Azure DevOps: Estructurar el estándar de la Integración continua.

Un saludo, espero se encuentre bien.

Tiempo de lectura: Tres minutos (3m).

Descripción.

A continuación, detallaremos, desde **Infraestructura Digicem - DevOps**, la guía de:

- Mantener o construir un nuevo proyecto desde un repositorio, consolidado en la siguiente estructura.

> **Emojis 👇**
>
> :pipeline::**Azure Pipelines.**

Guía.

- Clone y cree los archivos, como declaración de variables requeridas.
- Establezca el ci yml, dentro de un pipeline, para su futuro uso.
- Envie cambios o PRs al repositorio público del proyecto, para que corra la canalización, y construya el artefacto, con la nueva version mergeada en main, o almacenada en alguna rama feature de vida corta.
- Valide el resultado del artefacto.
- Evidencie el release requerido por ambiente.
- Corrobore el deploy correcto del recurso en operación, sobre la nube de Azure.

Publicaciones del repositorio del proyecto.

> - **Puedes usar libremente la estructura del siguiente archivo, solo Sí compete con soluciones de la organización.**
> - ⭐**: Mandatorio.**

./pipelines/backend.json

```
{
  'projectName': '$PROJECTNAME'
}

```

/pipelines/frontend.json

```
{
  'projectName': '$Project-Name'
}

```

- ⭐**Json:** Se debe crear un archivo, en la ruta raíz de pipelines. Donde se nombre según el tipo de proyecto, que se desea correr.

- ⭐**projectName (*):** Se debe declarar el nombre del proyecto del repositorio, tal cual se creará con igual nombre en la plataforma de Sonar.

./pipelines/ci.yml

```
---
# ================================================================================================================================
# Por favor, revisa este pipeline el primer lunes de cada mes.
# Comprueba que todos los comentarios y la lógica sigan siendo relevantes.
# ================================================================================================================================
name: '$(BuildefinitionName)_$(Build.SourceBranchName)_$(Major).$(Minor).$(Patch)'

# ================================================================================================================================
# VARIABLES DE VERSIÓN.
# ================================================================================================================================
variables:
  - name: Major
    value: 1

  - name: Minor
    value: 0

  - name: Patch
    value: $[counter(format('{0}.{1}', variables['Major'], variables['Minor']), 0)]

# ================================================================================================================================
# CONFIGURACIÓN DEL AGENTE.
# ================================================================================================================================
  - name: poolName
```

```yaml
    value: 'Agent Pool Nanaykuna'

# ========================================================================================================================================
# GRUPOS DE CREDENCIALES Y SEGURIDAD.
# ========================================================================================================================================
# Variables y grupos relacionados con la seguridad y acceso a recursos.
  - group: dev-key-vault-credential
  - group: pipelines

  - name: keyVaultCredential
    value: $(dev-key-vault-credential)

# ========================================================================================================================================
# RECURSOS EXTERNOS.
# ========================================================================================================================================
resources:
  repositories:
    - repository: nanaykuna-infra
      type: git
      name: "product development/nanaykuna-infra"

# ========================================================================================================================================
# TRIGGERS.
# ========================================================================================================================================
trigger:
    batch: true
    tags:
      include:
        - '*'
    branches:
      include:
      - main
    paths:
      include:
      - src
      exclude:
      - docs
      - pipelines

# ========================================================================================================================================
# ETAPAS DEL PIPELINE.
# Cada etapa sigue una estructura similar: construcción del proyecto y configuración de variables.
# Utilizamos plantillas para configurar variables desde archivos JSON.
# La herramienta 'jq' procesa estos archivos JSON en bash y se instala durante la ejecución ya que no está preinstalada en el agente.
# ========================================================================================================================================

# ========================================================================================================================================
# ETAPA BUILD DEV.
# ========================================================================================================================================
stages:
  - stage: 'DEV'
    condition: not(contains(variables['Build.SourceBranch'], '-'))
    jobs:
      - job: Build
        pool:
          name: $(poolName)

        steps:
        # - checkout: none

        # - script: |
        #     repoUrl=$(echo $(Build.Repository.Uri) | sed 's|https://||')
        #     git clone --depth 100 https://$(git)@$repoUrl .
        #     git checkout $(Build.SourceVersion)
        #   displayName: Manual Checkout

        - template: templates/setVariablesFromJson.yml@nanaykuna-infra
          parameters:
            stageName: 'Build'

      - job: UseFunctionTemplate
        dependsOn: Build
        condition: and(succeeded('Build'), eq(dependencies.Build.outputs['checkFile_Build.projectType'], 'backend'))
        pool:
          name: $(poolName)

        steps:
        - template: 'templates/functionsBuildSteps.yml@nanaykuna-infra'
          parameters:
            environmentName: 'dev'
            keyVaultCredential: $(keyVaultCredential)
            skipTasks: $(skipTasks)
            debugMode: $(debugMode)
```

```yaml

    - job: UseGeneralTemplate
      dependsOn: Build
      condition: and(succeeded('Build'), eq(dependencies.Build.outputs['checkFile_Build.projectType'], 'frontend'))
      pool:
        name: $(poolName)

      steps:
      - template: 'templates/generalBuildSteps.yml@nanaykuna-infra'
        parameters:
          environmentName: 'dev'
          skipTasks: $(skipTasks)
          debugMode: $(debugMode)

# ================================================================================================================================
# ETAPAS BUILD STA.
# ================================================================================================================================
  - stage: 'STA'
    condition: contains(variables['Build.SourceBranch'], '-rc')
    jobs:
      - job: Build
        pool:
          name: $(poolName)

        steps:
        - template: templates/setVariablesFromJson.yml@nanaykuna-infra
          parameters:
            stageName: 'Build'

      - job: UseFunctionTemplate
        dependsOn: Build
        condition: and(succeeded('Build'), eq(dependencies.Build.outputs['checkFile_Build.projectType'], 'backend'))
        pool:
          name: $(poolName)

        steps:
        - template: 'templates/functionsBuildSteps.yml@nanaykuna-infra'
          parameters:
            environmentName: 'sta'
            keyVaultCredential: $(keyVaultCredential)
            skipTasks: $(skipTasks)
            debugMode: $(debugMode)

      - job: UseGeneralTemplate
        dependsOn: Build
        condition: and(succeeded('Build'), eq(dependencies.Build.outputs['checkFile_Build.projectType'], 'frontend'))

        pool:
          name: $(poolName)

        steps:
        - template: 'templates/generalBuildSteps.yml@nanaykuna-infra'
          parameters:
             environmentName: 'sta'
             skipTasks: $(skipTasks)
             debugMode: $(debugMode)

# ================================================================================================================================
# ETAPAS BUILD PROD.
# ================================================================================================================================
  - stage: 'PROD'
    condition: contains(variables['Build.SourceBranch'], '-ga')
    jobs:
      - job: Build
        pool:
          name: $(poolName)

        steps:
        - template: templates/setVariablesFromJson.yml@nanaykuna-infra
          parameters:
            stageName: 'Build'

      - job: UseFunctionTemplate
        dependsOn: Build
        condition: and(succeeded('Build'), eq(dependencies.Build.outputs['checkFile_Build.projectType'], 'backend'))
        pool:
          name: $(poolName)

        steps:
        - template: 'templates/functionsBuildSteps.yml@nanaykuna-infra'
          parameters:
            environmentName: 'prd'
            keyVaultCredential: $(keyVaultCredential)
```

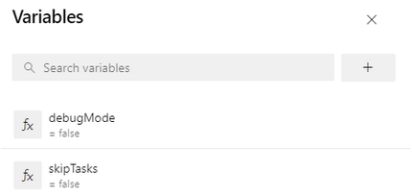
```yaml
            skipTasks: $(skipTasks)
            debugMode: $(debugMode)

      - job: UseGeneralTemplate
        dependsOn: Build
        condition: and(succeeded('Build'), eq(dependencies.Build.outputs['checkFile_Build.projectType'], 'frontend'))

        pool:
          name: $(poolName)

        steps:
        - template: 'templates/generalBuildSteps.yml@nanaykuna-infra'
          parameters:
            environmentName: 'prd'
            skipTasks: $(skipTasks)
            debugMode: $(debugMode)

```



Tokens del pipeline para saltar tasks de sonar, snyk. Como debugger al Código.

- ⭐ **ci yml:** Se debe crear un archivo, en la ruta raíz de pipelines. Donde se clone tal cual el contenido del pipeline. El cual se utiliza como el pipeline estándar. Donde se usará los templates correspondientes, bajo el archivo json presentado. Con ello concluir el paso y build de la publicación del artefacto, según proyecto requerido.

Publicaciones de los templates.

- **Puedes usar libremente la estructura del siguiente archivo, solo Sí compete con soluciones de la organización.**

./templates/setVariablesFromJson.yml

```yaml
# setVariablesFromJson.yml

parameters:
  jsonPath: './pipelines/*.json'
  stageName: ''

steps:
  - script: |
      if [ -f ./pipelines/backend.json ]; then
          echo "##vso[task.setvariable variable=projectType;isOutput=true]backend"
          echo "##vso[task.setvariable variable=jsonFilePath]./pipelines/backend.json"
      elif [ -f ./pipelines/frontend.json ]; then
          echo "##vso[task.setvariable variable=projectType;isOutput=true]frontend"
          echo "##vso[task.setvariable variable=jsonFilePath]./pipelines/frontend.json"
      else
          echo "ERROR: No se encontró ni backend.json ni frontend.json."
          exit 1
      fi
    displayName: "Check and Set Variables"
    name: checkFile_${{ parameters.stageName }}

```

./templates/functionsBuildSteps.yml

```yaml
---
# Este template fue revisado por última vez en [09/11/2023]. Es recomendable revisarlo periódicamente para asegurar su óptimo funcionamiento.
# Realiza el proceso de construcción, prueba y despliegue para cada proyecto BackEnd.

parameters:
  - name: skipTasks
    type: string
    default: ''

  - name: debugMode
    default: ''

  - name: environmentName
    type: string
```

```yaml
    default: ''

  - name: keyVaultCredential
    type: string
    default: ''

  - name: jsonPath
    default: './pipelines/backend.json'

steps:

  # ======================================
  # DEPURACIÓN.
  # ======================================

  - script: |
        # Primero, verificamos el valor de debugMode
        echo "debugMode value: ${{ parameters.debugMode }}"

        if [ "${{ parameters.debugMode }}" != "true" ] && [ "${{ parameters.debugMode }}" != "false" ]; then
            echo "ERROR: El valor de debugMode es inválido. Debe ser 'true' o 'false'."
            debugMode="false"
        fi

        if [ "${{ parameters.debugMode }}" == "true" ]; then
            echo "Starting Debug Validations..."

            if [ -z $environmentName ]; then
                echo "ERROR: environmentName is empty."
                exit 1
            fi

            echo "Finished Debug Validations."
        fi
    displayName: 'T00 - Debug Validations'

  - script: |
      echo "Configuración del Pipeline:"

      if [ -f "$(SLNPath)" ]; then
          echo "El archivo de solución .sln en la ruta $(SLNPath) existe."
      else
          echo "ERROR: El archivo de solución .sln en la ruta $(SLNPath) no existe."
          exit 1
      fi

      env | grep -i "Path" >> registro.log 2>&1
      grep 'ERROR' registro.log >> errores.log
      grep -v 'DEBUG' registro.log >> logs_limpio.log
    displayName: 'T01 - Operaciones basadas en archivo .sln'
    condition: and(succeeded(), eq(variables['debugMode'], 'true'))

  - task: PublishBuildArtifacts@1
    inputs:
      PathtoPublish: 'registro.log'
      ArtifactName: 'logs'
    displayName: 'T02 - Publicar logs'
    condition: eq(variables['debugMode'], 'true')

  # ======================================
  # VARIABLES.
  # ======================================

  - script: |
      sudo apt-get update >> /dev/null && sudo apt-get install -y jq >> /dev/null
      legacyStatus=$(jq -r '.Legacy' "${{ parameters.jsonPath }}") >> /dev/null
      projectName=$(jq -r '.projectName' "${{ parameters.jsonPath }}") >> /dev/null
      projectNameLower=$(echo $projectName | tr '[:upper:]' '[:lower:]' | sed 's/s\./g' | sed 's/s$//g' | tr '.' '-' )
      echo $projectNameLower
      echo "##vso[task.setvariable variable=environmentName]${{ parameters.environmentName }}"
      echo "##vso[task.setvariable variable=PROJPath]nanaykuna-$projectNameLower"
      echo "##vso[task.setvariable variable=KPROJPath]nanaykuna_nanaykuna-$projectNameLower"
      clear

      if [ "$projectName" == "Db.Migrations.Tools" ]; then
          echo "Inside Db.Migrations.Tools block..."
          #echo "##vso[task.setvariable variable=PROJPath]nanaykuna-db-migrations-tools"
          #echo "##vso[task.setvariable variable=KPROJPath]nanaykuna_nanaykuna-db-migrations-tools"
          #echo "##vso[task.setvariable variable=SRCPath]src/Nanaykuna.$projectName.App"

      elif [ "$projectName" == "CustomerLoyalty" ]; then
          echo "Inside CustomerLoyalty block..."
          echo "##vso[task.setvariable variable=PROJPath]nanaykuna-customer-loyalty"
```

```
        echo "##vso[task.setvariable variable=KPROJPath]nanaykuna_nanaykuna-customer-loyalty"

    elif [ "$projectName" == "TicketManager" ]; then
        echo "Inside Integration block..."
        echo "##vso[task.setvariable variable=PROJPath]nanaykuna-ticket-manager"
        echo "##vso[task.setvariable variable=KPROJPath]nanaykuna_nanaykuna-ticket-manager"

    elif [ "$projectName" == "Bff" ]; then
        echo "Inside Integration block..."
        echo "##vso[task.setvariable variable=PROJPath]nanaykuna-bff-integration"
        echo "##vso[task.setvariable variable=KPROJPath]nanaykuna_nanaykuna-bff-integration"

    elif [[ "$projectName" == "BackOffice.Api" ]]; then
        echo "Entrando al bloque Back Office Api..."
        echo "##vso[task.setvariable variable=APPSETTINGS]skip"
        echo "##vso[task.setvariable variable=KPROJPath]nanaykuna_nanaykuna-backoffice-api"
        echo "##vso[task.setvariable variable=PROJPath]nanaykuna-backoffice-api"
        echo "...Estableciendo un appsetting Vacia."

    else
        echo "Entrando al bloque No-Legacy..."
        #echo "##vso[task.setvariable variable=PROJPath]'nanaykuna-"$projectName"-functions'"
        #echo "##vso[task.setvariable variable=KPROJPath]nanaykuna_nanaykuna-"$projectName"-functions"
        #echo "##vso[task.setvariable variable=SRCPath]src/Nanaykuna."$projectName".Functions.Presentation"
    fi

    if [[ "$projectName" == "Back.Office" ]]; then
        echo "Entrando al bloque Back Office..."
        echo "##vso[task.setvariable variable=PROJPath]nanaykuna-back-office-functions"
        echo "##vso[task.setvariable variable=KPROJPath]nanaykuna_nanaykuna-back-office-functions"
    fi

    archivos=($(find ./src/* -type f -name "*settings*.json"))
    archivo_dev=""
    archivo_sta=""
    archivo_prd=""
    archivo_json=""

    for archivo in "${archivos[@]}"; do
      if [[ $archivo == *dev* ]]; then
        archivo_dev="$archivo"
      elif [[ $archivo == *sta* ]]; then
        archivo_sta="$archivo"
      elif [[ $archivo == *prd* ]]; then
        archivo_prd="$archivo"
      elif echo "$archivo" | grep -q -e "local.settings.json\|appsettings.json"; then
        archivo_json="$archivo"
      fi
    done

    echo $archivo_dev

    if [ -n "$archivo_dev" ]; then
      echo "##vso[task.setvariable variable=ArchivoDev]$archivo_dev"
    fi
    if [ -n "$archivo_sta" ]; then
      echo "##vso[task.setvariable variable=ArchivoSTA]$archivo_sta"
    fi
    if [ -n "$archivo_prd" ]; then
      echo "##vso[task.setvariable variable=ArchivoPRD]$archivo_prd"
    fi
    if [ -n "$archivo_json" ]; then
      echo "##vso[task.setvariable variable=ArchivoJSON]$archivo_json"
    else
      echo "##vso[task.setvariable variable=ArchivoJSON]skip"
    fi

    sln_files=$(find ./src -type f -name "*.sln" -exec echo {} \;)

    sln_paths=""
    for sln_file in $sln_files; do
        sln_paths="$sln_file"
    done

    csproj_file=$(find ./src -type f -name "*.csproj" -exec echo {} \;)

    csproj_paths=""
    for csproj_file in $csproj_file; do
        if [[ "$csproj_file" == *"ts.csproj" ]]; then
            continue
        fi
        csproj_paths="$csproj_paths $csproj_file"
    done
```

```
      csproj=$(basename "$csproj_paths")

      echo "##vso[task.setvariable variable=CsprojPaths]$csproj"
      echo "##vso[task.setvariable variable=SlnPaths]$sln_paths"
    displayName: 'T03 - Set variable exists'
    timeoutInMinutes: 1

  # ========================================
  # INSTALLATIONS.
  # ========================================

  - template: './installSOPS.yml'

  # Usamos la herramienta, en lugar de Otras porque: maneja mejor los errores de tipo Cifrado.
  - template: './decryptSOPS.yml'
    parameters:
      DecryptedFileWithPath: $(ArchivoJSON)
      FileToDecryptFileWithPath2: $(ArchivoSTA)
      FileToDecryptFileWithPath3: $(ArchivoPRD)
      FileToDecryptFileWithPath: $(ArchivoDev)
      shouldRun: $[and(ne(variables['ArchivoJSON'], 'skip'), eq(variables['environmentName'], 'dev'))]

  # Compila el proyecto con las configuraciones de Sonar.
  - template: './installJava.yml@nanaykuna-infra'

  # ========================================
  # PRUEBAS Y CONSTRUCCIÓN.
  # ========================================

  - template: './dotnetBuild.yml'
    parameters:
      skipTasks: $(skipTasks)
      PublishProject: $(CsprojPaths)
      SonarProjectKey: $(KPROJPath)
      SonarProjectName: $(PROJPath)

  # ========================================
  # DESPLIEGUE.
  # ========================================

  - template: ./sonar/sonar.yml
    parameters:
      skipTasks: '$(skipTasks)'

  - task: DotNetCoreCLI@2
    condition: and(succeeded(), eq(variables['SKIPTASKS'], 'false'), ne(variables['Build.Reason'], 'PullRequest'))
    inputs:
      command: publish
      projects: "src/**/$(CsprojPaths)"
      publishWebProjects: false
      arguments: "--configuration Release -v q --output $(build.artifactstagingdirectory)"
      zipAfterPublish: True
    displayName: "T14 - Publish locally"

  - task: PublishBuildArtifacts@1
    condition: and(succeeded(), ne(variables['Build.Reason'], 'PullRequest'))
    inputs:
      PathtoPublish: "$(build.artifactstagingdirectory)"
    displayName: "T15 - Publish to Azure DevOps Artifacts"

  - template: "./snykTest.yml"
    parameters:
      SolutionPath: $(SlnPaths)
      skipTasks: ${{ parameters.skipTasks }}
```

./templates/generalBuildSteps.yml

```
---
# Este template fue revisado por última vez en [27/10/2023]. Es recomendable revisarlo periódicamente para asegurar su óptimo funcionamiento.
# Realiza el proceso de construcción, prueba y despliegue para cada proyecto FrontEnd.

parameters:

  - name: skipTasks
    type: string
    default: ''

  - name: debugMode
    default: ''
```

```yaml
  - name: environmentName
    type: string
    default: ''

  - name: keyVaultCredential
    type: string
    default: ''

  - name: jsonPath
    default: './pipelines/frontend.json'

steps:

  # ======================================
  # Variables.
  # ======================================

  - script: |
      git config --global user.email 'dev.tools@progresol.com'
      git config --global user.name 'Dev Tool Progresol'

      sudo apt-get update >> /dev/null && sudo apt-get install -y jq >> /dev/null
      legacyStatus=$(jq -r '.Legacy' "${{ parameters.jsonPath }}") >> /dev/null
      projectName=$(jq -r '.projectName' "${{ parameters.jsonPath }}") >> /dev/null

      echo "##vso[task.setvariable variable=PROJPath]nanaykuna-$projectName"
      echo "##vso[task.setvariable variable=KPROJPath]nanaykuna_nanaykuna-$projectName"
    displayName: 'Set variable exists'
    timeoutInMinutes: 1

  # ======================================
  # DEPURACIÓN.
  # ======================================

  - script: |
        # Primero, verificamos el valor de debugMode
        echo "debugMode value: ${{ parameters.debugMode }}"

        if [ "${{ parameters.debugMode }}" != "true" ] && [ "${{ parameters.debugMode }}" != "false" ]; then
            echo "ERROR: El valor de debugMode es inválido. Debe ser 'true' o 'false'."
            debugMode="false"
        fi

        if [ "${{ parameters.debugMode }}" == "true" ]; then
            echo "Starting Debug Validations..."

            if [ -z $environmentName ]; then
                echo "ERROR: environmentName is empty."
                exit 1
            fi

            echo "Finished Debug Validations."
        fi
    displayName: 'Debug Validations'

  - script: |
      echo "Configuración del Pipeline:"

      if [ -f "$(SLNPath)" ]; then
          echo "El archivo de solución .sln en la ruta $(SLNPath) existe."
      else
          echo "ERROR: El archivo de solución .sln en la ruta $(SLNPath) no existe."
          exit 1
      fi

      env | grep -i "Path" >> registro.log 2>&1
      grep 'ERROR' registro.log >> errores.log
      grep -v 'DEBUG' registro.log >> logs_limpio.log
    displayName: 'Operaciones basadas en archivo .sln'
    condition: and(succeeded(), eq(variables['debugMode'], 'true'))

  - task: PublishBuildArtifacts@1
    inputs:
      PathtoPublish: 'registro.log'
      ArtifactName: 'logs'
    displayName: 'Publicar logs'
    condition: eq(variables['debugMode'], 'true')

  # ======================================
  # INSTALLATIONS.
  # ======================================

  - template: ./installSOPS.yml
```

```yaml

- template: ./decryptSOPS.yml
  parameters:
    DecryptedFileWithPath: $(System.DefaultWorkingDirectory)/.env.development
    FileToDecryptFileWithPath2: $(System.DefaultWorkingDirectory)/env.enc.sta
    FileToDecryptFileWithPath3: $(System.DefaultWorkingDirectory)/env.enc.prd
    FileToDecryptFileWithPath: $(System.DefaultWorkingDirectory)/env.enc.${{ parameters.environmentName }}

- template: installzip.yml@nanaykuna-infra
  parameters:
    message: 00 - Install Zip

# Compila el proyecto con las configuraciones de Sonar.
- template: installJava.yml@nanaykuna-infra

- task: NodeTool@0
  inputs:
    versionSpec: 16.16.0
  displayName: 01 - Install Node.js

- task: SonarCloudPrepare@1
  inputs:
    SonarCloud: Sonar Cloud
    organization: nanaykuna
    scannerMode: CLI
    configMode: manual
    cliProjectKey: '$(KPROJPath)'
    cliProjectName: '$(PROJPath)'
    cliSources: .
    extraProperties: sonar.javascript.lcov.reportPaths=./coverage/lcov.info
  displayName: 02 - Sonar Cloud Prepare

- script: |
    npm install -g yarn
    yarn -v
    yarn install
  continueOnError: true
  displayName: 03 - Yarn Install

- script: |
    yarn run standard-version
     current_message=$(git show -s --format=%s)
    echo $current_message
    commit_message="${current_message} [ci skip]"
    echo $commit_message
    git commit --amend -m --no-verify "$commit_message"
    git push --follow-tags --no-verify origin HEAD:$(Build.SourceBranchName)
  continueOnError: true
  displayName: 04 - Run Version

# ======================================
# PRUEBAS.
# ======================================

- script: |
    yarn test --coverage
  continueOnError: true
  displayName: 05 - Testing

- task: PublishCodeCoverageResults@1
  inputs:
    codeCoverageTool: Cobertura
    summaryFileLocation: $(System.DefaultWorkingDirectory)/coverage/cobertura-coverage.xml
  displayName: 06 - Publish Tests PublishCodeCoverageResults

- task: PublishTestResults@2
  inputs:
    testResultsFormat: JUnit
    testResultsFiles: "**/*unit.xml"
    mergeTestResults: true
  displayName: 07 - Publish Tests Results

- task: SonarCloudAnalyze@1
  continueOnError: false
  displayName: 08 - Sonar Cloud Analyze

- template: ./sonar/sonar.yml
  parameters:
    skipTasks: '$(skipTasks)'

- task: SnykSecurityScan@1
  inputs:
    serviceConnectionEndpoint: snyk
```

```
      testType: app
      monitorWhen: always
      failOnIssues: false
    continueOnError: true
    displayName: 09 - Security Scan

  # ======================================
  # CONSTRUCCIÓN.
  # ======================================

  - script: |
      yarn run build:dev
    displayName: 10 - Build
    continueOnError: false

  # ======================================
  # DESPLIEGUE.
  # ======================================

  - task: ArchiveFiles@2
    inputs:
      rootFolderOrFile: .
      includeRootFolder: false
      archiveType: zip
      archiveFile: $(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip
      replaceExistingArchive: true
    displayName: 11 - Publish locally

  - task: PublishBuildArtifacts@1
    inputs:
      PathtoPublish: $(Build.ArtifactStagingDirectory)
      ArtifactName: drop
    displayName: 12 - Publish to Azure DevOps Artifacts

```

💪Ahora se puede dimensionar el nivel de ejecución, como tiempo a la hora de gestionar este requerimiento.

¡Mil gracias por la atención prestada!

Cualquier duda me puedes contactar…

:WhatsApp: ***+573058288031,***

como mi usuario :slack:.