URL_PROD)"

# #-------------------------------------------------------------
# ### DEPLOY A PROD ACI
# #-------------------------------------------------------------

- stage: DeployToACIPROD
  displayName: 'ACI PROD'
  dependsOn: DeployImagesToAppServiceQA
  jobs:
    - deployment: DeployToProd
      displayName: 'Desplegar en Azure Container Instances PROD'
      environment: 'Production'
      strategy:
        runOnce:
          deploy:
            steps:
            #-------------------------------------------------------------
```

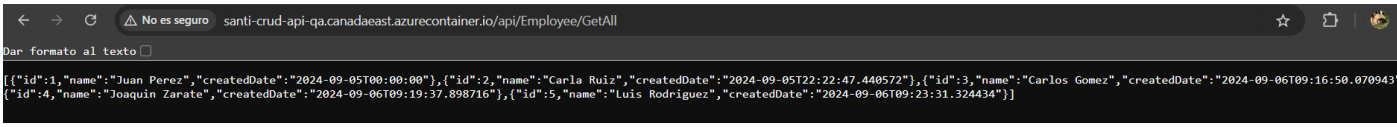
```
            # DEPLOY DOCKER BACK IMAGE A AZURE CONTAINER INSTANCES PROD
            #------------------------------------------------------------
              - task: AzureCLI@2
                displayName: 'Desplegar Imagen Docker de Back en ACI PROD'
                inputs:
                  azureSubscription: '$(ConnectedServiceName)'
                  scriptType: bash
                  scriptLocation: inlineScript
                  inlineScript: |
                    echo "Resource Group: $(ResourceGroupName)"
                    echo "Container Instance Name:
$(backContainerInstanceNamePROD)"
                    echo "ACR Login Server: $(acrLoginServer)"
                    echo "Image Name: $(backImageName)"
                    echo "Image Tag: $(backImageTag)"
                    echo "API URL: $(cnn-string-prod)"

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

                    az container create --resource-group $(ResourceGroupName)
\
                    --name $(backContainerInstanceNamePROD) \
                    --image
$(acrLoginServer)/$(backImageName):$(backImageTag) \
                    --registry-login-server $(acrLoginServer) \
                    --registry-username $(acrName) \
                    --registry-password $(az acr credential show --name
$(acrName) --query "passwords[0].value" -o tsv) \
                    --dns-name-label $(backContainerInstanceNamePROD) \
                    --ports 80 \
                    --environment-variables
ConnectionStrings__DefaultConnection="$(cnn-string-prod)" \
                    --restart-policy Always \
                    --cpu $(container-cpu-api-qa) \
                    --memory $(container-memory-api-qa)

              - task: AzureCLI@2
                displayName: 'Desplegar Imagen Docker de Front en ACI PROD'
                inputs:
                  azureSubscription: '$(ConnectedServiceName)'
                  scriptType: bash
                  scriptLocation: inlineScript
                  inlineScript: |
                    echo "Resource Group: $(ResourceGroupName)"
                    echo "Container Instance Name:
$(frontContainerInstanceNamePROD)"
                    echo "ACR Login Server: $(acrLoginServer)"
```

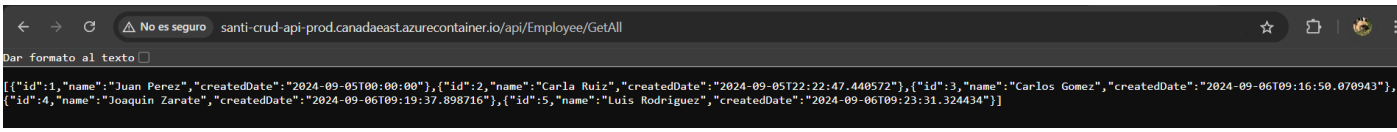
```
                    echo "Image Name: $(frontImageName)"
                    echo "Image Tag: $(frontImageTag)"
                    echo "API URL: $(API_URL_PROD)"

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

                    az container create --resource-group $(ResourceGroupName)
\
                    --name $(frontContainerInstanceNamePROD) \
                    --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 $(frontContainerInstanceNamePROD) \
                    --ports 80 \
                    --environment-variables API_URL=$(API_URL_PROD) \
                    --restart-policy Always \
                    --cpu $(container-cpu-front-prod) \
                    --memory $(container-memory-front-prod)
```

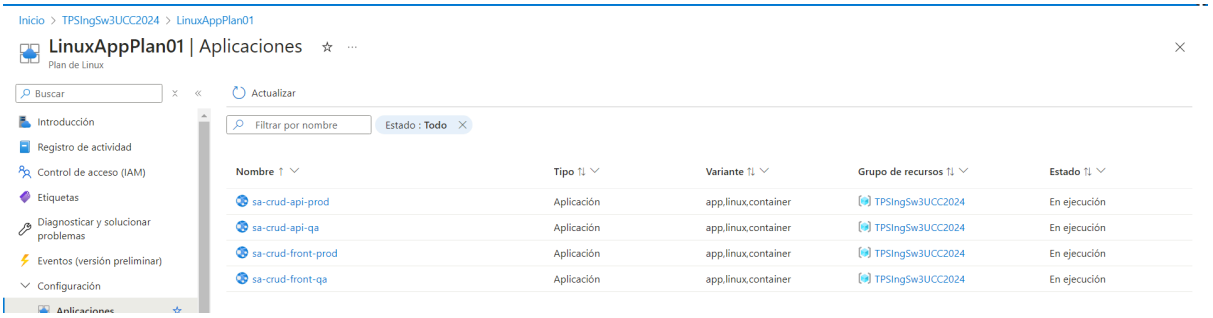


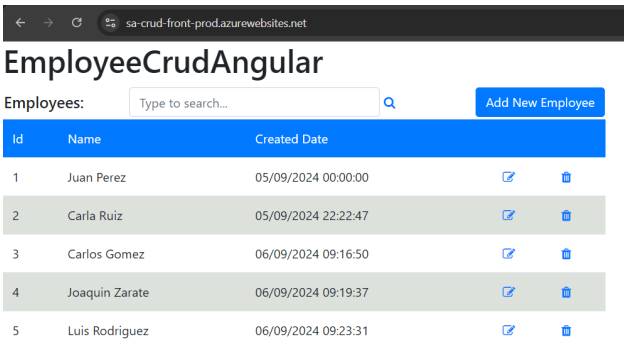


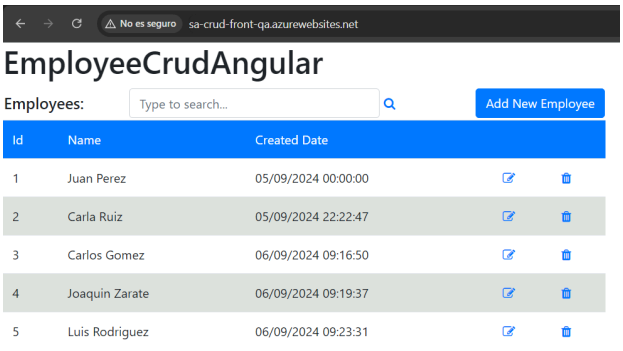


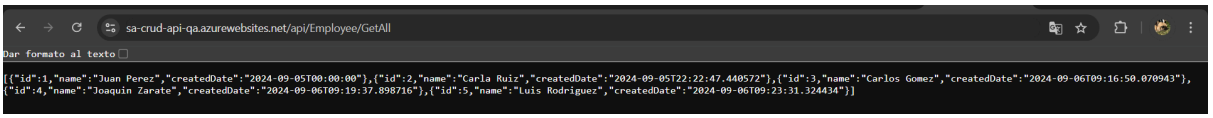

No es seguro
santi-crud-api-qa.canadaeast.azurecontainer.io/api/Employee/GetAll
Dar formato al texto
[{"id":1,"name":"Juan Perez","createdDate":"2024-09-05T00:00:00"},{"id":2,"name":"Carla Ruiz","createdDate":"2024-09-05T22:22:47.440572"},{"id":3,"name":"Carlos Gomez","createdDate":"2024-09-06T09:16:50.070943"},
{"id":4,"name":"Joaquin Zarate","createdDate":"2024-09-06T09:19:37.898716"},{"id":5,"name":"Luis Rodriguez","createdDate":"2024-09-06T09:23:31.324434"}]

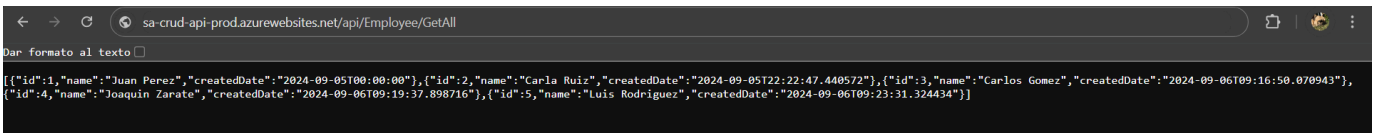

No es seguro
santi-crud-api-prod.canadaeast.azurecontainer.io/api/Employee/GetAll
Dar formato al texto
[{"id":1,"name":"Juan Perez","createdDate":"2024-09-05T00:00:00"},{"id":2,"name":"Carla Ruiz","createdDate":"2024-09-05T22:22:47.440572"},{"id":3,"name":"Carlos Gomez","createdDate":"2024-09-06T09:16:50.070943"},
{"id":4,"name":"Joaquin Zarate","createdDate":"2024-09-06T09:19:37.898716"},{"id":5,"name":"Luis Rodriguez","createdDate":"2024-09-06T09:23:31.324434"}]


Inicio > TPSIngSw3UCC2024 > LinuxAppPlan01
LinuxAppPlan01 | Aplicaciones
Plan de Linux
Buscar
Introducción
Registro de actividad
Control de acceso (IAM)
Etiquetas
Diagnosticar y solucionar problemas
Eventos (versión preliminar)
Configuración
Aplicaciones
Actualizar
Filtrar por nombre
Estado : Todo
Nombre
Tipo
Variante
Grupo de recursos
Estado
sa-crud-api-prod
Aplicación
app,linux,container
TPSIngSw3UCC2024
En ejecución
sa-crud-api-qa
Aplicación
app,linux,container
TPSIngSw3UCC2024
En ejecución
sa-crud-front-prod
Aplicación
app,linux,container
TPSIngSw3UCC2024
En ejecución
sa-crud-front-qa
Aplicación
app,linux,container
TPSIngSw3UCC2024
En ejecución


sa-crud-front-prod.azurewebsites.net
EmployeeCrudAngular
Employees:
Type to search...
Add New Employee
Id
Name
Created Date
1
Juan Perez
05/09/2024 00:00:00
2
Carla Ruiz
05/09/2024 22:22:47
3
Carlos Gomez
06/09/2024 09:16:50
4
Joaquin Zarate
06/09/2024 09:19:37
5
Luis Rodriguez
06/09/2024 09:23:31


No es seguro
sa-crud-front-qa.azurewebsites.net
EmployeeCrudAngular
Employees:
Type to search...
Add New Employee
Id
Name
Created Date
1
Juan Perez
05/09/2024 00:00:00
2
Carla Ruiz
05/09/2024 22:22:47
3
Carlos Gomez
06/09/2024 09:16:50
4
Joaquin Zarate
06/09/2024 09:19:37
5
Luis Rodriguez
06/09/2024 09:23:31


sa-crud-api-qa.azurewebsites.net/api/Employee/GetAll
Dar formato al texto
[{"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"}]


sa-crud-api-prod.azurewebsites.net/api/Employee/GetAll
Dar formato al texto
[{"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"}]