

# 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.
- Envíe 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 🔗

```
1 {
2   'projectName': '$PROJECTNAME'
3 }
4
```

### ./pipelines/frontend.json 🔗

```
1 {
2   'projectName': '$Project-Name'
3 }
4
```

- 🌟 **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 🔗

```
1 ---
2 # =====
3 # Por favor, revisa este pipeline el primer lunes de cada mes.
4 # Comprueba que todos los comentarios y la lógica sigan siendo relevantes.
5 # =====
6 name: '$(BuildDefinitionName)_$(Build.SourceBranchName)_$(Major).$(Minor).$(Patch)'
7
8 # =====
9 # VARIABLES DE VERSIÓN.
10 # =====
11 variables:
12   - name: Major
13     value: 1
14
15   - name: Minor
16     value: 0
17
18   - name: Patch
19     value: ${counter(format('{0}.{1}', variables['Major'], variables['Minor']), 0)}
20
21 # =====
22 # CONFIGURACIÓN DEL AGENTE.
23 # =====
24   - name: poolName
```

```

25     value: 'Agent Pool Nanaykuna'
26
27 # =====
28 # GRUPOS DE CREDENCIALES Y SEGURIDAD.
29 # =====
30 # Variables y grupos relacionados con la seguridad y acceso a recursos.
31 - group: dev-key-vault-credential
32 - group: pipelines
33
34 - name: keyVaultCredential
35   value: $(dev-key-vault-credential)
36
37 # =====
38 # RECURSOS EXTERNOS.
39 # =====
40 resources:
41   repositories:
42     - repository: nanaykuna-infra
43       type: git
44       name: "product development/nanaykuna-infra"
45
46 # =====
47 # TRIGGERS.
48 # =====
49 trigger:
50   batch: true
51   tags:
52     include:
53       - '*'
54   branches:
55     include:
56       - main
57   paths:
58     include:
59       - src
60     exclude:
61       - docs
62       - pipelines
63
64 # =====
65 # ETAPAS DEL PIPELINE.
66 # Cada etapa sigue una estructura similar: construcción del proyecto y configuración de variables.
67 # Utilizamos plantillas para configurar variables desde archivos JSON.
68 # La herramienta 'jq' procesa estos archivos JSON en bash y se instala durante la ejecución ya que no está preinstalada en el agente.
69 # =====
70
71 # =====
72 # ETAPA BUILD DEV.
73 # =====
74 stages:
75 - stage: 'DEV'
76   condition: not(contains(variables['Build.SourceBranch'], '-'))
77   jobs:
78   - job: Build
79     pool:
80       name: $(poolName)
81
82     steps:
83     # - checkout: none
84
85     # - script: |
86     #   repoUrl=$(echo $(Build.Repository.Uri) | sed 's|https://|'|)
87     #   git clone --depth 100 https://$(git)@repoUrl .
88     #   git checkout $(Build.SourceVersion)
89     #   displayName: Manual Checkout
90
91   - template: templates/setVariablesFromJson.yml@nanaykuna-infra
92     parameters:
93       stageName: 'Build'
94
95   - job: UseFunctionTemplate
96     dependsOn: Build
97     condition: and(succeeded('Build'), eq(dependencies.Build.outputs['checkFile_Build.projectType'], 'backend'))
98     pool:
99       name: $(poolName)
100
101     steps:
102     - template: 'templates/functionsBuildSteps.yml@nanaykuna-infra'
103       parameters:
104         environmentName: 'dev'
105         keyVaultCredential: $(keyVaultCredential)
106         skipTasks: $(skipTasks)
107         debugMode: $(debugMode)

```

```

108
109   - job: UseGeneralTemplate
110     dependsOn: Build
111     condition: and(succeeded('Build'), eq(dependencies.Build.outputs['checkFile_Build.projectType'], 'frontend'))
112     pool:
113       name: $(poolName)
114
115     steps:
116     - template: 'templates/generalBuildSteps.yml@nanaykuna-infra'
117       parameters:
118         environmentName: 'dev'
119         skipTasks: $(skipTasks)
120         debugMode: $(debugMode)
121
122 # =====
123 # ETAPAS BUILD STA.
124 # =====
125 - stage: 'STA'
126   condition: contains(variables['Build.SourceBranch'], '-rc')
127   jobs:
128   - job: Build
129     pool:
130       name: $(poolName)
131
132     steps:
133     - template: templates/setVariablesFromJson.yml@nanaykuna-infra
134       parameters:
135         stageName: 'Build'
136
137   - job: UseFunctionTemplate
138     dependsOn: Build
139     condition: and(succeeded('Build'), eq(dependencies.Build.outputs['checkFile_Build.projectType'], 'backend'))
140     pool:
141       name: $(poolName)
142
143     steps:
144     - template: 'templates/functionsBuildSteps.yml@nanaykuna-infra'
145       parameters:
146         environmentName: 'sta'
147         keyVaultCredential: $(keyVaultCredential)
148         skipTasks: $(skipTasks)
149         debugMode: $(debugMode)
150
151   - job: UseGeneralTemplate
152     dependsOn: Build
153     condition: and(succeeded('Build'), eq(dependencies.Build.outputs['checkFile_Build.projectType'], 'frontend'))
154
155     pool:
156       name: $(poolName)
157
158     steps:
159     - template: 'templates/generalBuildSteps.yml@nanaykuna-infra'
160       parameters:
161         environmentName: 'sta'
162         skipTasks: $(skipTasks)
163         debugMode: $(debugMode)
164
165 # =====
166 # ETAPAS BUILD PROD.
167 # =====
168 - stage: 'PROD'
169   condition: contains(variables['Build.SourceBranch'], '-ga')
170   jobs:
171   - job: Build
172     pool:
173       name: $(poolName)
174
175     steps:
176     - template: templates/setVariablesFromJson.yml@nanaykuna-infra
177       parameters:
178         stageName: 'Build'
179
180   - job: UseFunctionTemplate
181     dependsOn: Build
182     condition: and(succeeded('Build'), eq(dependencies.Build.outputs['checkFile_Build.projectType'], 'backend'))
183     pool:
184       name: $(poolName)
185
186     steps:
187     - template: 'templates/functionsBuildSteps.yml@nanaykuna-infra'
188       parameters:
189         environmentName: 'prd'
190         keyVaultCredential: $(keyVaultCredential)

```

```

191     skipTasks: ${skipTasks}
192     debugMode: ${debugMode}
193
194   - job: UseGeneralTemplate
195     dependsOn: Build
196     condition: and(succeeded('Build'), eq(dependencies.Build.outputs['checkFile_Build.projectType'], 'frontend'))
197
198     pool:
199       name: ${poolName}
200
201     steps:
202     - template: 'templates/generalBuildSteps.yml@nanaykuna-infra'
203       parameters:
204         environmentName: 'prd'
205         skipTasks: ${skipTasks}
206         debugMode: ${debugMode}
207

```

## Variables



Search variables



debugMode  
= false

skipTasks  
= false

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](#) [🔗](#)

```

1  # setVariablesFromJson.yml
2
3  parameters:
4    jsonPath: './pipelines/*.json'
5    stageName: ''
6
7  steps:
8    - script: |
9      if [ -f ./pipelines/backend.json ]; then
10        echo "##vso[task.setvariable variable=projectType;isOutput=true]backend"
11        echo "##vso[task.setvariable variable=jsonFilePath]./pipelines/backend.json"
12      elif [ -f ./pipelines/frontend.json ]; then
13        echo "##vso[task.setvariable variable=projectType;isOutput=true]frontend"
14        echo "##vso[task.setvariable variable=jsonFilePath]./pipelines/frontend.json"
15      else
16        echo "ERROR: No se encontró ni backend.json ni frontend.json."
17        exit 1
18      fi
19    displayName: "Check and Set Variables"
20    name: checkFile_${ parameters.stageName }
21

```

### [./templates/functionsBuildSteps.yml](#)

```

1  ---
2  # Este template fue revisado por última vez en [09/11/2023]. Es recomendable revisarlo periódicamente para asegurar su óptimo funcionamiento.
3  # Realiza el proceso de construcción, prueba y despliegue para cada proyecto BackEnd.
4
5  parameters:
6    - name: skipTasks
7      type: string
8      default: ''
9
10   - name: debugMode
11     type: string
12     default: ''
13
14   - name: environmentName
15     type: string

```

```

15     default: ''
16
17     - name: keyVaultCredential
18       type: string
19       default: ''
20
21     - name: jsonPath
22       default: './pipelines/backend.json'
23
24 steps:
25
26     # =====
27     # DEPURACIÓN.
28     # =====
29
30     - script: |
31         # Primero, verificamos el valor de debugMode
32         echo "debugMode value: ${ parameters.debugMode }"
33
34         if [ "${ parameters.debugMode }" != "true" ] && [ "${ parameters.debugMode }" != "false" ]; then
35             echo "ERROR: El valor de debugMode es inválido. Debe ser 'true' o 'false'."
36             debugMode="false"
37         fi
38
39         if [ "${ parameters.debugMode }" == "true" ]; then
40             echo "Starting Debug Validations..."
41
42             if [ -z $environmentName ]; then
43                 echo "ERROR: environmentName is empty."
44                 exit 1
45             fi
46
47             echo "Finished Debug Validations."
48         fi
49     displayName: 'T00 - Debug Validations'
50
51     - script: |
52         echo "Configuración del Pipeline:"
53
54         if [ -f "$(SLNPath)" ]; then
55             echo "El archivo de solución .sln en la ruta $(SLNPath) existe."
56         else
57             echo "ERROR: El archivo de solución .sln en la ruta $(SLNPath) no existe."
58             exit 1
59         fi
60
61         env | grep -i "Path" >> registro.log 2>&1
62         grep 'ERROR' registro.log >> errores.log
63         grep -v 'DEBUG' registro.log >> logs_limpio.log
64     displayName: 'T01 - Operaciones basadas en archivo .sln'
65     condition: and(succeeded(), eq(variables['debugMode'], 'true'))
66
67     - task: PublishBuildArtifacts@1
68       inputs:
69         PathToPublish: 'registro.log'
70         ArtifactName: 'logs'
71       displayName: 'T02 - Publicar logs'
72       condition: eq(variables['debugMode'], 'true')
73
74     # =====
75     # VARIABLES.
76     # =====
77
78     - script: |
79         sudo apt-get update >> /dev/null && sudo apt-get install -y jq >> /dev/null
80         legacyStatus=$(jq -r '.Legacy' "${ parameters.jsonPath }") >> /dev/null
81         projectName=$(jq -r '.projectName' "${ parameters.jsonPath }") >> /dev/null
82         projectNameLower=$(echo $projectName | tr '[:upper:]' '[:lower:]' | sed 's/s\././g' | sed 's/s$/g' | tr '.' '-' )
83         echo $projectNameLower
84         echo "##vso[task.setvariable variable=environmentName]${ parameters.environmentName }"
85         echo "##vso[task.setvariable variable=PROJPath]nanaykuna-$projectNameLower"
86         echo "##vso[task.setvariable variable=KPROJPath]nanaykuna_nanaykuna-$projectNameLower"
87         clear
88
89         if [ "$projectName" == "Db.Migrations.Tools" ]; then
90             echo "Inside Db.Migrations.Tools block..."
91             echo "##vso[task.setvariable variable=PROJPath]nanaykuna-db-migrations-tools"
92             echo "##vso[task.setvariable variable=KPROJPath]nanaykuna_nanaykuna-db-migrations-tools"
93             echo "##vso[task.setvariable variable=SRCPath]src/Nanaykuna.$projectName.App"
94
95         elif [ "$projectName" == "CustomerLoyalty" ]; then
96             echo "Inside CustomerLoyalty block..."
97             echo "##vso[task.setvariable variable=PROJPath]nanaykuna-customer-loyalty"

```

```

98     echo "##vso[task.setvariable variable=KPROJPath]nanaykuna_nanaykuna-customer-loyalty"
99
100 elif [ "$ProjectName" == "TicketManager" ]; then
101     echo "Inside Integration block..."
102     echo "##vso[task.setvariable variable=PROJPath]nanaykuna-ticket-manager"
103     echo "##vso[task.setvariable variable=KPROJPath]nanaykuna_nanaykuna-ticket-manager"
104
105 elif [ "$ProjectName" == "Bff" ]; then
106     echo "Inside Integration block..."
107     echo "##vso[task.setvariable variable=PROJPath]nanaykuna-bff-integration"
108     echo "##vso[task.setvariable variable=KPROJPath]nanaykuna_nanaykuna-bff-integration"
109
110 elif [ "$ProjectName" == "BackOffice.Api" ]; then
111     echo "Entrando al bloque Back Office Api..."
112     echo "##vso[task.setvariable variable=APPSSETTINGS]skip"
113     echo "##vso[task.setvariable variable=KPROJPath]nanaykuna_nanaykuna-backoffice-api"
114     echo "##vso[task.setvariable variable=PROJPath]nanaykuna-backoffice-api"
115     echo "...Estableciendo un appsetting Vacía."
116
117 else
118     echo "Entrando al bloque No-Legacy..."
119     #echo "##vso[task.setvariable variable=PROJPath]'nanaykuna-'$ProjectName'-functions'"
120     #echo "##vso[task.setvariable variable=KPROJPath]nanaykuna_nanaykuna-'$ProjectName'-functions"
121     #echo "##vso[task.setvariable variable=SRCPath]src/Nanaykuna.'$ProjectName'.Functions.Presentation"
122 fi
123
124 if [ "$ProjectName" == "Back.Office" ]; then
125     echo "Entrando al bloque Back Office..."
126     echo "##vso[task.setvariable variable=PROJPath]nanaykuna-back-office-functions"
127     echo "##vso[task.setvariable variable=KPROJPath]nanaykuna_nanaykuna-back-office-functions"
128 fi
129
130 archivos=$(find ./src/* -type f -name "*settings*.json")
131 archivo_dev=""
132 archivo_sta=""
133 archivo_prd=""
134 archivo_json=""
135
136 for archivo in "${archivos[@]}; do
137     if [ "$archivo" == *dev* ]; then
138         archivo_dev="$archivo"
139     elif [ "$archivo" == *sta* ]; then
140         archivo_sta="$archivo"
141     elif [ "$archivo" == *prd* ]; then
142         archivo_prd="$archivo"
143     elif echo "$archivo" | grep -q -e "local.settings.json\|appsettings.json"; then
144         archivo_json="$archivo"
145     fi
146 done
147
148 echo $archivo_dev
149
150 if [ -n "$archivo_dev" ]; then
151     echo "##vso[task.setvariable variable=ArchivoDev]$archivo_dev"
152 fi
153 if [ -n "$archivo_sta" ]; then
154     echo "##vso[task.setvariable variable=ArchivoSTA]$archivo_sta"
155 fi
156 if [ -n "$archivo_prd" ]; then
157     echo "##vso[task.setvariable variable=ArchivoPRD]$archivo_prd"
158 fi
159 if [ -n "$archivo_json" ]; then
160     echo "##vso[task.setvariable variable=ArchivoJSON]$archivo_json"
161 else
162     echo "##vso[task.setvariable variable=ArchivoJSON]skip"
163 fi
164
165 sln_files=$(find ./src -type f -name "*.sln" -exec echo {} \;)
166
167 sln_paths=""
168 for sln_file in $sln_files; do
169     sln_paths="$sln_file"
170 done
171
172 csproj_file=$(find ./src -type f -name "*.csproj" -exec echo {} \;)
173
174 csproj_paths=""
175 for csproj_file in $csproj_file; do
176     if [ "$csproj_file" == *.ts.csproj ]; then
177         continue
178     fi
179     csproj_paths="$csproj_paths $csproj_file"
180 done

```

```

181
182     csproj=$(basename "$csproj_paths")
183
184     echo "##vso[task.setvariable variable=CsprojPaths]$csproj"
185     echo "##vso[task.setvariable variable=SlnPaths]$sln_paths"
186     displayName: 'T03 - Set variable exists'
187     timeoutInMinutes: 1
188
189     # =====
190     # INSTALLATIONS.
191     # =====
192
193     - template: './installSOPS.yml'
194
195     # Usamos la herramienta, en lugar de Otras porque: maneja mejor los errores de tipo Cifrado.
196     - template: './decryptSOPS.yml'
197     parameters:
198         DecryptedFilePath: $(ArchivoJSON)
199         FileToDecryptFilePath2: $(ArchivoSTA)
200         FileToDecryptFilePath3: $(ArchivoPRD)
201         FileToDecryptFilePath: $(ArchivoDev)
202         shouldRun: $[and(ne(variables['ArchivoJSON'], 'skip'), eq(variables['environmentName'], 'dev'))]
203
204     # Compila el proyecto con las configuraciones de Sonar.
205     - template: './installJava.yml@nanaykuna-infra'
206
207     # =====
208     # PRUEBAS Y CONSTRUCCIÓN.
209     # =====
210
211     - template: './dotnetBuild.yml'
212     parameters:
213         skipTasks: $(skipTasks)
214         PublishProject: $(CsprojPaths)
215         SonarProjectKey: $(KPROJPath)
216         SonarProjectName: $(PROJPath)
217
218     # =====
219     # DESPLIEGUE.
220     # =====
221
222     - template: './sonar/sonar.yml'
223     parameters:
224         skipTasks: '$(skipTasks)'
225
226     - task: DotNetCoreCLI@2
227     condition: and(succeeded(), eq(variables['SKIPTASKS'], 'false'), ne(variables['Build.Reason'], 'PullRequest'))
228     inputs:
229         command: publish
230         projects: "src/**/$(CsprojPaths)"
231         publishWebProjects: false
232         arguments: "--configuration Release -v q --output $(build.artifactstagingdirectory)"
233         zipAfterPublish: True
234         displayName: "T14 - Publish locally"
235
236     - task: PublishBuildArtifacts@1
237     condition: and(succeeded(), ne(variables['Build.Reason'], 'PullRequest'))
238     inputs:
239         PathToPublish: "$(build.artifactstagingdirectory)"
240         displayName: "T15 - Publish to Azure DevOps Artifacts"
241
242     - template: './snkyTest.yml'
243     parameters:
244         SolutionPath: $(SlnPaths)
245         skipTasks: ${{ parameters.skipTasks }}
246

```

#### [./templates/generalBuildSteps.yml](#)

```

1  ---
2  # Este template fue revisado por última vez en [27/10/2023]. Es recomendable revisarlo periódicamente para asegurar su óptimo funcionamiento.
3  # Realiza el proceso de construcción, prueba y despliegue para cada proyecto FrontEnd.
4
5  parameters:
6
7      - name: skipTasks
8        type: string
9        default: ''
10
11     - name: debugMode
12       default: ''
13

```

```

14 - name: environmentName
15   type: string
16   default: ''
17
18 - name: keyVaultCredential
19   type: string
20   default: ''
21
22 - name: jsonPath
23   default: './pipelines/frontend.json'
24
25 steps:
26
27   # =====
28   # Variables.
29   # =====
30
31   - script: |
32     git config --global user.email 'dev.tools@progresol.com'
33     git config --global user.name 'Dev Tool Progresol'
34
35     sudo apt-get update >> /dev/null && sudo apt-get install -y jq >> /dev/null
36     legacyStatus=$(jq -r '.Legacy' "${{ parameters.jsonPath }}") >> /dev/null
37     projectName=$(jq -r '.projectName' "${{ parameters.jsonPath }}") >> /dev/null
38
39     echo "##vso[task.setvariable variable=PROJPath]nanaykuna-$projectName"
40     echo "##vso[task.setvariable variable=KPROJPath]nanaykuna_nanaykuna-$projectName"
41     displayName: 'Set variable exists'
42     timeoutInMinutes: 1
43
44   # =====
45   # DEPURACIÓN.
46   # =====
47
48   - script: |
49     # Primero, verificamos el valor de debugMode
50     echo "debugMode value: ${{ parameters.debugMode }}"
51
52     if [ "${{ parameters.debugMode }}" != "true" ] && [ "${{ parameters.debugMode }}" != "false" ]; then
53       echo "ERROR: El valor de debugMode es inválido. Debe ser 'true' o 'false'."
54       debugMode="false"
55     fi
56
57     if [ "${{ parameters.debugMode }}" == "true" ]; then
58       echo "Starting Debug Validations..."
59
60       if [ -z $environmentName ]; then
61         echo "ERROR: environmentName is empty."
62         exit 1
63       fi
64
65       echo "Finished Debug Validations."
66     fi
67     displayName: 'Debug Validations'
68
69   - script: |
70     echo "Configuración del Pipeline:"
71
72     if [ -f "$(SLNPath)" ]; then
73       echo "El archivo de solución .sln en la ruta $(SLNPath) existe."
74     else
75       echo "ERROR: El archivo de solución .sln en la ruta $(SLNPath) no existe."
76       exit 1
77     fi
78
79     env | grep -i "Path" >> registro.log 2>&1
80     grep 'ERROR' registro.log >> errores.log
81     grep -v 'DEBUG' registro.log >> logs_limpio.log
82     displayName: 'Operaciones basadas en archivo .sln'
83     condition: and(succeeded(), eq(variables['debugMode'], 'true'))
84
85   - task: PublishBuildArtifacts@1
86     inputs:
87       PathToPublish: 'registro.log'
88       ArtifactName: 'logs'
89       displayName: 'Publicar logs'
90       condition: eq(variables['debugMode'], 'true')
91
92   # =====
93   # INSTALLATIONS.
94   # =====
95
96   - template: ./installSOPS.yml

```



```

97
98 - template: ./decryptSOPS.yml
99 parameters:
100   DecryptedFilePath: $(System.DefaultWorkingDirectory)/.env.development
101   FileToDecryptFilePath2: $(System.DefaultWorkingDirectory)/env.enc.sta
102   FileToDecryptFilePath3: $(System.DefaultWorkingDirectory)/env.enc.pr
103   FileToDecryptFilePath: $(System.DefaultWorkingDirectory)/env.enc.${{ parameters.environmentName }}
104
105 - template: installzip.yml@nanaykuna-infra
106 parameters:
107   message: 00 - Install Zip
108
109 # Compila el proyecto con las configuraciones de Sonar.
110 - template: installJava.yml@nanaykuna-infra
111
112 - task: NodeTool@0
113 inputs:
114   versionSpec: 16.16.0
115   displayName: 01 - Install Node.js
116
117 - task: SonarCloudPrepare@1
118 inputs:
119   SonarCloud: Sonar Cloud
120   organization: nanaykuna
121   scannerMode: CLI
122   configMode: manual
123   cliProjectKey: '$(KPROJPath)'
124   cliProjectName: '$(PROJPath)'
125   cliSources: .
126   extraProperties: sonar.javascript.lcov.reportPaths=./coverage/lcov.info
127   displayName: 02 - Sonar Cloud Prepare
128
129 - script: |
130   npm install -g yarn
131   yarn -v
132   yarn install
133   continueOnError: true
134   displayName: 03 - Yarn Install
135
136 - script: |
137   yarn run standard-version
138   current_message=$(git show -s --format=%s)
139   echo $current_message
140   commit_message="{current_message} [ci skip]"
141   echo $commit_message
142   git commit --amend -m --no-verify "$commit_message"
143   git push --follow-tags --no-verify origin HEAD:${Build.SourceBranchName}
144   continueOnError: true
145   displayName: 04 - Run Version
146
147 # =====
148 # PRUEBAS.
149 # =====
150
151 - script: |
152   yarn test --coverage
153   continueOnError: true
154   displayName: 05 - Testing
155
156 - task: PublishCodeCoverageResults@1
157 inputs:
158   codeCoverageTool: Cobertura
159   summaryFileLocation: $(System.DefaultWorkingDirectory)/coverage/cobertura-coverage.xml
160   displayName: 06 - Publish Tests PublishCodeCoverageResults
161
162 - task: PublishTestResults@2
163 inputs:
164   testResultsFormat: JUnit
165   testResultsFiles: "**/*unit.xml"
166   mergeTestResults: true
167   displayName: 07 - Publish Tests Results
168
169 - task: SonarCloudAnalyze@1
170 continueOnError: false
171 displayName: 08 - Sonar Cloud Analyze
172
173 - template: ./sonar/sonar.yml
174 parameters:
175   skipTasks: '$(skipTasks)'
176
177 - task: SnykSecurityScan@1
178 inputs:
179   serviceConnectionEndpoint: snyk

```

```

180     testType: app
181     monitorWhen: always
182     failOnIssues: false
183     continueOnError: true
184     displayName: 09 - Security Scan
185
186     # =====
187     # CONSTRUCCIÓN.
188     # =====
189
190     - script: |
191         yarn run build:dev
192         displayName: 10 - Build
193         continueOnError: false
194
195     # =====
196     # DESPLIEGUE.
197     # =====
198
199     - task: ArchiveFiles@2
200       inputs:
201         rootFolderOrFile: .
202         includeRootFolder: false
203         archiveType: zip
204         archiveFile: $(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip
205         replaceExistingArchive: true
206         displayName: 11 - Publish locally
207
208     - task: PublishBuildArtifacts@1
209       inputs:
210         PathToPublish: $(Build.ArtifactStagingDirectory)
211         ArtifactName: drop
212         displayName: 12 - Publish to Azure DevOps Artifacts
213

```

👉 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:.