

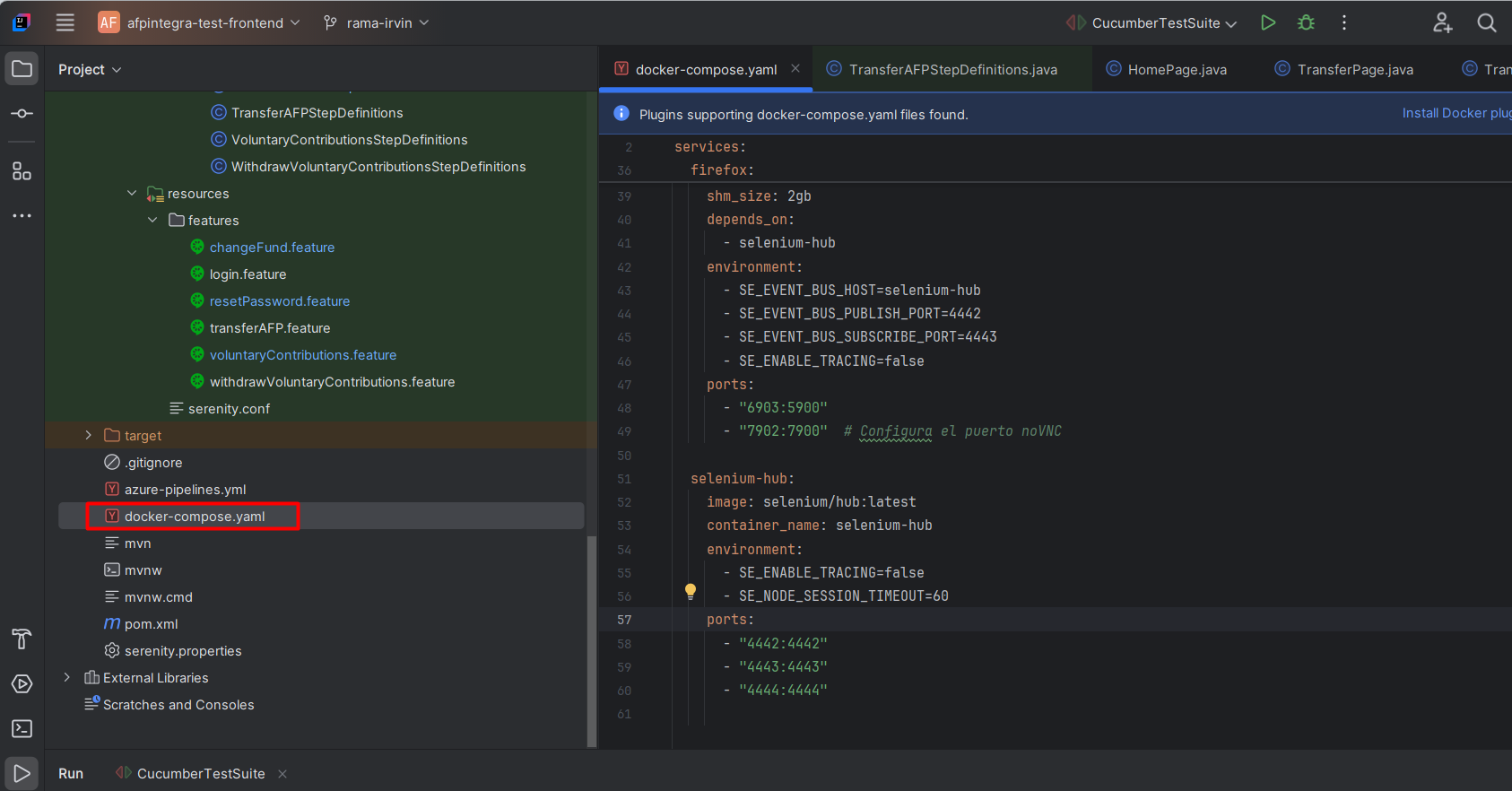
SERENITY

serenity {  
 encoding = "UTF-8"  
 compress.filenames = true  
 take.screenshots = FOR\_FAILURES *# Cambiar a FOR\_FAILURES para capturar solo en fallos*  
report.include.step.details = false *# Excluir detalles adicionales de pasos*  
report.include.hooks = false *# Excluir tiempos de hooks (@Before y @After)*  
report.show.manual.tests = false *# No mostrar pruebas manuales*  
report.show.released.tests = false *# No mostrar pruebas liberadas*  
report.generate.manual.tests = false *# No generar evidencias de pruebas manuales*  
report.generate.released.tests = false *# No generar evidencias de pruebas liberadas*  
 arn {  
 deviceFarm = "arn:aws:devicefarm:us-west-2:740893539825:testgrid-project:55a2dd70-1fe9-47eb-aa64-e16ffd3371e4"  
 }  
}  
  
 environments {  
 default{  
 login.url = "<https://test.afpintegra.pe/iniciar-sesion>"  
 principal.url = "<https://test.afpintegra.pe/>"  
 outlook.url = "<https://go.microsoft.com/fwlink/p/?LinkID=2125442&amp;clcid=0x409&amp;culture=en-us&amp;country=us>"  
 }  
 dev {  
 login.url = "<https://test.afpintegra.pe/iniciar-sesion>"  
 principal.url = "<https://test.afpintegra.pe/>"  
 outlook.url = "<https://go.microsoft.com/fwlink/p/?LinkID=2125442&amp;clcid=0x409&amp;culture=en-us&amp;country=us>"  
 }  
}

AZURE

trigger:  
 - develop  
  
parameters:  
 - name: tags  
 displayName: 'Tags'  
 type: string  
 default: '@esc01resetPassword'  
 - name: environment  
 displayName: 'Environment'  
 type: string  
 default: 'dev'  
 values:  
 - 'dev'  
 - 'uat'  
 - name: browser  
 displayName: 'Browser'  
 type: string  
 default: 'chrome'  
 values:  
 - 'chrome'  
 - 'firefox'  
 - 'edge'  
 - name: machine  
 displayName: 'Machine'  
 type: string  
 default: 'windows-latest'  
 values:  
 - 'windows-latest'  
 - 'ubuntu-latest'  
 - name: parallelExecutions  
 displayName: 'Parallel Executions'  
 type: number  
 default: 1  
 - name: sendEmail  
 displayName: 'Send email'  
 type: boolean  
 default: true  
  
variables:  
 SERENITY\_REPORTS: '$(Build.ArtifactStagingDirectory)/serenity-reports.zip'  
 CUCUMBER\_REPORTS: '$(Build.ArtifactStagingDirectory)/cucumber-reports.zip'  
 EXECUTION\_ENV: 'AWS' *# Establece la variable de entorno a AWS*  
  
stages:  
 - stage: 'Build'  
 displayName: 'Construcción'  
 variables:  
 - group: TestBase  
 - group: Email\_Settings  
 - group: Email\_Recipients  
 jobs:  
 - job: 'Build'  
 displayName: 'Construcción del artefacto'  
 pool:  
 vmImage: ${{ parameters.machine }}  
 strategy:  
 parallel: ${{ parameters.parallelExecutions }}  
 steps:  
 - task: Maven@4  
 env:  
 AWS\_ACCESS\_KEY\_ID: $(AWS\_ACCESS\_KEY\_ID)  
 AWS\_SECRET\_ACCESS\_KEY: $(AWS\_SECRET\_ACCESS\_KEY)  
 inputs:  
 mavenPomFile: 'pom.xml'  
 mavenOptions: '-Xmx3072m'  
 javaHomeOption: 'JDKVersion'  
 jdkVersionOption: 1.11  
 jdkArchitectureOption: 'x64'  
 publishJUnitResults: true  
 testResultsFiles: '\*\*/surefire-reports/TEST-\*.xml'  
 goals: 'clean install'  
 options: '-Dcucumber.filter.tags="${{ parameters.tags }}" -Denvironment=${{ parameters.environment }} -Dbrowser=${{ parameters.browser }}'  
  
 - script: |  
 echo "Generando informes de Serenity y Cucumber..."  
 mvn serenity:aggregate  
 displayName: 'Generar informes de Serenity y Cucumber'  
 condition: always()  
  
 - task: ArchiveFiles@2  
 condition: always()  
 inputs:  
 rootFolderOrFile: 'target/site/serenity'  
 includeRootFolder: false  
 archiveType: 'zip'  
 archiveFile: '$(SERENITY\_REPORTS)'  
 replaceExistingArchive: true  
 displayName: 'Archivar informes de Serenity'  
  
 - task: ArchiveFiles@2  
 condition: always()  
 inputs:  
 rootFolderOrFile: 'target/cucumber-reports'  
 includeRootFolder: false  
 archiveType: 'zip'  
 archiveFile: '$(CUCUMBER\_REPORTS)'  
 replaceExistingArchive: true  
 displayName: 'Archivar informes de Cucumber'  
  
 - task: PowerShell@2  
 displayName: 'Enviar reporte por correo'  
 condition: and(always(), eq('${{ parameters.sendEmail }}', true))  
 inputs:  
 targetType: 'inline'  
 script: |  
 $smtpServer = "smtp.office365.com"  
 $smtpPort = 587  
 $smtpUser = "$(smtpUser)" # Usamos las variables del grupo aquí  
 $smtpPassword = "$(smtpPassword)" # Usamos las variables del grupo aquí  
 $recipients = "$(smtpRecipients)" # Usamos las variables del grupo aquí  
 # $subject = "Resultados de Pruebas Automatizadas"  
 $buildLink = "$(System.TeamFoundationCollectionUri)$(System.TeamProject)/\_build/results?buildId=$(Build.BuildId)&view=results"  
   
 # Obtener la hora local de PE   
 $timeZone = [System.TimeZoneInfo]::FindSystemTimeZoneById('SA Pacific Standard Time')  
 $peTime = [System.TimeZoneInfo]::ConvertTimeBySystemTimeZoneId([System.DateTime]::UtcNow, $timeZone.Id)  
  
 # Determinar el saludo según la hora del día en PE  
 $hour = $peTime.Hour  
 if ($hour -ge 6 -and $hour -lt 12) {  
 $greeting = "Buenos días"  
 } elseif ($hour -ge 12 -and $hour -lt 18) {  
 $greeting = "Buenas tardes"  
 } else {  
 $greeting = "Buenas noches"  
 }  
   
 ##########################################################################################  
 # Leer el archivo cucumber.json  
 $cucumberReportPath = "target/cucumber-reports/cucumber.json"  
 $cucumberReport = Get-Content $cucumberReportPath -Raw -Encoding UTF8 | ConvertFrom-Json  
 ##########################################################################################  
   
 #Inicializar variables para la hora de inicio y duración total  
 $startTime = [DateTime]::Parse($cucumberReport[0].elements[0].start\_timestamp)  
 $totalDurationNanoseconds = 0  
 $totalTestCases = 0  
 $totalPassed = 0  
 $totalFailed = 0  
   
 # Sumar las duraciones de todos los pasos en todos los escenarios  
 foreach ($feature in $cucumberReport) {  
 foreach ($scenario in $feature.elements) {  
 $scenarioDurationNanoseconds = 0  
 $scenarioStatus = "passed"  
 #Sumar las duraciones de todos los pasos del escenario  
 foreach ($step in $scenario.steps) {  
 $scenarioDurationNanoseconds += $step.result.duration  
 if ($step.result.status -eq "failed") {  
 $scenarioStatus = "failed"  
 }   
 }  
   
 # Contar el escenario según su estado  
 $totalTestCases++  
 if ($scenarioStatus -eq "passed") {  
 $totalPassed++  
 } elseif ($scenarioStatus -eq "failed") {  
 $totalFailed++  
 }  
   
 #Agregar la duración del escenario a la duración total  
 $totalDurationNanoseconds += $scenarioDurationNanoseconds  
 }  
 }  
  
 # Convertir la duración total de nanosegundos a un formato legible (hh:mm:ss)  
 $totalDurationSeconds = [math]::Round($totalDurationNanoseconds / 1000000000, 2)  
 $totalDurationFormatted = [TimeSpan]::FromSeconds($totalDurationSeconds).ToString("hh\:mm\:ss")  
   
 # Calcular la hora de finalización  
 $endTime = $startTime.AddSeconds($totalDurationSeconds)  
   
 # Formatear las fechas y la duración  
 $startTimeFormatted = $startTime.ToString("MMM dd, yyyy HH:mm:ss")  
 $endTimeFormatted = $endTime.ToString("MMM dd, yyyy HH:mm:ss")  
   
 #########################################################################  
 #########################################################################  
   
 $totalTestCases = 0  
 foreach ($feature in $cucumberReport) {  
 $totalTestCases += $feature.elements.Count  
 }  
   
 Write-Output "Número total de casos ejecutados: $totalTestCases"  
   
 # Configurar las credenciales de AWS como variables de entorno  
 $env:AWS\_ACCESS\_KEY\_ID = "$(AWS\_ACCESS\_KEY\_ID)"  
 $env:AWS\_SECRET\_ACCESS\_KEY = "$(AWS\_SECRET\_ACCESS\_KEY)"  
 $env:AWS\_DEFAULT\_REGION = "us-west-2" # Cambia la región si es necesario  
   
 # Validar las credenciales de AWS  
 Write-Host "Validando credenciales de AWS..."  
 aws sts get-caller-identity  
   
 # Ruta al archivo serenity.conf  
 $serenityConfigPath = "src/test/resources/serenity.conf"  
   
 # Leer el archivo serenity.conf  
 $serenityConfig = Get-Content $serenityConfigPath -Raw  
   
 # Extraer el ARN de Device Farm  
 $deviceFarmArn = ($serenityConfig -match 'deviceFarm\s\*=\s\*"([^"]+)"') | Out-Null  
 $deviceFarmArn = $matches[1]  
   
 # Validar que se haya encontrado el ARN  
 if (-not $deviceFarmArn) {  
 Write-Error "No se pudo encontrar el ARN de TestGrid en serenity.conf"  
 exit 1  
 }  
   
 Write-Host "ARN de TestGrid encontrado: $deviceFarmArn"  
   
 # Obtener las últimas sesiones del proyecto de TestGrid basadas en el número de casos ejecutados  
 $sessions = aws devicefarm list-test-grid-sessions `  
 --project-arn $deviceFarmArn `  
 --query "testGridSessions | sort\_by(@, &created) | reverse(@)[:$totalTestCases]" `  
 --output json | ConvertFrom-Json  
   
 # Validar que se hayan encontrado sesiones  
 if (-not $sessions) {  
 Write-Error "No se encontraron sesiones en el proyecto de TestGrid."  
 exit 1  
 }  
   
 # Invertir el orden de las sesiones para que el más reciente esté al final  
 $sessions = [System.Collections.ArrayList]@($sessions)  
 $sessions.Reverse()  
   
 Write-Host "Sesiones encontradas:"  
 Write-Host $sessions  
   
 # Extraer el ID del proyecto del ARN  
 $projectId = ($deviceFarmArn -split ':')[-1] -replace 'testgrid-project:', ''  
   
 # Generar enlaces de la consola de AWS  
 $videoLinksHtml = ""  
 foreach ($session in $sessions) {  
 $sessionArn = $session.arn  
 $sessionId = ($sessionArn -split '/')[-1] # Extraer el ID de la sesión del ARN  
 $region = "us-west-2" # Cambia esto si estás usando otra región  
   
 # Construir el enlace de la consola de AWS  
 $consoleLink = "[https://$region.console.aws.amazon.com/devicefarm/home#/browser/projects/$projectId/runsselenium/logs/$sessionId](#/browser/projects/$projectId/runsselenium/logs/$sessionId)"  
   
 # Agregar el enlace al HTML  
 $videoLinksHtml += "<li><a href='$consoleLink'>Video de prueba ($sessionId)</a></li>"  
 }  
  
 Write-Host "Enlaces de los videos generados:"  
 Write-Host $videoLinksHtml  
   
 ##############################################################################  
 ##############################################################################  
   
 $sessionIndex = 0  
   
 # Inicializar acumuladores para los totales  
 $totalSerenityDurationSeconds = 0  
 $totalCucumberDurationSeconds = 0  
   
 $featuresTable = "<table border='1' cellpadding='5' cellspacing='0'><tr><th>Feature</th><th>Escenario</th><th>Estado</th><th>Duración (Serenity)</th><th>Duración (cucumber)</th><th>Video</th></tr>"  
  
 # Leer el primer archivo JSON de Serenity en el directorio  
 $serenityJsonDirectory = "$(System.DefaultWorkingDirectory)/target/site/serenity"  
 $serenityJsonFile = Get-ChildItem -Path $serenityJsonDirectory -Filter \*.json | Select-Object -First 1  
   
 if (-not $serenityJsonFile) {  
 Write-Error "Error: No se encontró ningún archivo JSON en la ruta: $serenityJsonDirectory"  
 exit 1  
 }  
   
 Write-Host "Archivo JSON de Serenity seleccionado: $($serenityJsonFile.FullName)"  
 $serenityData = Get-Content -Path $serenityJsonFile.FullName -Raw -Encoding UTF8 | ConvertFrom-Json  
   
 # Validar que el archivo contenga datos válidos  
 if (-not $serenityData.testSteps) {  
 Write-Error "Error: El archivo $($serenityJsonFile.FullName) no contiene datos válidos de testSteps."  
 exit 1  
 }  
   
 # Crear un diccionario para mapear escenarios con sus duraciones en Serenity  
 $serenityDurations = @{}  
 foreach ($scenario in $serenityData.testSteps) {  
 $scenarioName = $scenario.description -replace '^\d+:\s\*', '' -replace '\s\*\(.\*\)$', '' # Limpiar el nombre del escenario  
 $scenarioDuration = [math]::Round($scenario.duration / 1000, 2) # Convertir duración de milisegundos a segundos  
 $scenarioDurationFormatted = "{0:mm\:ss}" -f ([timespan]::fromseconds($scenarioDuration))  
 $serenityDurations[$scenarioName] = $scenarioDurationFormatted  
 }  
   
 foreach ($feature in $cucumberReport) {  
 foreach ($element in $feature.elements) {  
   
 #Inicializar el estado del escenario como "passed"  
 $scenarioStatus = "passed"  
   
 $totalScenarioDuration = 0  
   
 foreach ($step in $element.steps) {  
 #Sumar la duración de cada paso  
 $totalScenarioDuration += $step.result.duration  
   
 if ($step.result.status -eq "failed") {  
 $scenarioStatus = "failed"  
 }  
 }  
   
 #Convertir la duración total de nanosegundos a un formato legible (segundos)  
 $scenarioDurationSeconds = [math]::Round($totalScenarioDuration / 1000000000, 2)  
 $scenarioDurationFormatted = "{0:mm\:ss}" -f ([timespan]::fromseconds($scenarioDurationSeconds))  
   
 # Sumar al total de Cucumber  
 $totalCucumberDurationSeconds += $scenarioDurationSeconds  
   
 # Obtener la duración de Serenity para el escenario  
 if ($serenityDurations.ContainsKey($element.name)) {  
 $serenityDuration = $serenityDurations[$element.name]  
 $serenityDurationSeconds = ([timespan]::Parse($serenityDuration)).TotalSeconds  
 $totalSerenityDurationSeconds += $serenityDurationSeconds  
 } else {  
 $serenityDuration = "No disponible"  
 }  
   
 #Obtener el enlace de video correspondiente  
 $videoLink = ""  
 if ($sessionIndex -lt $sessions.Count) {  
 $sessionArn = $sessions[$sessionIndex].arn  
 $sessionId = ($sessionArn -split '/')[-1] # Extraer el ID de la sesión del ARN  
 $region = "us-west-2" # Cambia esto si estás usando otra región  
 $videoLink = "[https://$region.console.aws.amazon.com/devicefarm/home#/browser/projects/$projectId/runsselenium/logs/$sessionId](#/browser/projects/$projectId/runsselenium/logs/$sessionId)"  
 $sessionIndex++  
 } else {  
 Write-Host "Advertencia: No hay más sesiones disponibles para asociar con los escenarios."  
 $videoLink = "No disponible"  
 }  
   
 #Agregar el escenario a la tabla con el enlace de video y duración de Serenity  
 $featuresTable += "<tr><td>$($feature.name)</td><td>$($element.name)</td><td>$scenarioStatus</td><td>$serenityDuration</td><td>$scenarioDurationFormatted</td><td><a href='$videoLink'>Ver Video</a></td></tr>"  
 }  
 }  
   
 # Formatear los totales en formato mm:ss  
 $totalSerenityDurationFormatted = "{0:mm\:ss}" -f ([timespan]::fromseconds($totalSerenityDurationSeconds))  
 $totalCucumberDurationFormatted = "{0:mm\:ss}" -f ([timespan]::fromseconds($totalCucumberDurationSeconds))  
   
 # Agregar la fila de totales al final de la tabla  
 $featuresTable += "<tr><td colspan='3'><b>Totales</b></td><td><b>$totalSerenityDurationFormatted</b></td><td><b>$totalCucumberDurationFormatted</b></td><td></td></tr>"  
   
 $featuresTable += "</table>"  
   
 ##############################################################################  
 ##############################################################################  
   
 $body = @"  
 <p>$greeting estimados,</p>  
 <p>Le informamos que se han completado las pruebas automatizadas para el siguiente flujo:</p>  
   
 <p><strong>Features y escenarios ejecutados:</strong></p>  
 $featuresTable  
   
 <p><strong>Detalles de la Ejecución:</strong></p>  
 <ul>  
 <li>Fecha y Hora de Inicio: <strong>$startTimeFormatted</strong></li>  
 <li>Fecha y Hora de Finalización: <strong>$endTimeFormatted</strong></li>  
 <li>Total de Casos: <strong>$totalTestCases</strong></li>  
 <li>Casos Passed: <strong>$totalPassed</strong></li>  
 <li>Casos Failed: <strong>$totalFailed</strong></li>  
 </ul>  
 <p>Puede descargar el reporte desde Azure DevOps haciendo clic en el siguiente enlace: <a href="$buildLink">Ver resultados</a></p>  
 <div>  
 <p>Por favor, siga estos pasos para acceder y revisar los reportes:</p>  
 <ol>  
 <li>Acceder a los archivos generados:  
 <ol>  
 <li>Hacer clic en el enlace proporcionado en este correo.</li>  
 <li>Buscar y hacer clic en "1 published" para ver los archivos generados.</li>  
 </ol>  
 </li>  
 <li>Revisar el reporte de Serenity:  
 <ol>  
 <li>Abrir la carpeta "AllReports".</li>  
 <li>Verificar que nos encontremos en la página "Artifacts".</li>  
 <li>Descargar y descomprimir <strong>serenity-reports.zip</strong>.</li>  
 <li>Abrir <strong>index.html</strong> para ver el reporte.</li>  
 </ol>  
 </li>  
 <li>Revisar el reporte de Cucumber:  
 <ol>  
 <li>Descargar y descomprimir <strong>cucumber-reports.zip</strong>.</li>  
 <li>Buscar <strong>cucumber.json</strong> para métricas.</li>  
 </ol>  
 </li>  
 </ol>  
 </div>  
 <p>Por favor, revisa los resultados y avísame si tienes alguna pregunta o necesitas alguna aclaración adicional.</p>  
 <p>Gracias por tu atención.</p>  
 <p>Saludos,</p>  
 "@  
   
 # Leer el archivo cucumber.json  
 $cucumberReportPath = "target/cucumber-reports/cucumber.json"  
 $cucumberReport = Get-Content $cucumberReportPath -Raw -Encoding UTF8 | ConvertFrom-Json  
  
 # Obtener los nombres de todos los features ejecutados  
 $featureNames = $cucumberReport | ForEach-Object { $\_.name }  
  
 # Concatenar los nombres de los features en una sola cadena separados por comas  
 $featuresList = $featureNames -join ', '  
  
 # Definir el asunto del correo con los nombres de los features  
 $subject = "Resultados de Pruebas Automatizadas - Features: $featuresList"  
  
 # Convertir la contraseña a un formato seguro  
 $securePassword = ConvertTo-SecureString $smtpPassword -AsPlainText -Force  
 $credential = New-Object System.Management.Automation.PSCredential($smtpUser, $securePassword)  
  
 # Validar conexión SMTP  
 Write-Host "Validando conexión SMTP..."  
 $smtp = New-Object Net.Mail.SmtpClient($smtpServer, $smtpPort)  
 $smtp.EnableSsl = $true  
 $smtp.Credentials = $credential  
  
 try {  
 $recipientList = $recipients -replace ';', ',' -split ','  
 # Enviar correo con el informe  
 Send-MailMessage -From $smtpUser `  
 -To $recipientList `  
 -Subject $subject `  
 -BodyAsHtml `  
 -Body $body `  
 -SmtpServer $smtpServer `  
 -Port $smtpPort `  
 -UseSsl `  
 -Credential $credential `  
 -Encoding ([System.Text.Encoding]::UTF8)  
  
 Write-Host "Correo enviado exitosamente."  
 } catch {  
 Write-Error "Error al conectar o enviar el correo: $\_"  
 }  
  
 - task: PublishBuildArtifacts@1  
 inputs:  
 pathToPublish: '$(Build.ArtifactStagingDirectory)'  
 artifactName: 'AllReports'  
 publishLocation: 'Container'  
 displayName: 'Publicar todos los informes'  
 condition: always()

DOCKER



version: '2'  
services:  
 chrome:  
 image: selenium/node-chrome:latest  
 container\_name: node-chrome  
 shm\_size: 2gb  
 depends\_on:  
 - selenium-hub  
 environment:  
 - SE\_EVENT\_BUS\_HOST=selenium-hub  
 - SE\_EVENT\_BUS\_PUBLISH\_PORT=4442  
 - SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443  
 - SE\_NODE\_MAX\_INSTANCES=5  
 - SE\_NODE\_MAX\_SESSIONS=5  
 - SE\_NODE\_SESSION\_TIMEOUT=20  
 - SE\_ENABLE\_TRACING=false  
 ports:  
 - "6901:5900"  
 - "7900:7900" *# Configura el puerto noVNC*  
  
edge:  
 image: selenium/node-edge:latest  
 container\_name: node-edge  
 shm\_size: 2gb  
 depends\_on:  
 - selenium-hub  
 environment:  
 - SE\_EVENT\_BUS\_HOST=selenium-hub  
 - SE\_EVENT\_BUS\_PUBLISH\_PORT=4442  
 - SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443  
 - SE\_ENABLE\_TRACING=false  
 ports:  
 - "6902:5900"  
 - "7901:7900" *# Configura el puerto noVNC*  
  
firefox:  
 image: selenium/node-firefox:latest  
 container\_name: node-firefox  
 shm\_size: 2gb  
 depends\_on:  
 - selenium-hub  
 environment:  
 - SE\_EVENT\_BUS\_HOST=selenium-hub  
 - SE\_EVENT\_BUS\_PUBLISH\_PORT=4442  
 - SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443  
 - SE\_ENABLE\_TRACING=false  
 ports:  
 - "6903:5900"  
 - "7902:7900" *# Configura el puerto noVNC*  
  
selenium-hub:  
 image: selenium/hub:latest  
 container\_name: selenium-hub  
 environment:  
 - SE\_ENABLE\_TRACING=false  
 - SE\_NODE\_SESSION\_TIMEOUT=60  
 ports:  
 - "4442:4442"  
 - "4443:4443"  
 - "4444:4444"