

# Trabajo Práctico 9

## Actividad

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

Crear plan de App Service ...

Datos básicosEtiquetasRevisar y crear

Resumen

Plan de App Service de Microsoft

Detalles

Suscripción

Azure subscription 1

Grupo de recursos

TP5IngSoft3UCC2024

Nombre

AppServiceMRM

Sistema operativo

Linux

Región

Brazil South

SKU

Premium V3

Tamaño

Pequeño

ACU

Mínimo de ACU/vCPU 195

Memoria

8 GB de memoria

Inicio > Microsoft.Web-ASP-Portal-ccc9306-61df | Información general

AppServiceMRMPlan de Linux

Eliminar Envíenos sus comentarios

Introducción

Registro de actividad

Control de acceso (IAM)

Etiquetas

Diagnosticar y solucionar problemas

Eventos (versión preliminar)

Configuración

Supervisión

Automation

Ayuda

Información esencial

Grupo de recursos (move)TP5IngSoft3UCC2024

Estado: Listo

Ubicación: Brazil South

Suscripción (move)Azure subscription 1

Id. de suscripción: 5a496d64-56ee-4ba3-9f0d-756852097ab

Etiquetas (add)Agregar etiquetas

Plan de precios: P1v3

Recuento de instancias: 1

Aplicaciones y espacios: 0/0

Sistema operativo: Linux

Con redundancia de zona: Deshabilitado

Porcentaje de CPU

Porcentaje de memoria

Datos de entrada

Datos salientes

- Edito pipeline

```
• frontImageTag: 'latest'
• backContainerInstanceNameProd: 'mrm-container-back-prod'
• frontContainerInstanceNameProd: 'mrm-container-front-prod'
• WebAppApiNameContainersQA: 'mrm-crud-api-qa'
• AppServicePlanLinux: 'AppServiceMRM'
```

```

372 #-----
373 ### STAGE DEPLOY TO AZURE APP SERVICE QA
374 #-----
375 - stage: DeployImagesToAppServiceQA
376 - displayName: 'Desplegar Imágenes en Azure App Service (QA)'
377 - dependsOn:
378 - - BuildAndTestBackAndFront
379 - - DockerBuildAndPush
380 - condition: succeeded()
381 - jobs:
382 - - job: DeployImagesToAppServiceQA
383 - - displayName: 'Desplegar Imágenes de API y Front en Azure App Service (QA)'
384 - - pool:
385 - - vmImage: 'ubuntu-latest'
386 - - steps:
387 - - #-----
388 - - # DEPLOY DOCKER API IMAGE TO AZURE APP SERVICE (QA)
389 - - #-----

```

```

387 #-----
388 # DEPLOY DOCKER API IMAGE TO AZURE APP SERVICE (QA)
389 #-----
390 #-----
391 # task: AzureCLI@2
392 # displayName: 'Verificar y crear el recurso Azure App Service para API (QA) si no existe'
393 # inputs:
394 #   azureSubscription: '$(ConnectedServiceName)'
395 #   scriptType: 'bash'
396 #   scriptLocation: 'inlineScript'
397 #   inlineScript: |
398 #     # Verificar si el App Service para la API ya existe
399 #     if ! az webapp list --query "[?name=='$(WebAppApiNameContainersQA)' && resourceGroup=='$(ResourceGroupName)']" --output json | grep -q "is"; then
400 #       # Crear el App Service sin especificar la imagen del contenedor
401 #       az webapp create --resource-group $(ResourceGroupName) --plan $(AppServicePlanLinux) --name $(WebAppApiNameContainersQA) --deployment-container-image-name "nginx" # Especifica una imagen temporal para permitir la cre
402 #     else
403 #       echo "El App Service para API QA ya existe. Actualizando la imagen..."
404 #     fi
405 #
406 # Configurar el App Service para usar Azure Container Registry (ACR)
407 # az webapp config container set --name $(WebAppApiNameContainersQA) --resource-group $(ResourceGroupName) \
408 #   --container-image-name $(acrLoginServer)/$(backImageName):$(backImageTag) \
409 #   --container-registry-url https://$(acrLoginServer) \
410 #   --container-registry-user $(acrName) \
411 #   --container-registry-password $(az acr credential show --name $(acrName) --query "passwords[0].value" -o tsv)
412 # Establecer variables de entorno
413 # az webapp config appsettings set --name $(WebAppApiNameContainersQA) --resource-group $(ResourceGroupName) \
414 #   --settings ConnectionStrings_DefaultConnection="$(conn-string-qa)"
415 #-----

```

**Azure DevOps** manuromeromedina12345 / TP07-Angular / Pipelines / TP07-Angular / 20241021.2

Search

**TP07-Angular**

- Overview
- Boards
- Repos
- Pipelines
- Environments
- Releases
- Library
- Task groups
- Deployment groups
- Test Plans
- Artifacts

**Summary** Tests Environments Associated pipelines Code Coverage

**Warnings** 2

- No data was written into the file /home/vsts/work/\_temp/task\_outputs/build\_1729534705067.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\_1729534761086.txt  
Construir y Subir Imágenes Docker a ACR • Construir y Subir Imágenes Docker a ACR • Construir Imagen Docker para Front

**Stages** Jobs

**Build and Test API an...**  
2 jobs completed 11m 8s  
100% tests passed  
4 artifacts

**Construir y Subir Imá...**  
1 job completed 2m 37s

**Desplegar Imágenes ...**  
1 job completed 2m 10s

**Desplegar PROD**  
1 job completed 4m 26s

**Desplegar en Azure C...**  
1 job completed 1m 56s

**Desplegar en Azure Conta...**

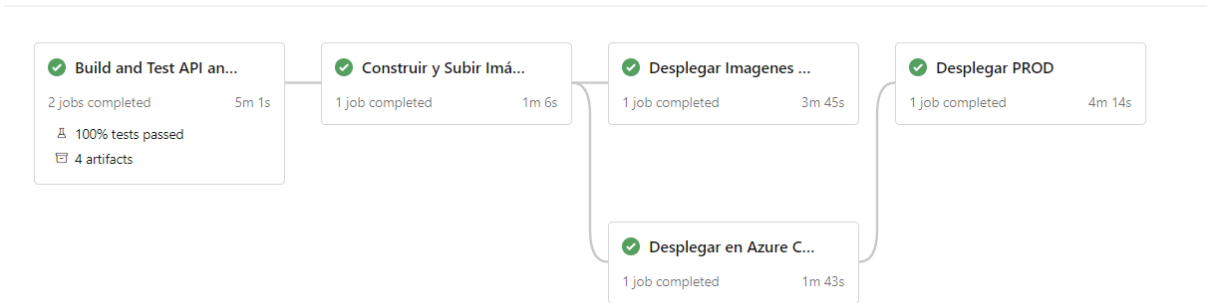
Run just this stage

Desafíos:

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

```
416 | .....  
417 | .....#-----  
418 | .....# DEPLOY DOCKER FRONT IMAGE TO AZURE APP SERVICE (QA)  
419 | .....#-----  
420 | .....Settings  
421 | .....-- task: AzureCLI@2  
422 | .....  displayName: 'Verificar y crear el recurso Azure App Service para Front (QA)'  
423 | .....  inputs:  
424 | .....    azureSubscription: '$(ConnectedServiceName)'  
425 | .....    scriptType: 'bash'  
426 | .....    scriptLocation: 'inlinescript'  
427 | .....    inlinescript: |  
428 | .....      # Verificar si el App Service para el Front QA ya existe  
429 | .....      if ! az webapp list --query "[?name=='$(WebAppFrontNameContainersQA)' && resourceGroup=='$(ResourceGroupName)']"; then  
430 | .....        echo "El App Service para Front QA no existe. Creando..."  
431 | .....        # Crear el App Service para el Front sin especificar la imagen del contenedor  
432 | .....        az webapp create --resource-group $(ResourceGroupName) --plan $(AppServicePlanLinux) --name $(WebAppFrontNameContainersQA)  
433 | .....      else  
434 | .....        echo "El App Service para Front QA ya existe. Actualizando la imagen..."  
435 | .....      fi  
436 | .....  
437 | .....      # Configurar el App Service para usar Azure Container Registry (ACR)  
438 | .....      az webapp config container set --name $(WebAppFrontNameContainersQA) --resource-group $(ResourceGroupName) \\  
439 | .....        --container-image-name $(acrLoginServer)/$(frontImageName):$(frontImageTag) \\  
440 | .....        --container-registry-url https://$(acrLoginServer) \\  
441 | .....        --docker-registry-server-url https://$(acrLoginServer) \\  
442 | .....        --docker-registry-server-user $(acrName) \\  
443 | .....        --docker-registry-server-password $(az acr credential show --name $(acrName) --query "passwords[0].value" --output tsv)  
444 | .....  
445 | .....      # Establecer variables de entorno  
446 | .....      az webapp config appsettings set --name $(WebAppFrontNameContainersQA) --resource-group $(ResourceGroupName) \\  
447 | .....        --settings API_URL="$(PROD)"
```

Stages Jobs



Inicio > AppServiceMRM

AppServiceMRM | Aplicaciones ☆

Plan de Linux

Buscar Actualizar

Filtrar por nombre Estado: Todo

Nombre	Tipo	Variante	Grupo de recursos	Estado
mrm-crud-api-qa	Aplicación	app.linux.container	TPSingSoft3UCC2024	En ejecución
mrm-crud-front-qa	Aplicación	app.linux.container	tpSingsoft3ucc2024	En ejecución

Introducción Registro de actividad Control de acceso (IAM) Etiquetas Diagnosticar y solucionar problemas Eventos (versión preliminar) Configuración

- **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 ×

---

fx
API\_PROD  
= mrm-crud-api-qa.azurewebsites.net/api/employee/getall

---

fx
API\_URL  
= mrm-crud-api-qa.eastus.azurecontainer.io/api/employee...

---

fx
cnn\_string\_qa  
= Server=tcp:mrmsql.database.windows.net,1433;Initial Cat...

---

fx
PROD  
= mrm\_container\_back\_prod-eastus.azurecontainer.io

#### 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.

```
##### Integration Test with Cypress
- job: IntegrationTesting
  displayName: 'Integrations Tests'
  dependsOn: 'DeployImagesToAppServiceQA'
  variables:
    baseUrl: '$(frontContainerInstanceNameQA).azurewebsites.net' # Construir la URL

  steps:

  - script: |
    cd $(Build.SourcesDirectory)/EmployeeCrudAngular
    npm install typescript ts-node
    displayName: 'Install TypeScript'

  - # Ejecutar pruebas de Cypress
    script: |
    cd $(Build.SourcesDirectory)/EmployeeCrudAngular
    npx cypress run --config-file cypress.config.ts --env baseUrl=$(baseUrl)
    displayName: 'Run Cypress E2E Tests'

  - # Publicar los resultados de las pruebas
    Settings
    task: PublishTestResults@2
    inputs:
      testResultsFiles: '$(Build.SourcesDirectory)/EmployeeCrudAngular/cypress/results/*.xml'
      testRunTitle: 'Cypress E2E Tests (QA)'
    displayName: 'Publicar resultados de Cypress'
```

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

```

451 ### DEPLOY A PROD
452 - stage: DeployImagesToAppServicePROD
453 - displayName: 'Desplegar Imágenes de API y Front en Azure App Service (PROD)'
454 - dependsOn: DeployImagesToAppServiceQA
455 - jobs:
456 - - deployment: DeployToProd
457 - - displayName: 'Desplegar Imágenes de API y Front en Azure App Service (PROD)'
458 - - environment: 'Production'
459 - - strategy:
460 - - runOnce:
461 - - deploy:
462 - - steps:
463 - - # -----
464 - - # DEPLOY DOCKER BACK IMAGE A AZURE APP SERVICE PROD
465 - - # -----
466 - - Settings
467 - - - task: AzureCLI@2
468 - - - displayName: 'Verificar y crear el recurso Azure App Service para API (PROD) si no existe'
469 - - - inputs:
470 - - - azureSubscription: '$(ConnectedServiceName)'
471 - - - scriptType: 'bash'
472 - - - scriptLocation: 'inlineScript'
473 - - - inlineScript: |
474 - - - # Verificar si el App Service para la API ya existe
475 - - - if ! az webapp list --query "[?name=='$(WebAppApiNameContainersPROD)' && resourceGroup=='$(ResourceGroupName)']" | length(0) -o tsv | grep -q '^1$'; then
476 - - - echo "El App Service para API PROD no existe. Creando..."
477 - - - # Crear el App Service sin especificar la imagen del contenedor
478 - - - az webapp create --resource-group $(ResourceGroupName) --plan $(AppServicePlanLinux) --name $(WebAppApiNameContainersPROD)
479 - - - else
480 - - - echo "El App Service para API PROD ya existe. Actualizando la imagen..."
481 - - - fi
482 - - -
483 - - # Configurar el App Service para usar Azure Container Registry (ACR)
484 - - - Settings
485 - - - - task: AzureCLI@2
486 - - - - displayName: 'Configurar App Service para usar Azure Container Registry (ACR) -- API'
487 - - - - inputs:
488 - - - - azureSubscription: '$(ConnectedServiceName)'
489 - - - - scriptType: 'bash'
490 - - - - scriptLocation: 'inlineScript'
491 - - - - inlineScript: |
492 - - - - az webapp config container set --name $(WebAppApiNameContainersPROD) --resource-group $(ResourceGroupName) \
493 - - - - --container-image-name $(acrLoginServer)/$(backImageName):$(backImageTag) \
494 - - - - --container-registry-url https://$(acrLoginServer) \
495 - - - - --container-registry-user $(acrName) \
496 - - - - --container-registry-password $(az acr credential show --name $(acrName) --query "passwords[0].value" -o tsv)
497 - - - -
498 - - - # Establecer variables de entorno
499 - - - - Settings
500 - - - - - task: AzureCLI@2
501 - - - - - displayName: 'Establecer variables de entorno -- API'
502 - - - - - inputs:
503 - - - - - azureSubscription: '$(ConnectedServiceName)'
504 - - - - - scriptType: 'bash'
505 - - - - - scriptLocation: 'inlineScript'
506 - - - - - inlineScript: |
507 - - - - - az webapp config appsettings set --name $(WebAppApiNameContainersPROD) --resource-group $(ResourceGroupName) \
508 - - - - - --settings ConnectionStrings_DefaultConnection="$(cnn-string-prod)"
509 - - - -
510 - - - # -----
511 - - - # DEPLOY DOCKER FRONT IMAGE A AZURE APP SERVICE PROD
512 - - - # -----
513 - - - Settings
514 - - - - task: AzureCLI@2
515 - - - - displayName: 'Verificar y crear el recurso Azure App Service para FRONT (PROD) si no existe'
516 - - - - inputs:
517 - - - - azureSubscription: '$(ConnectedServiceName)'
518 - - - - scriptType: 'bash'
519 - - - - scriptLocation: 'inlineScript'
520 - - - - inlineScript: |
521 - - - - # Verificar si el App Service para el FRONT ya existe
522 - - - - if ! az webapp list --query "[?name=='$(WebAppFrontNameContainersPROD)' && resourceGroup=='$(ResourceGroupName)']" | length(0) -o tsv | grep -q '^1$'; then
523 - - - - echo "El App Service para FRONT PROD no existe. Creando..."
524 - - - - # Crear el App Service sin especificar la imagen del contenedor
525 - - - - az webapp create --resource-group $(ResourceGroupName) --plan $(AppServicePlanLinux) --name $(WebAppFrontNameContainersPROD)
526 - - - - else
527 - - - - echo "El App Service para FRONT PROD ya existe. Actualizando la imagen..."
528 - - - - fi
529 - - - -
530 - - - # Configurar el App Service para usar Azure Container Registry (ACR)
531 - - - - Settings
532 - - - - - task: AzureCLI@2
533 - - - - - displayName: 'Configurar App Service para usar Azure Container Registry (ACR) -- FRONT'
534 - - - - - inputs:
535 - - - - - azureSubscription: '$(ConnectedServiceName)'
536 - - - - - scriptType: 'bash'
537 - - - - - scriptLocation: 'inlineScript'
538 - - - - - inlineScript: |
539 - - - - - az webapp config container set --name $(WebAppFrontNameContainersPROD) --resource-group $(ResourceGroupName) \
540 - - - - - --container-image-name $(acrLoginServer)/$(frontImageName):$(frontImageTag) \
541 - - - - - --container-registry-url https://$(acrLoginServer) \
542 - - - - - --container-registry-user $(acrName) \
543 - - - - - --container-registry-password $(az acr credential show --name $(acrName) --query "passwords[0].value" -o tsv)
544 - - - - -
545 - - - # Establecer variables de entorno
546 - - - - Settings
547 - - - - - task: AzureCLI@2
548 - - - - - displayName: 'Establecer variables de entorno -- FRONT'
549 - - - - - inputs:
550 - - - - - azureSubscription: '$(ConnectedServiceName)'
551 - - - - - scriptType: 'bash'
552 - - - - - scriptLocation: 'inlineScript'
553 - - - - - inlineScript: |
554 - - - - - az webapp config appsettings set --name $(WebAppFrontNameContainersPROD) --resource-group $(ResourceGroupName) \
555 - - - - - --settings API_URL="$(PROD2)"
556 - - - -

```

- **4.2.5 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**

```
# ASP.NET Core (.NET Framework)
# Build and test ASP.NET Core projects targeting the full .NET Framework.
# Add steps that publish symbols, save build artifacts, and more:
# https://docs.microsoft.com/azure/devops/pipelines/languages/dotnet-core
```

```
trigger:
- main
```

```
pool:
  vmImage: 'windows-latest'
```

```
variables:
  solution: '**/*.sln'
  buildPlatform: 'Any CPU'
  buildConfiguration: 'Release'
  frontPath: './EmployeeCrudAngular'
  backPath: './EmployeeCrudApi'
  ConnectedServiceName: 'ServiceConnectionARM'
  acrLoginServer: 'mrmingsoft3uccacr.azurecr.io'
  backImageName: 'employee-crud-api'
  frontImageName: 'employee-crud-front'
  ResourceGroupName: 'TP5IngSoft3UCC2024'
  backContainerInstanceNameQA: 'mrm-crud-api-qa'
  backImageTag: 'latest'
  container-cpu-api-qa: 1
  container-memory-api-qa: 1.5
  acrName: 'MRMIngSoft3UCCACR'
  frontContainerInstanceNameQA: 'mrm-crud-front-qa'
  container-cpu-front-qa: 1
  container-memory-front-qa: 1.5
  frontImageTag: 'latest'
  backContainerInstanceNameProd: 'mrm-container-back-prod'
  frontContainerInstanceNameProd: 'mrm-container-front-prod'
  WebAppApiNameContainersQA: 'mrm-crud-api-qa'
  AppServicePlanLinux: 'AppServiceMRM'
```

```

WebAppFrontNameContainersQA: 'mrm-crud-front-qa'
WebAppApiNameContainersPROD: 'mrm-container-back-prod'
WebAppFrontNameContainersPROD: 'mrm-container-front-prod'

stages:
- stage: BuildAndTest
  displayName: "Construir y Probar API y Front"
  jobs:
  - job: BuildDotnet
    displayName: "Construir y Probar API"
    pool:
      vmImage: 'windows-latest'
    steps:
    - task: DotNetCoreCLI@2
      displayName: 'Restaurar paquetes NuGet'
      inputs:
        command: restore
        projects: '$(solution)'
    - task: DotNetCoreCLI@2
      displayName: 'Build de Back'
      inputs:
        command: 'build'
        projects: 'EmployeeCrudApi/EmployeeCrudApi/EmployeeCrudApi.csproj'
        arguments: '--configuration $(buildConfiguration) --output $(buildOutput)/api
--self-contained false'
    - task: DotNetCoreCLI@2
      displayName: 'Publicar Back-End'
      inputs:
        command: publish
        publishWebProjects: true
        arguments: '--configuration $(buildConfiguration) --output
$(Build.ArtifactStagingDirectory)'
        zipAfterPublish: true
    - task: PublishBuildArtifacts@1
      displayName: 'Publicar Artefactos de Back'
      inputs:
        pathToPublish: '$(buildOutput)'
        artifactName: 'drop-back'
        publishLocation: 'Container'
    - task: PublishPipelineArtifact@1
      displayName: 'Publicar Dockerfile de Back'
      inputs:
        targetPath: '$(Build.SourcesDirectory)/docker/api/dockerfile'
        artifact: 'dockerfile-back'
  - job: Frontend
    displayName: 'Build y Análisis del Front-End'
    steps:
    - task: NodeTool@0
      inputs:
        versionSpec: '18.x'
      displayName: 'Instalar Node.js'
    - task: Cache@2
      inputs:
        key: 'npm | "$(Agent.OS)" | EmployeeCrudAngular/package-lock.json'
        path: 'EmployeeCrudAngular/EmployeeCrudAngular/EmployeeCrudAngular/node_modules'
        restoreKeys: |
          npm | "$(Agent.OS)"
      displayName: 'Cachear dependencias de npm'
    - task: CmdLine@2
      displayName: 'Instalar Dependencias de Front'
      inputs:

```

```

        script: npm install
        workingDirectory: '${(frontPath)}'
- task: CmdLine@2
  displayName: 'Build de Front'
  condition: succeeded()
  inputs:
    script: npx ng build --configuration production
    workingDirectory: '${(frontPath)}'
- task: PublishBuildArtifacts@1
  displayName: 'Publicar Artefactos de Front'
  inputs:
    pathToPublish: '${(frontPath)}/dist/employee-crud-angular/browser'
    artifactName: 'drop-front'
    publishLocation: 'Container'
- task: PublishPipelineArtifact@1
  displayName: 'Publicar Dockerfile de Front'
  inputs:
    targetPath: '${(Build.SourcesDirectory)}/docker/front/dockerfile'
    artifact: 'dockerfile-front'

- stage: DockerBuildAndPush
  displayName: 'Construir y Subir Imágenes Docker a ACR'
  dependsOn:

- BuildAndTest
  jobs:
- job: docker_build_and_push
  displayName: 'Construir y Subir Imágenes Docker de Back a ACR'
  pool:
    vmImage: 'ubuntu-latest'
  steps:
- 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: 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'

- job: docker_build_and_push_front
  displayName: 'Construir y Subir Imágenes Docker de Front a ACR'
  pool:
    vmImage: 'ubuntu-latest'
  steps:
    - task: DownloadPipelineArtifact@2
      displayName: 'Descargar Artefactos de Front'
      inputs:
        buildType: 'current'
        artifactName: 'drop-front'
        targetPath: '$(Pipeline.Workspace)/drop-front'
    - task: DownloadPipelineArtifact@2
      displayName: 'Descargar Dockerfile de Front'
      inputs:
        buildType: 'current'
        artifactName: 'dockerfile-front'
        targetPath: '$(Pipeline.Workspace)/dockerfile-front'
    - 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: 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
        tags: 'latest'
    - task: Docker@2
      displayName: 'Subir Imagen Docker de Front a ACR'
      inputs:
        command: push
        repository: $(acrLoginServer)/$(frontImageName)
        tags: 'latest'

# Deploy App Services
- stage: DeployAppServices
  displayName: 'Deploy site to App Services (QA)'
  dependsOn:
    - BuildAndTest
  condition: succeeded()
  pool:
    vmImage: 'windows-latest'
  jobs:
    - job: DeployBack
      displayName: 'Deploy Backend'
      steps:
        - task: DownloadBuildArtifacts@1
          inputs:
            buildType: 'current'
            downloadType: 'single'

```

```

        artifactName: 'drop-back'
        downloadPath: '$(System.ArtifactsDirectory)'
- task: CmdLine@2
  displayName: 'Listar archivos generados del Frontend'
  inputs:
    script: ls -R $(System.ArtifactsDirectory)
- task: AzureRmWebAppDeployment@4
  inputs:
    azureSubscription: 'Azure subscription 1'
    appType: 'webApp'
    WebAppName: 'MiWebApp1'
    package: '$(System.ArtifactsDirectory)/drop-back/**/*.*.zip'

- job: DeployFront
  displayName: 'Deploy Frontend'
  steps:
- task: DownloadBuildArtifacts@1
  inputs:
    buildType: 'current'
    downloadType: 'single'
    artifactName: 'drop-front'
    downloadPath: '$(System.ArtifactsDirectory)'
- task: AzureRmWebAppDeployment@4
  inputs:
    azureSubscription: 'Azure subscription 1'
    appType: 'webApp'
    WebAppName: 'MiWebApp1-prod'
    package: '$(System.ArtifactsDirectory)/drop-front'

# Desplegar en Azure Container Instances (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:
- 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)

- 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)

- job: IntegrationTesting
  displayName: 'Cypress'
  dependsOn: deploy_to_aci_qa
  variables:
    - name: baseUrl
      value: '$(frontContainerInstanceNameQA).eastus.azurecontainer.io'

  steps:
    - script: |
        cd $(Build.SourcesDirectory)/EmployeeCrudAngular
        npm install typescript ts-node
        displayName: 'Install TypeScript'

        # Crear la carpeta results si no existe
    - script: |
        mkdir
$(Build.SourcesDirectory)\EmployeeCrudAngular\EmployeeCrudAngular\EmployeeCrudAngular\cypress\result
s

        displayName: 'Create Results Directory'

```

```

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

- task: PublishTestResults@2
  inputs:
    testResultsFiles:
      '$(Build.SourcesDirectory)/EmployeeCrudAngular/EmployeeCrudAngular/EmployeeCrudAngular/cypress/resul
ts/*.xml'
    testRunTitle: 'Cypress E2E Tests (QA)'
    displayName: 'Publish Cypress Test Results'

#-----
### STAGE DEPLOY TO AZURE APP SERVICE QA
#-----
- stage: DeployImagesToAppServiceQA
  displayName: 'Desplegar Imagenes en Azure App Service (QA)'
  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)" \

#-----
# DEPLOY DOCKER FRONT IMAGE TO AZURE APP SERVICE (QA)
#-----

- task: AzureCLI@2
  displayName: 'Verificar y crear el recurso Azure App Service para Front (PROD) si no
existe'
  inputs:
    azureSubscription: '$(ConnectedServiceName)'
    scriptType: 'bash'
    scriptLocation: 'inlineScript'
    inlineScript: |
      # Verificar si el App Service para el Front ya existe
      if ! az webapp list --query "[?name=='$(WebAppFrontNameContainersQA)']" --resource-group=$(ResourceGroupName) --output tsv | grep -q '^1$'; then
        echo "El App Service para Front PROD 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" # Especifica una
        imagen temporal para permitir la creación
      else
        echo "El App Service para Front PROD 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" --output tsv)

      # Establecer variables de entorno
      az webapp config appsettings set --name $(WebAppFrontNameContainersQA)
--resource-group $(ResourceGroupName) \
        --settings API_URL="$(PROD)"

- stage: DeployToProdWebApps
  displayName: 'Deploy to Azure Web Apps (Production)'
  dependsOn:
    - 'DeployAppServices'
  condition: succeeded()
  jobs:
    - deployment: DeployBackProd
      displayName: 'Deploy Backend to Production'
      environment:
        name: 'Production'
      strategy:
        runOnce:
          deploy:
            steps:
              - task: DownloadBuildArtifacts@1
                inputs:
                  buildType: 'current'
                  downloadType: 'single'
                  artifactName: 'drop-back'
                  downloadPath: '$(System.ArtifactsDirectory)'
              - task: AzureRmWebAppDeployment@4
                inputs:
                  azureSubscription: '$(ConnectedServiceName)'

```

```

        appType: 'webApp'
        WebAppName: 'MiWebApp1-Prod'
        package: '$(System.ArtifactsDirectory)/drop-back/**/*.*.zip'
- job: DeployFrontProd
  displayName: 'Deploy Frontend to Production'
  steps:
  - task: DownloadBuildArtifacts@1
    inputs:
      buildType: 'current'
      downloadType: 'single'
      artifactName: 'drop-front'
      downloadPath: '$(System.ArtifactsDirectory)'
  - task: AzureRmWebAppDeployment@4
    inputs:
      azureSubscription: '$(ConnectedServiceName)'
      appType: 'webApp'
      WebAppName: 'MiWebApp1-prod'
      package: '$(System.ArtifactsDirectory)/drop-front'

# DEPLOY A PROD
- stage: DeployToACIPROD
  displayName: 'Desplegar PROD'
  dependsOn:
  - DeployToACIQA
  jobs:
  - deployment: DeployToProd
    displayName: 'Desplegar PRODD'
    environment:
      name: 'Production'
    strategy:
      runOnce:
        deploy:
          steps:

# BACK
  - 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 "Connection String: $(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)

# FRONT
- 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: $(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 $(frontContainerInstanceNameProd) \
        --ports 80 \
        --environment-variables API_URL="$(PROD)" \
        --restart-policy Always \
        --cpu $(container-cpu-front-qa) \
        --memory $(container-memory-front-qa)

#-----
### STAGE DEPLOY TO AZURE APP SERVICE PROD
#-----
- stage: DeployImagesToAppServiceProd
  displayName: 'Desplegar Imágenes en Azure App Service (PROD)'
  dependsOn:
    - DeployImagesToAppServiceQA
  condition: succeeded()
  jobs:
    - deployment: DeployImagesToAppServiceProd
      displayName: 'Desplegar Imágenes de API y Front en Azure App Service (PROD)'
      environment:
        name: 'Production'
      strategy:
        runOnce:
          deploy:
            steps:
              - 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(0)" -o tsv | grep -q '^1$'; then
        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-container-image-name
"nginx" # Especifica una imagen temporal para permitir la creación
    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):$(backImageTag) \
        --container-registry-url https://$(acrLoginServer) \
        --container-registry-user $(acrName) \
        --container-registry-password $(az acr credential show --name $(acrName) --query
"passwords[0].value" -o tsv)
    # Establecer variables de entorno para API
    az webapp config appsettings set --name $(WebAppApiNameContainersProd)
--resource-group $(ResourceGroupName) \
        --settings ConnectionStrings__DefaultConnection="$(cnn-string-prod)"
- task: AzureCLI@2
    displayName: 'Verificar y crear el recurso Azure App Service para Front (PROD) si no
existe'

inputs:
    azureSubscription: '$(ConnectedServiceName)'
    scriptType: 'bash'
    scriptLocation: 'inlineScript'
    inlineScript: |
        # Verificar si el App Service para el Front ya existe
        if ! az webapp list --query "[?name=='$(WebAppFrontNameContainersProd)']" &&
resourceGroup=='$(ResourceGroupName)'] | length(0)" -o tsv | grep -q '^1$'; then
            echo "El App Service para Front PROD 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" # Especifica una imagen temporal para permitir la creación
        else
            echo "El App Service para Front PROD 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 para el Front
        az webapp config appsettings set --name $(WebAppFrontNameContainersProd)
--resource-group $(ResourceGroupName) \
            --settings API_URL="$(PROD2)"

```



