

Informe de CI/CD

Se presentará primeramente los ambientes para los que se desarrollaron los Pipelines. Luego, se explicará brevemente que realiza cada Pipeline para cada ambiente y finalmente se indicará como es el Deploy de la aplicación en cada uno de sus ambientes.

Características de herramientas utilizadas:

Para el desarrollo del Pipeline se utilizó un repositorio creado dentro de Azure DevOps que está vinculado con el Pipeline que también yace dentro del mismo proyecto en Azure DevOps.

Se asoció un Agente de Azure en el servidor donde está la Knowledge Base, el mismo corre como servicio y sirve para ejecutar el código desarrollado en el Pipeline.

Documentación: <https://learn.microsoft.com/es-es/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=yaml%2Cbrowser>

Dentro del desarrollo del Pipeline se utilizaron Scripts realizados en Powershell y funciones de MSBuild para lograr el objetivo.

Documentación MSBuild:
<https://wiki.genexus.com/commwiki/wiki?3908,MSBuild+Tasks>

Cada Pipeline cuenta con un archivo README.md que resume el funcionamiento y el flujo de trabajo de estos.

Características de cada ambiente:

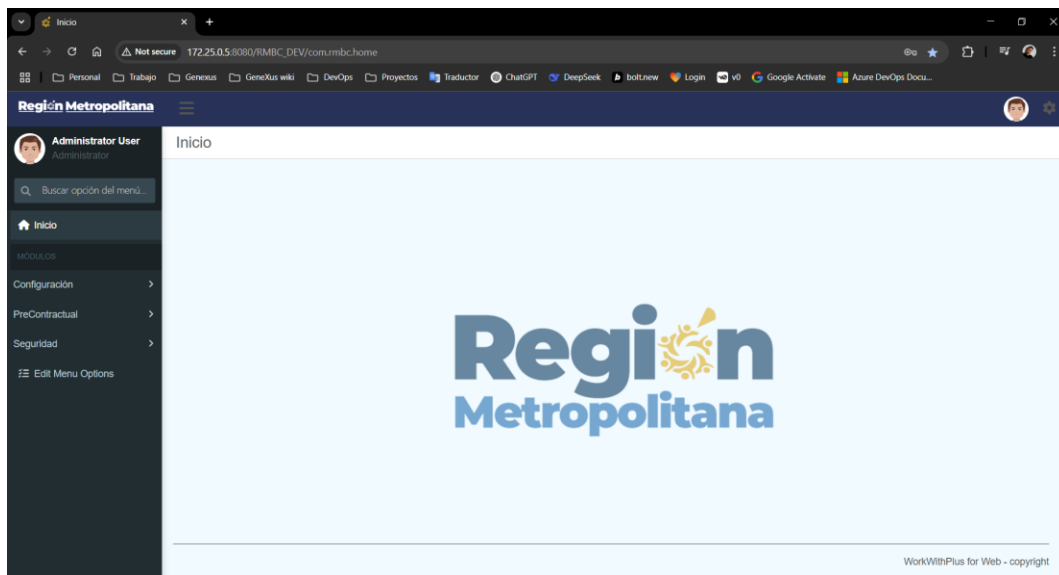
Ambiente de Desarrollo (DEV):

Este ambiente es la primera instancia dentro del flujo de los Pipelines.

La Knowledge Base (KB) utiliza la base de datos Postgres 'rmbc' y 'rmbc_gam' dentro del servidor para el almacenamiento de datos de la aplicación y de GAM respectivamente.

La URL de publicación de este ambiente es:

http://172.25.0.5:8080/RMBC_DEV/com.rmbc.home



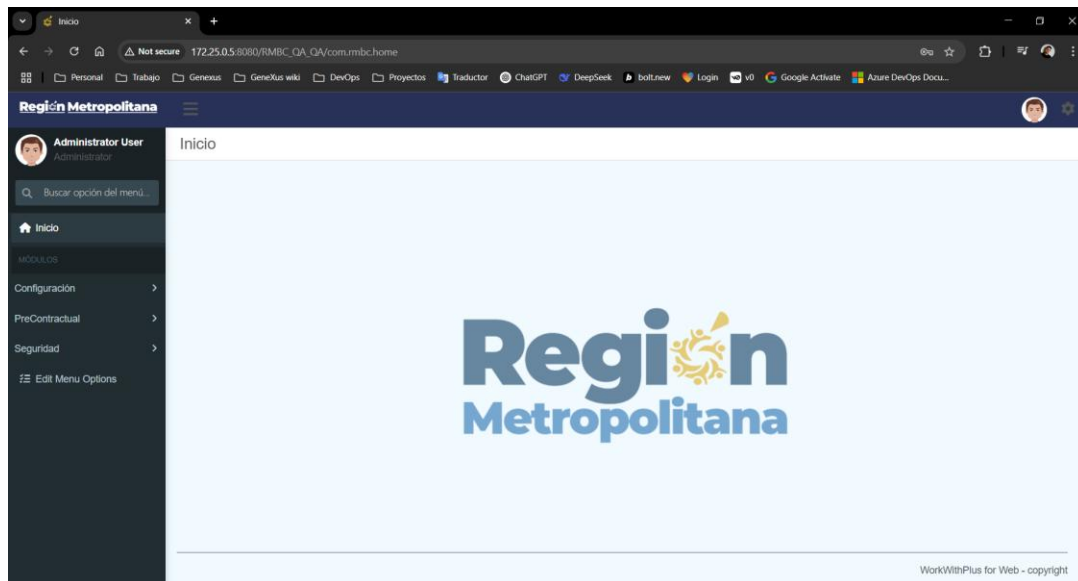
Ambiente de Testing (QA):

Este ambiente es la segunda instancia dentro del flujo de los Pipelines.

La Knowledge Base (KB) utiliza la base de datos Postgres 'rmbc_qa' y 'rmbc_gam_qa' dentro del servidor para el almacenamiento de datos de la aplicación y de GAM respectivamente.

La URL de publicación de este ambiente es:

http://172.25.0.5:8080/RMBC_QA_QA/com.rmbc.home



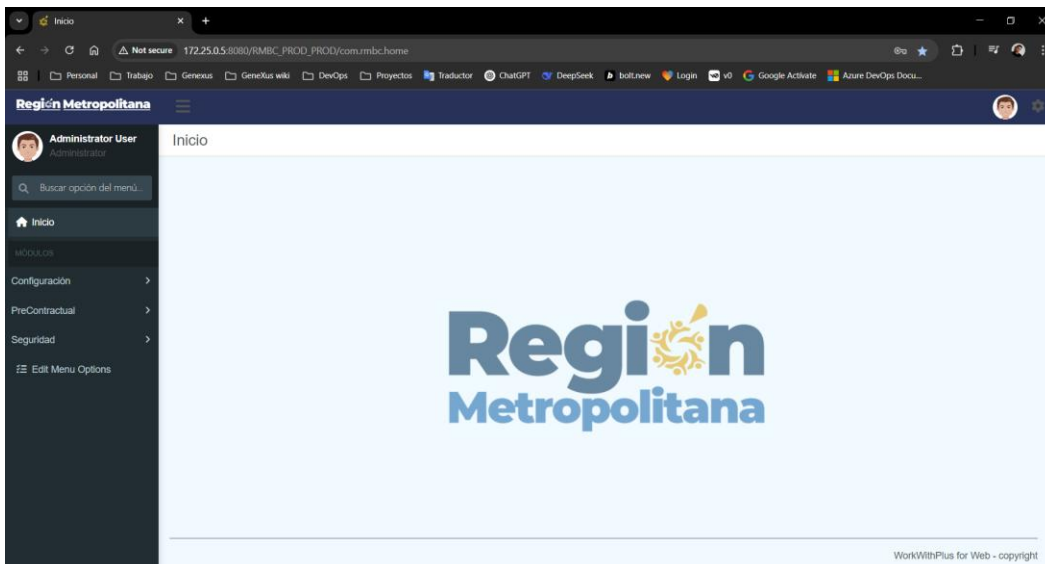
Ambiente de Producción (PROD):

Este ambiente es la última instancia dentro del flujo de los Pipelines.

La Knowledge Base (KB) utiliza la base de datos Postgres 'rmbc_prd' y 'rmbc_gam_prd' dentro del servidor para el almacenamiento de datos de la aplicación y de GAM respectivamente.

La URL de publicación de este ambiente es:

http://172.25.0.5:8080/RMBC_PROD_PROD/com.rmbc.home



Implementación:

Pipeline desarrollado para el ambiente de Desarrollo (DEV):

El Pipeline se llama 'Azure_RMBC_DEV' y consta de cinco Stages: 'Update KB DEV', 'Build KB DEV', 'Create WAR file DEV', 'Deploy a Tomcat' y 'Push al repositorio de Azure'.

Stage 'Update KB DEV':

```
.. - stage: UpdateKBDEV
  .. displayName: 'Update KB DEV'
  .. jobs:
    .. - job: UpdateJob
      .. steps:
        .. - script: call $(updateMSBuildScript)
          .. displayName: 'Actualizar KB del ambiente de desarrollo'
```

Ester Stage se encarga de descargar los cambios realizados a la Knowledge Base en Genexus Server para el ambiente de Desarrollo.

Stage 'Build KB DEV':

```
.. - stage: BuildKBDEV
  .. displayName: 'Build KB DEV'
  .. dependsOn: UpdateKBDEV
  .. condition: succeeded()
  .. jobs:
    .. - job: BuildJob
      .. steps:
        .. - script: call $(buildMSBuildScript)
          .. displayName: 'Ejecutar Build de la KB del ambiente de desarrollo'
```

Este Stage compila la Knowledge Base con los cambios obtenidos por el Stage anterior.

Stage 'Create WAR file DEV':

```
..- stage: CreateWARFileDEV
  ..- displayName: 'Create WAR file DEV'
  ..- dependsOn: BuildKBDEV
  ..- condition: succeeded()
  ..- jobs:
    ..- job: CreateWARFileJob
      ..- steps:
        ..- script: call $(createGXDPROJFileScript)
          ..- displayName: 'Crear archivo GXDPROJ del ambiente de desarrollo'
        ..- script: call $(createWARFileScript)
          ..- displayName: 'Crear archivo WAR del ambiente de desarrollo'
```

Este Stage Ejecuta un script para crear el archivo GXDPROJ y luego otro script para crear el archivo WAR, este archivo se utilizará posteriormente para desplegar la aplicación.

Stage 'Deploy a Tomcat':

```
..- stage: DeployToTomcat
  ..- displayName: 'Deploy a Tomcat'
  ..- dependsOn: CreateWARFileDEV
  ..- condition: succeeded()
  ..- jobs:
    ..- job: DeployToTomcatJob
      ..- steps:
        Settings
        ..- task: PowerShell@2
          ..- displayName: 'Desplegar en aplicacion web servidor local Tomcat 10.1'
          ..- inputs:
            ..- filePath: 'ps1\deployToTomcat.ps1'
            ..- arguments: >
              ..- -WarFilePath "$(WARFilePath)"
```

Este Stage mueve el archivo WAR generado anteriormente a la carpeta 'webapps' del servidor local de Tomcat.

Stage 'Push al repositorio de Azure':

```
-- stage: PushToAzure
-- displayName: 'Push al repositorio de Azure'
-- dependsOn: DeployToTomcat
-- condition: succeeded()
-- jobs:
-- -- job: DeployOnAzureJob
-- -- steps:
-- -- Settings
-- -- -- task: PowerShell@2
-- -- -- displayName: 'Realizar un Push del archivo WAR al repositorio de Azure'
-- -- -- inputs:
-- -- -- -- filePath: 'ps1\pushToAzureRepository.ps1'
-- -- -- -- arguments: >
-- -- -- -- -BaseDirectory 'C:\RMBC_pipeline'
-- -- -- -- -PATToken $(PATToken)
-- -- -- -- -WarFilePath "$(WARFilePath)"
```

Este Stage realiza un push al repositorio de Azure 'Azure_RMBC_DEV_deploy' del archivo WAR generado previamente. Una vez ejecutado el script queda en funcionamiento la aplicación web en el URL indicado.

Pipeline desarrollado para el ambiente de Testing (QA):

El Pipeline se llama 'Azure_RMBC_QA' y consta de cinco Stages: 'Merge KB QA', 'Build KB QA', 'Create WAR file QA', 'Deploy a Tomcat' y 'Push al repositorio de Azure'.

Stage 'Merge KB QA':

```
..- stage: MergeKBQA
..  displayName: 'Merge KB QA'
..  jobs:
..    - job: MergeJob
..      steps:
..        - powershell: |
..          $lastMergeDate = & ps1\getLastMergeDate.ps1
..          Write-Host "##vso[task.setvariable variable=lastMergeDate]$lastMergeDate"
..          Write-Host "Ultima fecha de merge obtenida: $lastMergeDate"
..          displayName: 'Obtener la última fecha de merge'
..        - script: call $(mergeMSBuildScript)
..          displayName: 'Realizar merge al ambiente de QA'
..        - powershell: ps1\updateLastMergeDate.ps1
..          displayName: 'Actualizar la fecha de merge'
```

Este Stage realiza un Merge trayendo los cambios realizados en la Knowledge Base del ambiente de Desarrollo.

Stage 'Build KB QA':

```
..- stage: BuildKBQA
..  displayName: 'Build KB QA'
..  dependsOn: MergeKBQA
..  condition: succeeded()
..  jobs:
..    - job: BuildJob
..      steps:
..        - script: call $(buildMSBuildScript)
..          displayName: 'Ejecutar Build de la KB del ambiente de QA'
```

Este Stage compila la Knowledge Base con los cambios obtenidos por el Stage anterior.

Stage 'Create WAR file QA':

```
..- stage: CreateWARFileQA
..  displayName: 'Create WAR file QA'
..  dependsOn: BuildKBQA
..  condition: succeeded()
..  jobs:
..    - job: CreateWARFileJob
..      steps:
..        - script: call $(createGXDPROJFileScript)
..          displayName: 'Crear archivo GXDPROJ del ambiente de QA'
..        - script: call $(createWARFileScript)
..          displayName: 'Crear archivo WAR del ambiente de QA'
```

Este Stage al igual que en el pipeline para el ambiente de Desarrollo crea el archivo GXDPROJ y luego el archivo WAR que se usara posteriormente.

Stage 'Deploy a Tomcat':

```
..- stage: DeployToTomcat
..  displayName: 'Deploy a Tomcat'
..  dependsOn: CreateWARFileQA
..  condition: succeeded()
..  jobs:
..    - job: DeployToTomcatJob
..      steps:
..        Settings
..        - task: PowerShell@2
..          displayName: 'Desplegar en aplicacion web servidor local Tomcat 10.1'
..          inputs:
..            filePath: 'ps1/deployToTomcat.ps1'
..            arguments: >
..              -WarFilePath "$(WARFilePath)"
```

Este Stage al igual que en el Pipeline para el ambiente de Desarrollo mueve el archivo WAR generado anteriormente a la carpeta 'webapps' del servidor local de Tomcat.

Stage 'Push al repositorio de Azure':

```
..- stage: PushToAzure
..- displayName: 'Push al repositorio de Azure'
..- dependsOn: DeployToTomcat
..- condition: succeeded()
..- jobs:
..- - job: DeployOnAzureJob
..- - steps:
    Settings
    ..- task: PowerShell@2
    ..- displayName: 'Realizar un Push del archivo WAR al repositorio de Azure'
    ..- inputs:
    ..- filePath: 'ps1/pushToAzureRepository.ps1'
    ..- arguments: >
    ..- -BaseDirectory "C:\RMBC_pipeline\Ambientes\QA"
    ..- -PATToken $(PATToken)
    ..- -WarFilePath "$(WARFilePath)"
```

Este Stage al igual que el Pipeline para el ambiente de Desarrollo realiza un push al repositorio de Azure 'Azure_RMBC_QA_deploy' del archivo WAR generado previamente. Una vez ejecutado el script queda en funcionamiento la aplicación web en el URL indicado.

Pipeline desarrollado para el ambiente de Desarrollo (PROD):

El Pipeline se llama 'Azure_RMBC_PROD' y consta de cinco Stage: 'Merge KB PROD', 'Build KB PROD', 'Create WAR file PROD', 'Deploy a Tomcat' y 'Push al repositorio de Azure'.

Stage 'Merge KB PROD':

```
... - stage: MergeKBPROD
...   displayName: 'Merge KB PROD'
...   jobs:
...     - job: MergeJob
...       steps:
...         - powershell: |
...             $lastMergeDate = & ps1\getLastMergeDate.ps1
...             Write-Host "##vso[task.setvariable variable=lastMergeDate]$lastMergeDate"
...             Write-Host "Ultima fecha de merge obtenida: $lastMergeDate"
...             displayName: 'Obtener la última fecha de merge'
...         - script: call $(mergeMSBuildScript)
...             displayName: 'Realizar merge al ambiente de PROD'
...         - powershell: ps1\updateLastMergeDate.ps1
...             displayName: 'Actualizar la fecha de merge'
```

Este Stage al igual que el Pipeline para el ambiente de Testing realiza un Merge trayendo los cambios realizados en la Knowledge Base del ambiente de Testing.

Stage 'Build KB PROD':

```
..- stage: BuildKBPROD
..- displayName: 'Build KB PROD'
..- dependsOn: MergeKBPROD
..- condition: succeeded()
..- jobs:
..- - job: BuildJob
..-   steps:
..-   - script: call $(buildMSBuildScript)
..-   - displayName: 'Ejecutar Build de la KB del ambiente de PROD'
```

Este Stage compila la Knowledge Base con los cambios obtenidos por el Stage anterior.

Stage 'Create WAR file PROD':

```
..- stage: CreateWARFilePROD
..- displayName: 'Create WAR file PROD'
..- dependsOn: BuildKBPROD
..- condition: succeeded()
..- jobs:
..- - job: CreateWARFileJob
..-   steps:
..-   - script: call $(createGXDPROJFileScript)
..-   - displayName: 'Crear archivo GXDPROJ del ambiente de PROD'
..-   - script: call $(createWARFileScript)
..-   - displayName: 'Crear archivo WAR del ambiente de PROD'
```

Este Stage al igual que en los Pipelines para los ambientes de Desarrollo y de Testing crea el archivo GXDPROJ y luego el archivo WAR que se usara posteriormente.

Stage 'Deploy a Tomcat':

```
..- stage: DeployToTomcat
..- displayName: 'Deploy a Tomcat'
..- dependsOn: CreateWARFilePROD
..- condition: succeeded()
..- jobs:
..- - job: DeployToTomcatJob
..-   steps:
..-     Settings
..-     - task: PowerShell@2
..-       displayName: 'Desplegar en aplicacion web servidor local Tomcat 10.1'
..-       inputs:
..-         filePath: 'ps1/deployToTomcat.ps1'
..-         arguments: >
..-         -WarFilePath "$(WARFilePath)"
```

Este Stage al igual que en los Pipelines para los ambientes de Desarrollo y Testing mueve el archivo WAR generado anteriormente a la carpeta 'webapps' del servidor local de Tomcat.

Stage 'Push al repositorio de Azure':

```
.. - stage: PushToAzure
.. - displayName: 'Push al repositorio de Azure'
.. - dependsOn: DeployToTomcat
.. - condition: succeeded()
.. - jobs:
.. - - job: DeployOnAzureJob
.. - - steps:
.. - - Settings
.. - - task: PowerShell@2
.. - - displayName: 'Realizar un Push del archivo WAR al repositorio de Azure'
.. - - inputs:
.. - - filePath: 'ps1/pushToAzureRepository.ps1'
.. - - arguments: >
.. - - -BaseDirectory "C:\RMBC_pipeline\Ambientes\PROD"
.. - - -PATToken $(PATToken)
.. - - -WarFilePath "$(WARFilePath)"
```

Este Stage al igual que en los Pipelines para los ambientes de Desarrollo y Testing realiza un push al repositorio de Azure 'Azure_RMBC_PROD_deploy' del archivo WAR generado previamente. Una vez ejecutado el script queda en funcionamiento la aplicación web en el URL indicado.

Scripts:

Script 'deployToTomcat.ps1':

A continuación, se realizará un resumen del Script 'deployToTomcat.ps1' que se ejecuta para realizar el Deploy de cada ambiente al servidor local de Tomcat.

Primeramente, se procede a detener el proceso asociado al servidor de Tomcat:

```
# Detener Tomcat
Write-Host "Deteniendo Tomcat..."
Stop-Service -Name "Tomcat10.1" -Force -ErrorAction SilentlyContinue
Start-Sleep -Seconds 5
```

Una vez detenido el servidor de Tomcat, se elimina la carpeta que existía previamente de la aplicación y también se borra el archivo WAR generado anteriormente en otra ejecución del Pipeline:

```
# Eliminar carpeta y WAR anterior si existen
if (Test-Path $OldWarFolder) {
    Write-Host "Eliminando carpeta antigua: $OldWarFolder"
    Remove-Item -Recurse -Force $OldWarFolder
}

if (Test-Path $OldWarPath) {
    Write-Host "Eliminando WAR antiguo: $OldWarPath"
    Remove-Item -Force $OldWarPath
}
```

Luego de borrar los archivos viejos, se copia el archivo WAR generado por el Pipeline:

```
# Copiar nuevo WAR a Tomcat
Write-Host "Copiando archivo $WarFilePath en directorio $TargetPath"
Copy-Item -Path $WarFilePath -Destination $TargetPath -Force
```

Finalmente se vuelve a iniciar el proceso asociado al servidor de Tomcat:

```
# Iniciar Tomcat
Write-Host "Iniciando Tomcat..."
Start-Service -Name "Tomcat10.1"
Write-Host "Despliegue completado."
```

Una vez ejecutado este Script se tendrá una nueva versión de la aplicación web ejecutándose para el ambiente dado.

Script 'pushToAzureRepository.ps1':

Este Script realiza el push al repositorio de Azure y se ejecuta en cada ambiente.

Primero, se cambia de carpeta base donde se está y se define la ruta del repositorio de Azure donde se va a realizar el push del archivo WAR

```
# Cambiar al directorio base
Write-Host "Cambiendo al directorio base: $BaseDirectory"
Set-Location -Path $BaseDirectory

# Definir ruta del repositorio
$LastDeployPath = Join-Path -Path $BaseDirectory -ChildPath "last_deploy_QA"
```

Se consulta si la carpeta del proyecto ya existe, si es así, se intenta realizar un 'git pull', si este comando falla se realiza un 'git clone' del repositorio.

Si la carpeta del repositorio no existe se realiza un 'git clone' directamente.

```
# Verificar si el directorio last_deploy_QA existe
if (Test-Path $LastDeployPath) {
    Write-Host "El directorio 'last_deploy_QA' ya existe. Intentando git pull..."
    Set-Location -Path $LastDeployPath

    # Intentar git pull, si falla intentar git clone desde cero
    try {
        git reset --hard
        git pull origin main
        Write-Host "Git pull completado con éxito."
    } catch {
        Write-Host "ERROR: No se pudo hacer git pull. Intentando clonar de nuevo..." -ForegroundColor Yellow
        Set-Location -Path $BaseDirectory
        Remove-Item -Path $LastDeployPath -Recurse -Force -ErrorAction SilentlyContinue
        try {
            git clone "https://$PATToken@dev.azure.com/regionmetropolitana/Observatorio%20Metropolitano/_git/Azure_RMBC_QA_deploy"
            $LastDeployPath
            Write-Host "Repositorio clonado exitosamente en '$LastDeployPath'."
        } catch {
            Write-Host "ERROR: No se pudo clonar el repositorio." -ForegroundColor Red
            exit 1
        }
    }
} else {
    Write-Host "El directorio 'last_deploy_QA' no existe. Clonando el repositorio..."

    try {
        git clone "https://$PATToken@dev.azure.com/regionmetropolitana/Observatorio%20Metropolitano/_git/Azure_RMBC_QA_deploy"
        $LastDeployPath
        Write-Host "Repositorio clonado exitosamente en '$LastDeployPath'."
    } catch {
        Write-Host "ERROR: No se pudo clonar el repositorio." -ForegroundColor Red
        exit 1
    }
}
```

Se realiza una validación de si existe la carpeta del repositorio descargado/actualizado:

```
# Validar si el directorio last_deploy_QA existe después de clonación o pull
if (-Not (Test-Path $LastDeployPath)) {
    Write-Host "ERROR: El directorio 'last_deploy_QA' no se encuentra después de la operación. Terminando la ejecución."
    -ForegroundColor Red
    exit 1
}
```

Una vez con el repositorio descarga o actualizado se copia el archivo WAR generado por el Pipeline:

```
# Copiar el archivo .war al repositorio
Write-Host "Copiando el archivo .war: $WarFilePath"
Copy-Item -Path $WarFilePath -Destination $LastDeployPath -Force
```

Se cambia la carpeta base a donde está el repositorio descargado y se configura el usuario de git antes de realizar el push al repositorio:

```
# Cambiar al directorio del repositorio clonado o actualizado
Set-Location -Path $LastDeployPath

# Configuración de usuario Git
Write-Host "Configurando usuario Git..."
git config user.name "Pipeline de Azure"
git config user.email "devops@regionmetropolitana.gov.co"
```

Se cambia el nombre del archivo WAR para agregarle la fecha actual:

```
# Obtener la fecha y hora actual
$CurrentDateTime = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
Write-Host "Fecha y hora actual: $CurrentDateTime"
```

Se agrega el comentario al realizar el commit y se realiza el push:

```
# Realizar commit y push
Write-Host "Realizando commit y push al repositorio..."
git add .
git commit -m "Subiendo archivo .war generado automáticamente el $CurrentDateTime"
git push
```

Script 'getLastMergeDate.ps1':

Este Script se ejecuta tanto en el Pipeline del ambiente de Testing como de Produccion. Su función es obtener la ultima fecha de Merge realizada que esta dentro de un archivo TXT.

Se comprueba si el archivo TXT donde está la última fecha de realización de Merge y si existe, devuelve el contenido de este.

```
if (Test-Path $lastMergeDateFile) {  
    $lastMergeDate = Get-Content $lastMergeDateFile | ForEach-Object { $_.Trim() }  
} else {  
    $lastMergeDate = $defaultDate  
    Set-Content -Path $lastMergeDateFile -Value $lastMergeDate  
}  
  
Write-Output $lastMergeDate
```

Script 'updateLastMergeDate.ps1':

Este Script se ejecuta tanto en el Pipeline del ambiente de Testing como de Produccion. Su función es actualizar la fecha de ultima ejecución del Merge en el archivo.

```
Set-Content -Path $lastMergeDateFile -Value $currentDate  
  
Write-Output "Fecha de merge actualizada a: $currentDate"
```