

Precondiciones:

1. Crear una Cuenta gratuita en Azure Devops con tu cuenta de correo de Choucair
<https://dev.azure.com/>
2. Una vez finalizado el proceso de crear la cuenta, procedemos a crear un nuevo Proyecto en Azure Devops y cargar el código del siguiente enlace:

Nota: Puedes también cargar un código que tengas de ejemplo.

https://drive.google.com/open?id=1robn4F1R3UeA4NHA0-As_NLS5FIPEY

Create a project to get started

Project name *

TestProject ✓

Visibility

☒ Public
Anyone on the internet can view the project. Certain features like TFVC are not supported.

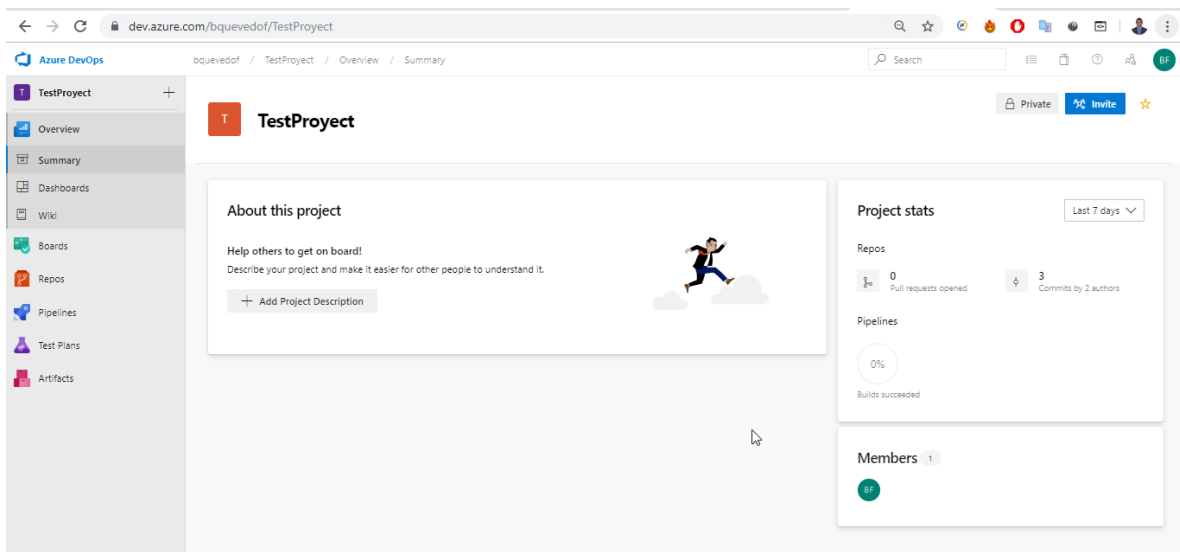
☐ Private
Only people you give access to will be able to view this project.

+ Create project

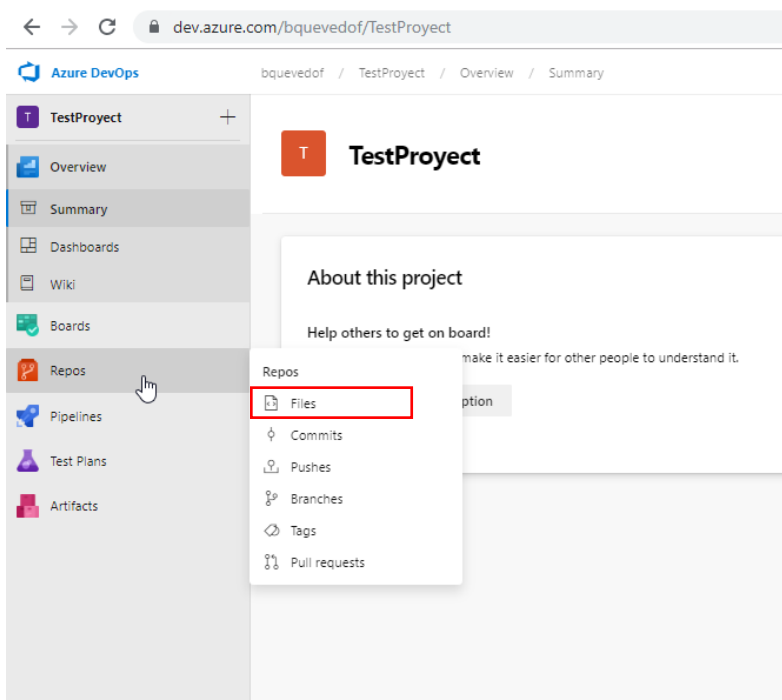
3. Se debe crear un repositorio que en nuestro caso lo llamaremos **"ProyectTest"**.

A continuación, explicaremos como crear un repositorio en azure DevOps.

Después de crear el proyecto, este se debería visualizar de la siguiente manera:



Para crear el repositorio, en el menú izquierdo nos posicionamos sobre “Repos>Files”.



Luego de haber dado clic debemos ver la siguiente pantalla.

ProjectTest is empty. Add some code!

^ Clone to your computer

HTTPS SSH OR

Having problems authenticating in Git? Be sure to get the latest version of [Git for Windows](#) or our plugins for [IntelliJ](#), [Eclipse](#), [Android Studio](#) or [Windows command line](#).

^ or push an existing repository from command line

HTTPS SSH

```
git remote add origin https://bquevedof@dev.azure.com/bquevedof/ProjectTest/_git/ProjectTest
git push -u origin --all
```

^ or import a repository

^ or initialize with a README or gitignore

☒ Add a README

Ahora continuaremos con la configuración del README y el archivo gitignore los cuales se pueden visualizar al final de la página, para ello habilitaremos el check de “Add a README”, daremos clic en botón “Add a .gitignore” y en el input buscaremos para palabra “Java” y por ultimo damos clic al botón “Initialize”.

^ or push an existing repository

HTTPS SSH

Java

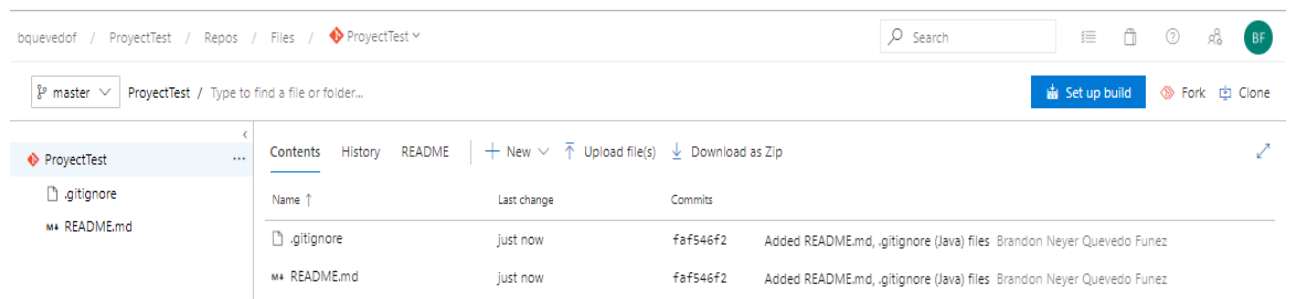
```
git remote add origin https://bquevedof@dev.azure.com/bquevedof/ProjectTest/_git/ProjectTest
git push -u origin --all
```

^ or import a repository

^ or initialize with a README or gitignore

☒ Add a README

Luego de haber realizado el paso anterior, debemos visualizar lo siguiente:



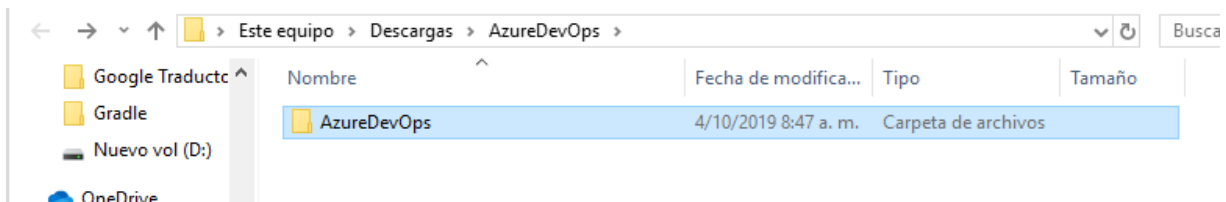
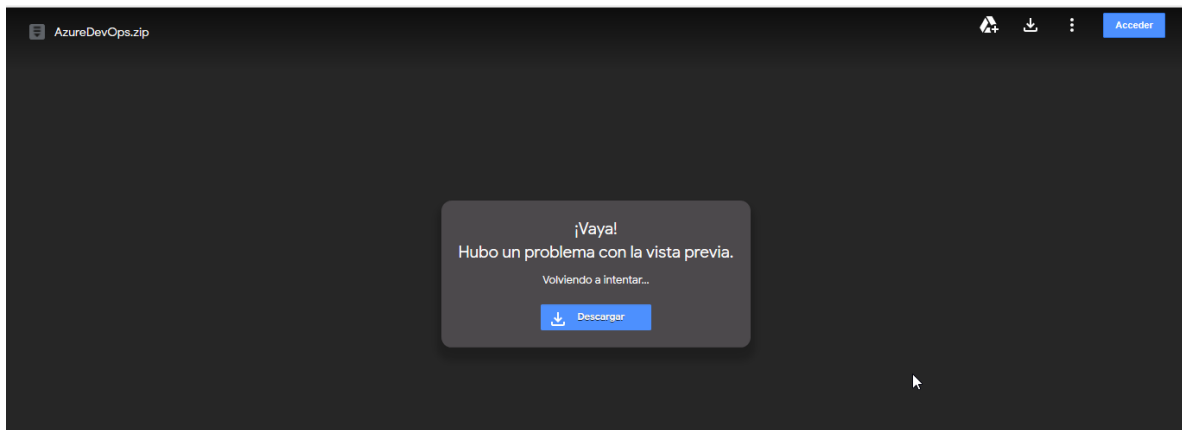
The screenshot shows a GitHub repository page for 'ProjectTest'. The breadcrumb navigation at the top reads 'bquevedof / ProjectTest / Repos / Files / ProjectTest'. A search bar is located in the top right. Below the navigation, there's a dropdown for 'master' and a search input 'ProjectTest / Type to find a file or folder...'. Action buttons include 'Set up build', 'Fork', and 'Clone'. The left sidebar shows the file tree with '.gitignore' and 'README.md'. The main content area has tabs for 'Contents', 'History', and 'README'. Below these tabs are links for '+ New', 'Upload file(s)', and 'Download as Zip'. A table lists the repository's contents:

Name ↑	Last change	Commits	
.gitignore	just now	faf546f2	Added README.md, .gitignore (Java) files Brandon Neyer Quevedo Funez
README.md	just now	faf546f2	Added README.md, .gitignore (Java) files Brandon Neyer Quevedo Funez

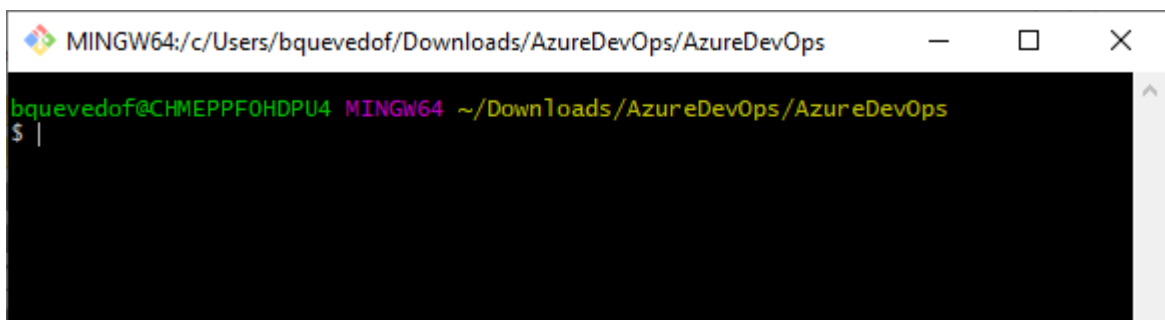
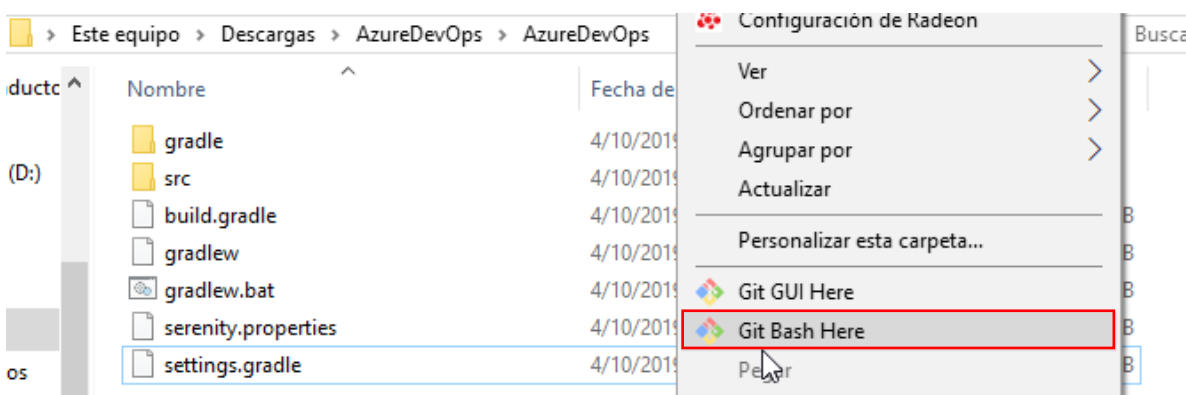
A partir de este momento vamos a hacer un paréntesis, debido a que vamos a cargar el proyecto a él repositorio de Azure a través de git.

Nota: El proyecto a cargar esta realizado en el patrón Screenplay y realiza una traducción de una palabra en el traductor de Google.

Una vez descargado el proyecto de la URL entrega en la parte inicial del presente Laboratorio, lo descomprimos en el lugar que deseemos.



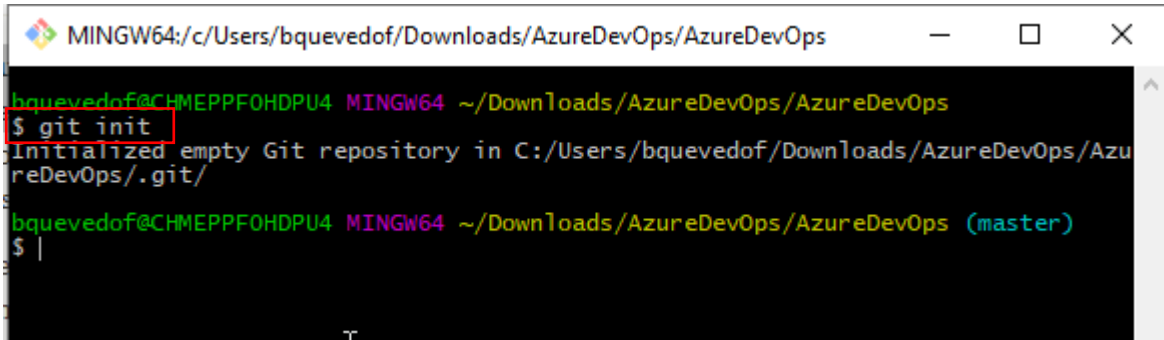
Después vamos a ingresar a la carpeta donde se encuentra el mismo y vamos a dar clic derecho y abriremos la consola de Git Bash.



Desde esta, y a través de los comandos de git nos podremos subir nuestro proyecto al repositorio de Azure DevOps.

El primer comando que utilizaremos será:

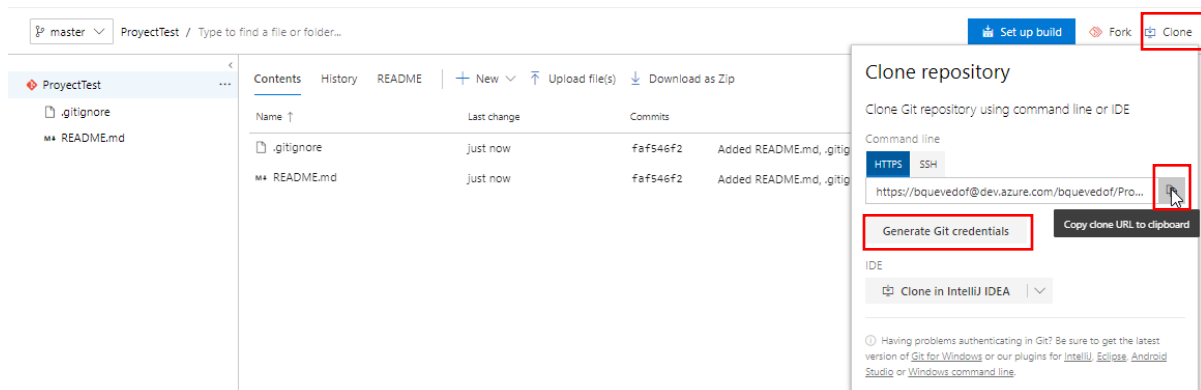
- “**git init**”. Con este comando se crea un nuevo subdirectorio llamado “. git” que contiene todos los archivos necesarios del repositorio es decir un esqueleto de un repositorio Git.



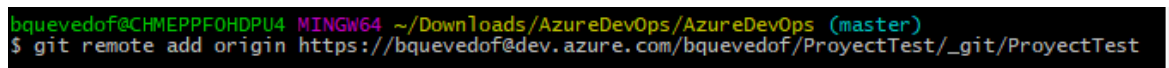
```
MINGW64/c:/Users/bquevedof/Downloads/AzureDevOps/AzureDevOps
bquevedof@CHMEPPFOHDP4 MINGW64 ~/Downloads/AzureDevOps/AzureDevOps
$ git init
Initialized empty Git repository in C:/Users/bquevedof/Downloads/AzureDevOps/AzureDevOps/.git/
bquevedof@CHMEPPFOHDP4 MINGW64 ~/Downloads/AzureDevOps/AzureDevOps (master)
$ |
```

Seguidamente utilizaremos el comando:

- “**git remote add origin <<url repositorio remoto>>**” que será el encargado de conectarnos con el repositorio remoto de azure, para esto vamos a volver al azure a buscar el link para conectarnos al repositorio.



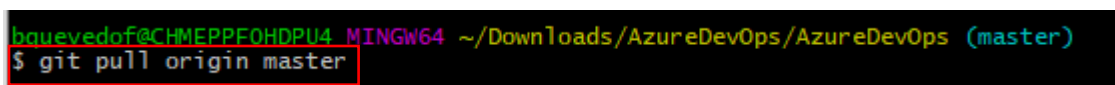
Vamos a dar clic al botón “Clone” , donde nos veremos la url que necesitamos, seguidamente damos clic en el botón “**Generate Git credentials**” y por ultimo copiamos la Url y volvemos a la consola de git.



```
bquevedof@CHMEPPFOHDP4 MINGW64 ~/Downloads/AzureDevOps/AzureDevOps (master)
$ git remote add origin https://bquevedof@dev.azure.com/bquevedof/ProjectTest/_git/ProjectTest
```

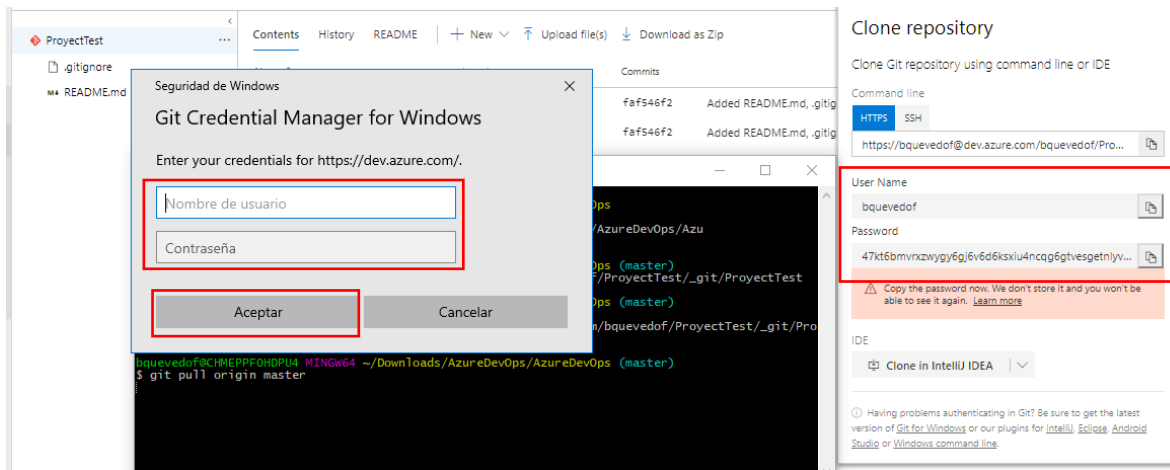
Continuamos con el comando:

- “**git pull origin <<rama>>**”, con este comando vamos a bajar los dos archivos que tenemos en el repositorio el README y el .gitignore.

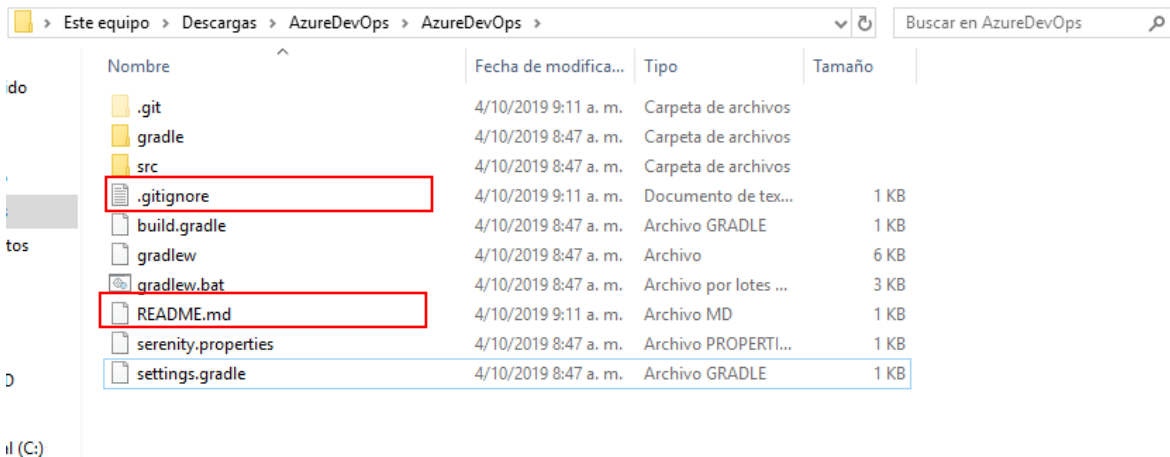


```
bquevedof@CHMEPPFOHDP4 MINGW64 ~/Downloads/AzureDevOps/AzureDevOps (master)
$ git pull origin master
```

Una vez ejecutado este comando, nos solicitara credenciales las cuales fueron generadas cuando dimos clic al botón **“Generate Git credentials”** en Azure. Ingresamos el usuario y contraseña y damos clic en el botón aceptar.



Una vez ingresadas las credenciales, procedemos a verificar que en nuestro proyecto local se hayan descargado los dos archivos que estaban en el repositorio (**README.md y .gitignore**).



Ahora procedemos a modificar el archivo “.gitignore” de nuestro proyecto local, lo vamos a abrir en un editor de texto y vamos a adicionar las siguientes líneas al final del mismo. Guardamos los cambios y cerramos el archivo.

```
.gradle
build/
# Ignore Gradle GUI config
gradle-app.setting
# Avoid ignoring Gradle wrapper jar file (.jar files are usually
ignored)
!gradle-wrapper.jar
# Cache of project
.gradle.taskname.cache
```

Nuevamente volvemos a la consola de git para subir nuestro proyecto automatizado con los últimos cambios realizados, para esto utilizaremos el comando:

- “git add -A” para preparar los archivos que vamos a subir.

```
bquevedof@CHMEPPFOHDP4 MINGW64 ~/Downloads/AzureDevOps/AzureDevOps (master)
$ git add -A
```

Luego usamos el comando:

- “git commit -m “Observacion” con el preparamos todo y asignamos la observación de los cambios que hemos realizado.

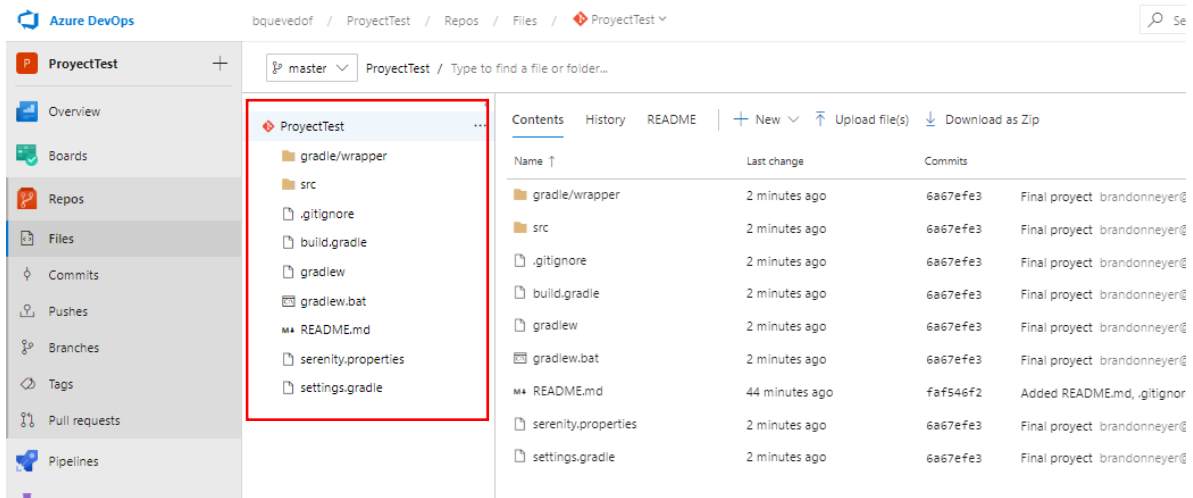
```
bquevedof@CHMEPPFOHDP4 MINGW64 ~/Downloads/AzureDevOps/AzureDevOps (master)
$ git commit -m "Final project"
```

Y por último utilizamos el comando:

- “git push origin master” para subir los archivos al repositorio de Azure DevOps.

```
bquevedof@CHMEPPFOHDP4 MINGW64 ~/Downloads/AzureDevOps/AzureDevOps (master)
$ git push origin master
Counting objects: 42, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (26/26), done.
Writing objects: 100% (42/42), 4.49 MiB | 733.00 KiB/s, done.
Total 42 (delta 2), reused 0 (delta 0)
remote: Analyzing objects... (42/42) (6520 ms)
remote: Storing packfile... done (242 ms)
remote: Storing index... done (37 ms)
To https://dev.azure.com/bquevedof/ProyectTest/_git/ProyectTest
   faf546f..6a67efe master -> master
```

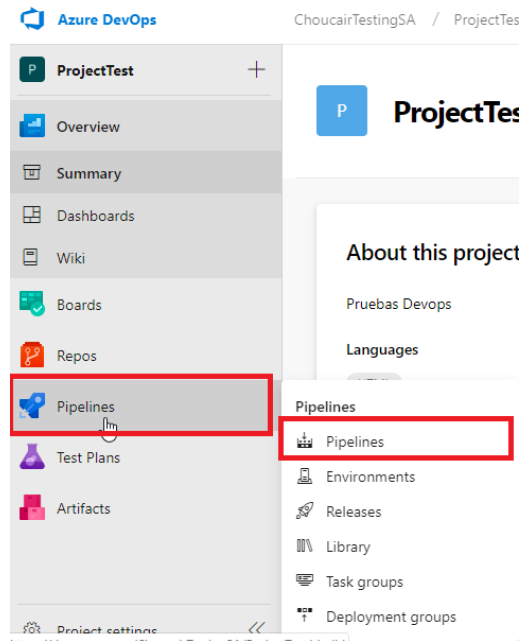

Una vez terminado el proceso, nos dirigimos al repositorio de Azure y al dar refresh a nuestra pantalla debemos visualizar todos los archivos ya cargados de nuestro proyecto.



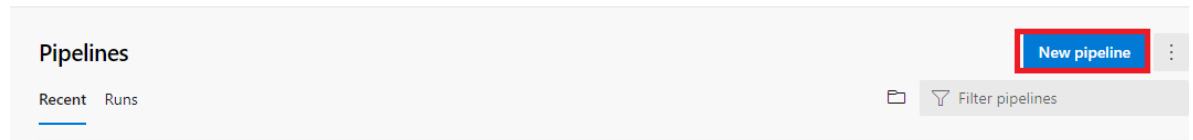
Ahora si estamos listos para crear los PipeLine (CI, CD, RM).

Creación Pipeline CI

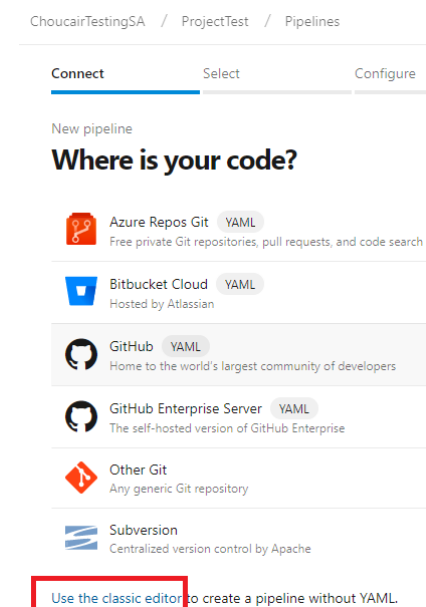
1. Acceder a Azure DevOps: <https://dev.azure.com/>
2. Dar clic en la opción Pipelines




3. Clic en “New Pipeline”



4. Como no tenemos ningún Template para nuestro Pipeline, procedemos a dar clic en el link de la parte inferior izquierda “Use the classic editor”.




5. Seleccionamos nuestro Project, Repositorio y rama Master y clic en el botón Continuar




Select your repository

Tell us where your sources are.
You can customize how to get these sources from the repository later.


Select a source




Azure Repos Git




GitHub




GitHub Enterprise Server



Subversion




Bitbucket Cloud




Other Git


Team project

 ProjectTest

Repository


 ProjectTest

Default branch for manual and scheduled builds

 master

6. Seleccionamos para continuar con “Empty Job”

Select a template

Or start with an  **Empty job**

7. Le damos un nombre nemotécnico a nuestro Pipeline CI: <<Nombre_Proyecto>>_CI

ProjectTest-CI

Tasks Variables Triggers Options Retention History Save & queue Discard Summary Queue ...

Pipeline Build pipeline

Get sources
ProjectTest master

Agent job 1 Run on agent

Name *
ProjectTest-CI

Agent pool * Pool information Manage
Azure Pipelines

Agent Specification *
vs2017-win2016

Parameters ⓘ

This pipeline doesn't have any pipeline parameters. Create them to share the most important settings between tasks and change them in one place.

8. Se debe limpiar los directorios cada que se compile la aplicación, para evitar fallas en el repositorio. Para esto en Get “Sources” se debe poner la opción “Clean” en **true** y “Clean Options” en **All builds directories**

Tasks Variables Triggers Options Retention History | Save & queue Discard Summary Queue ...

Pipeline
Build pipeline

Get sources
AW1112001_WebFondosPanama master

Phase 1
Run on agent

Prepare analysis on SonarQube
Prepare Analysis Configuration

gradlew build
Gradle

Run Code Analysis
Run Code Analysis

Default branch for manual and scheduled builds
master

Clean
true

Clean options
All build directories

Tag sources
☒ Never
☐ On success
☐ Always

9. Posteriormente, agregamos la Tarea de **SONAR** para nuestro “Agent Job” llamada “Prepare analys on SonarQube” dando clic en el “+”

Agent job 1
Run on agent

Add tasks | Refresh

prepare anal

Prepare Analysis Configuration
Prepare SonarQube analysis configuration

Deprecated tasks

10. Agregamos nuestro SonarQube Server Endpoint dando clic en “+ New”

- a. Clic en New

Display name *

Prepare analysis on SonarQube

SonarQube Server Endpoint * | Manage

+ New

- b. Diligenciamos los datos del Servicio de Conexión de Sonar y clic en “OK”

Name: sonartestch

Sever Url: http://sq-labch1028.eastasia.cloudapp.azure.com:9000/

Token: 0c7dd5d2200e0b7a4b9c8fb30833e8a9d6047a5f

Add SonarQube service connection

Connection name

Server Url ⓘ

Token ⓘ

☒ Allow all pipelines to use this connection.

OK

Close

11. Continuamos diligenciando los datos de nuestra tarea, seleccionando el Server Endpoint que acabamos de generar y los demás campos.

Display name *

SonarQube Server Endpoint * ⓘ | [Manage](#)

▼

Choose the way to run the analysis * ⓘ

☐ Integrate with MSBuild ☐ Integrate with Maven or Gradle

☒ Use standalone scanner

Mode * ⓘ

☐ Store configuration with my source code (sonar-project.properties)

☒ Manually provide configuration

Project Key and Name: \$(Build.Repository.Name)

Project Key * ⓘ

Project Name ⓘ

Project Version ⓘ

Sources directory root * ⓘ ...

Damos clic en la sección **Advanced** y en el campo “Additional Properties” ingresará lo siguiente.

```
sonar.sources= $(Build.SourcesDirectory)/src/main/java
sonar.tests = $(Build.SourcesDirectory)/src/test/java
sonar.java.binaries=$(Build.SourcesDirectory)/build/classes
```


- Configuración de los “Triggers”

Esta configuración se debe habilitar para el Pipeline de CI donde se habilitará la ejecución automática cuando se realicen “commits” en las distintas ramas.

1. develop
2. release
3. master

Tasks Variables **Triggers** Options Retention History | Save & queue Discard Summary Queue ...

Continuous integration

☒ Enable continuous integration 

☐ Batch changes while a build is in progress

Branch filters

Type	Branch specification
Include	master
Include	develop
Include	release

+ Add

Path filters


+ Add

12. Nuevamente, agregamos otra Tarea para nuestro “Agent Job” llamada “Gradle” dando clic en el “+”.

Agent job 1
Run on agent

+ Add

Add tasks | Refresh | gradle

 **Gradle**
Build using a Gradle wrapper script

Deprecated tasks

- a. Diligenciamos los siguientes datos en la tarea

Display name: gradlew clean build -x test

Task: clean build -x test

Gradle ⓘ [Link settings](#) [View YAML](#) [Re](#)

Task version 2.* ▼

Display name *
gradlew clean build -x test

Gradle wrapper * ⓘ
gradlew ...

Working directory ⓘ
...

Options ⓘ

Tasks * ⓘ
clean build -x test

JUnit Test Results ^
☐ Publish to Azure Pipelines ⓘ

Code Coverage ^
Code coverage tool ⓘ
None ▼

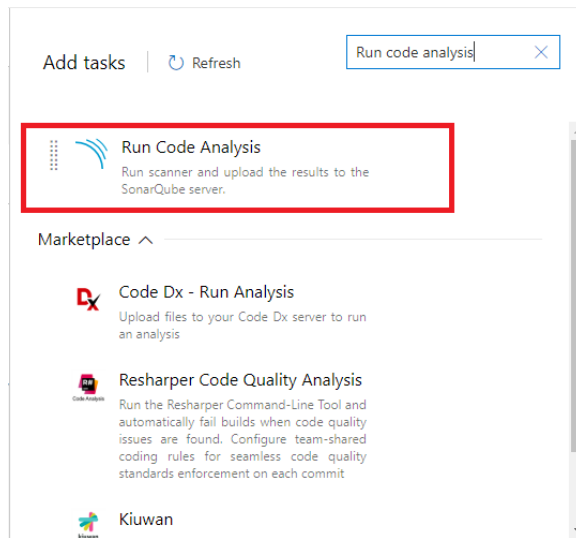
Advanced ▼

Code Analysis ^
☐ Run SonarQube or SonarCloud Analysis ⓘ
☐ Run Checkstyle ⓘ
☐ Run FindBugs ⓘ

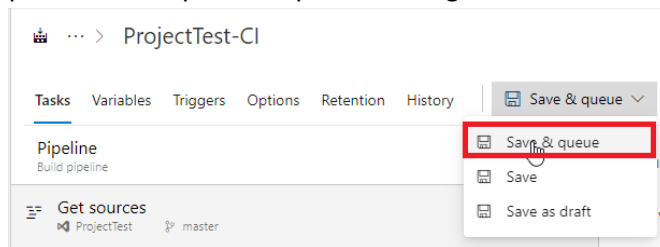
13. Para terminar, agregamos una nueva tarea para nuestro “Agent Job” llamada “Run Code Analysis” dando clic en el “+” y dejamos tal como la carga nuestra herramienta.

Agent job 1
Run on agent

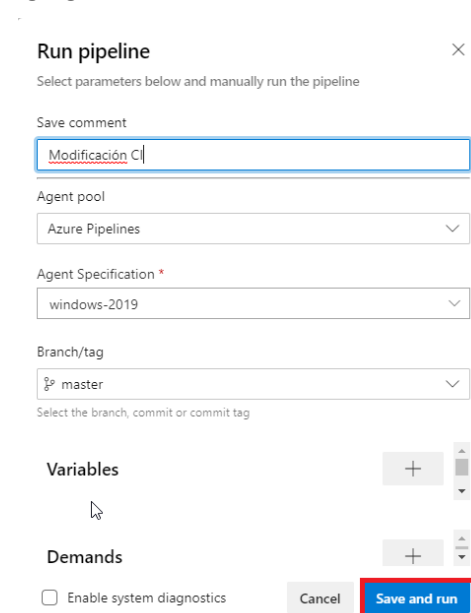
+



14. Terminado de configurar nuestro Pipeline de CI, procedemos a guardar y correr el mismo para verificar que no se presente ningún error.



Agregamos comentarios sobre la modificación realizada, grabamos y ejecutar.



Podemos ir observando la ejecución, dando clic en el job.

#10 version 1.0
on ProjectTest-CI

Cancel

Summary

Manually run by Yeison Arias

ProjectTest master 684fc63

Just now

Duration: -

Tests: -

Changes: 1 commit

Work items: -

Artifacts: -

Jobs

Name	Status	Duration
Agent job 1	Queued	

← Jobs in run #10
ProjectTest-CI

Agent job 1 24s

Agent job 1

Job information

1 Pool: Azure Pipelines

2 Image: windows-2019

3 Agent: Hosted Agent

4 Started: Just now

5

6 The agent request is already running or has already completed.

7 ▶ Job preparation parameters

8 ▶ fr 1 queue time variable used

✓ Initialize job 3s

✓ Checkout 5s

✓ Prepare analysis on SonarQube 1s

○ gradlew clean build -x test 13s

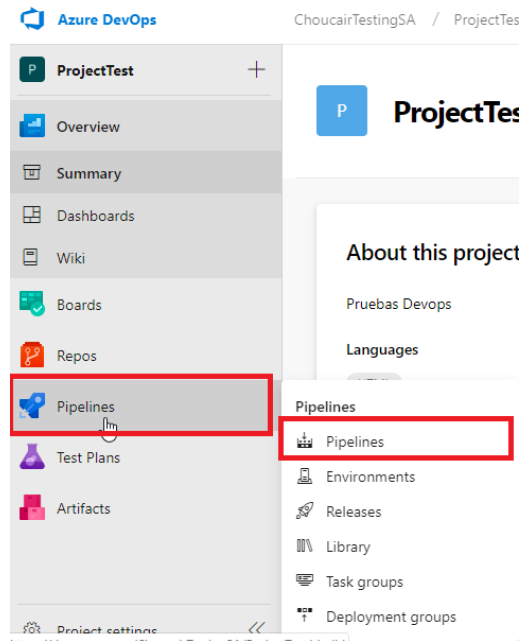
○ Run Code Analysis

○ Post-job: Checkout

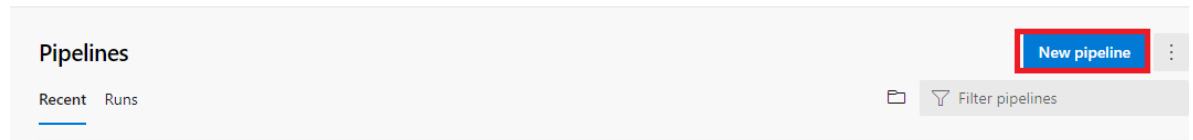
↓ ↑

Creación Pipeline CD

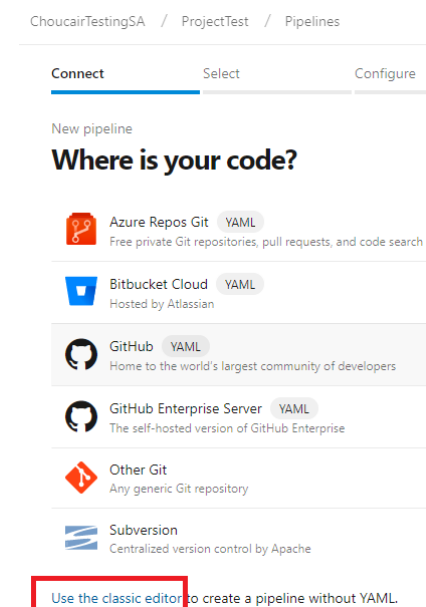
1. Acceder a Azure DevOps: <https://dev.azure.com/>
2. Dar clic en la opción Pipelines



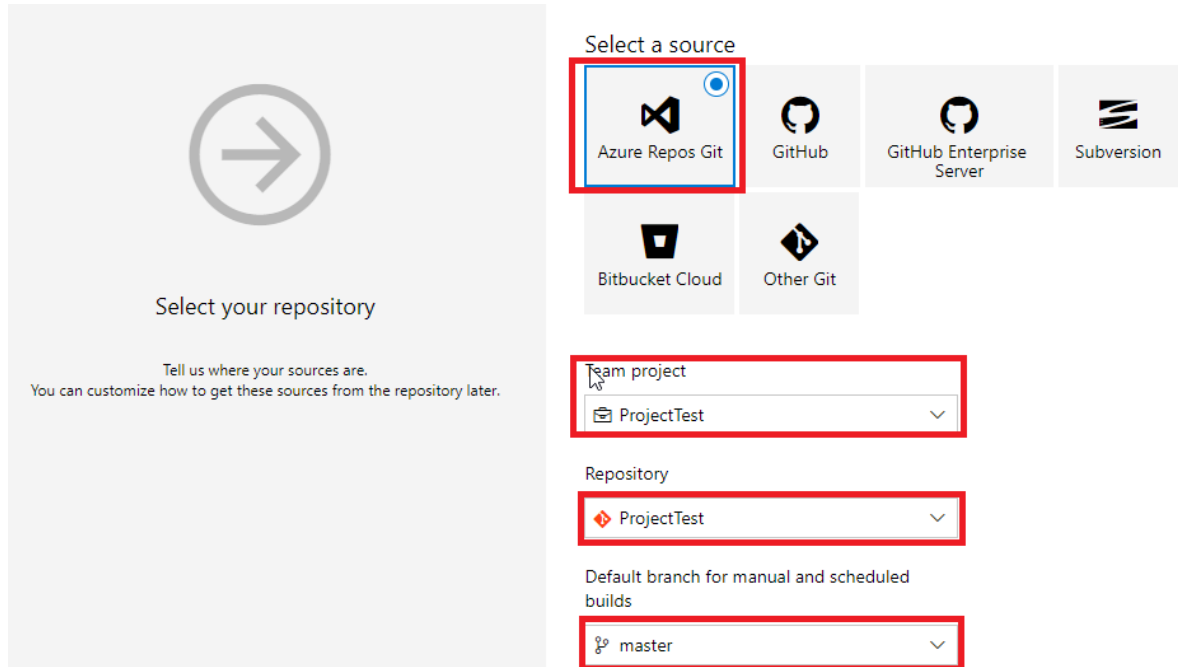
3. Clic en “New Pipeline”



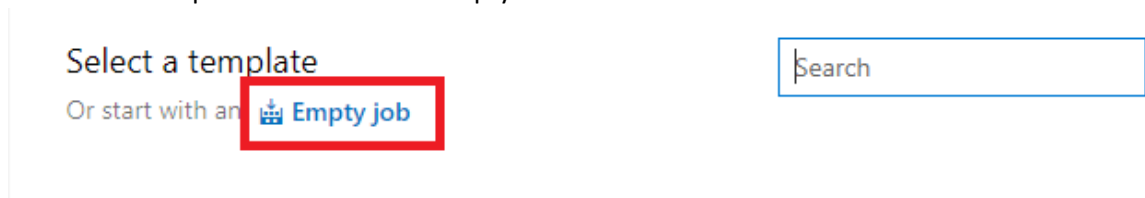
4. Como no tenemos ningún Template para nuestro Pipeline, procedemos a dar clic en el link de la parte inferior izquierda “Use the classic editor”.



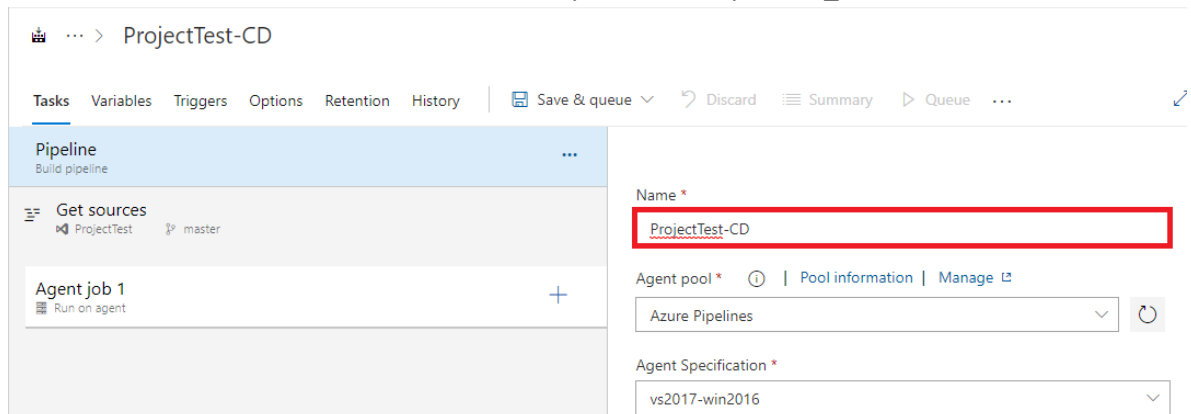
5. Seleccionamos nuestro Project, Repositorio y rama Master y clic en el botón Continuar



6. Seleccionamos para continuar con “Empty Job”



7. Le damos un nombre nemotécnico a nuestro Pipeline: <<Proyecto>>_CD



8. Se deben limpiar los directorios cada que se compile la aplicación, para evitar fallas en el repositorio. Para esto en “Get Sources” se debe poner la opción “Clean” en **true** y “Clean Options” en **All builds directories**

The screenshot shows the Jenkins Pipeline configuration page for a pipeline named 'Get sources'. The left sidebar shows the pipeline structure with 'Get sources' selected. The main area shows the configuration for 'Get sources' on the 'master' branch. The 'Clean' option is set to 'true' and the 'Clean options' are set to 'All build directories'. The 'Tag sources' option is set to 'Never'.

9. Ahora vamos a comenzar a agregar nuestras Tareas “task” a nuestro “Agent job”

The screenshot shows the Jenkins 'Agent job 1' configuration page. The 'Agent job 1' section is highlighted with a red box, and a red box with a plus sign is visible on the right side of the section, indicating where to click to add tasks.

La primera de ella es “gradle” la cual una vez agregada la configuramos.

The screenshot shows the 'Add tasks' input field with the text 'gradle' entered. A red box highlights the input field.

The screenshot shows the 'Gradle' task configuration. The task is named 'Gradle' and has the description 'Build using a Gradle wrapper script'. A red box highlights the 'Add' button.

Display Name: **gradlew clean build -x test**

Tasks: **clean build -x test**

Task version 2.*

Display name *

gradlew clean build -x test

Gradle wrapper *

gradlew

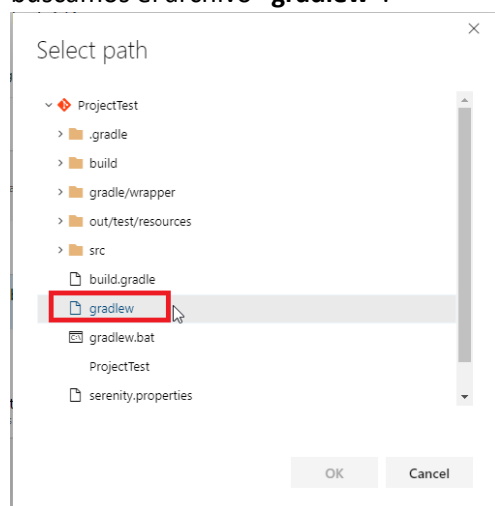
Working directory

Options

Tasks *

clean build -x test

Para el campo Gradle wrapper, damos clic en el campo para seleccionar ubicación y buscamos el archivo “**gradlew**”:

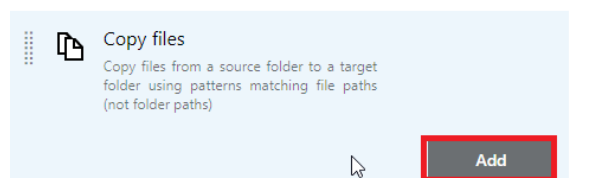


10. Agregar una nueva tarea llamada “Copy files”

Add tasks

Refresh

Copy files



Y se configuran sus parámetros

Display Name: **Copy files**

Source folder: **\$(system.defaultworkingdirectory)**

Contents:

****/***
!build

Target folder: **\$(build.artifactstagingdirectory)**

Display name *

Copy Files

Source Folder ⓘ

\$(system.defaultworkingdirectory) ...

Contents * ⓘ

**/*
!build

Target Folder * ⓘ


\$(build.artifactstagingdirectory)

Advanced ^

11. Agregamos la última tarea llamada “Publish Artifact”:

Add tasks | Refresh

Publish artifacts

**Publish build artifacts**
Publish build artifacts to Azure Pipelines or a Windows file share

Add

Para esta configuramos los valores:

Display name: **Publish Artifact**

Path to publish: **\$(Build.ArtifactStagingDirectory)**

Artifact name: **ProjectTestArtifact**

Display name *

Path to publish * ⓘ
 ...

Artifact name * ⓘ

Artifact publish location * ⓘ

Una vez finalizado, procedemos a Salvar y ejecutar nuestro pipeline.

Tasks Variables Triggers Options Retention History **Save & queue**

Pipeline
Build pipeline

Get sources
ProjectTest master

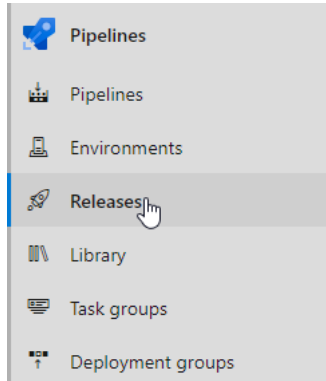
- Save & queue
- Save
- Save as draft

Run pipeline

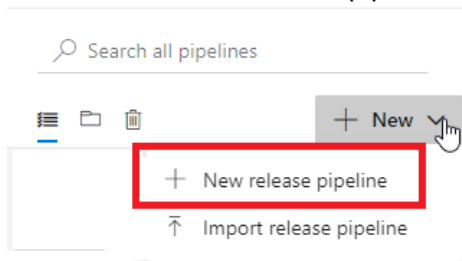
Save and run

Creación Pipeline RM

1. Damos clic en la opción de menú: Releases



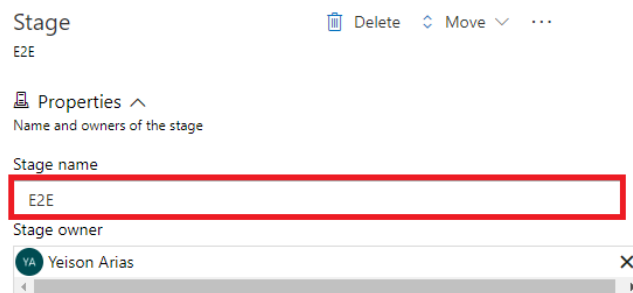
2. Creamos un nuevo "Release pipeline"



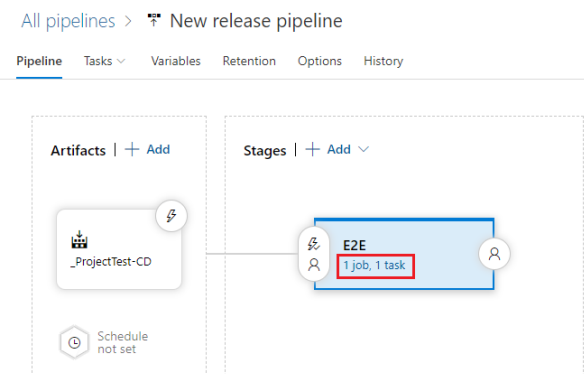
3. Seleccionamos "Empty Job"



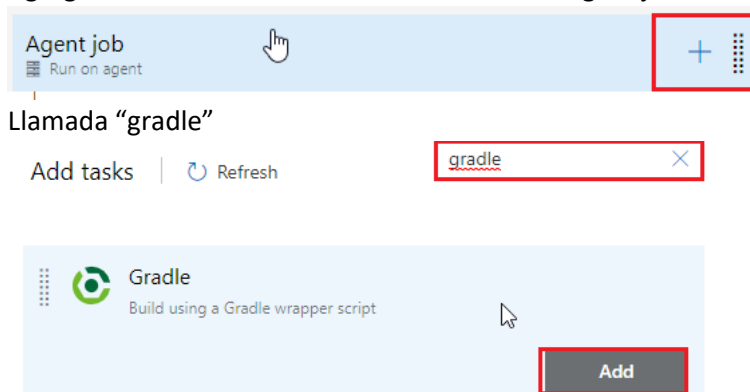
4. Le damos nombre a nuestro Stage "E2E"



5. Damos clic en Job, task



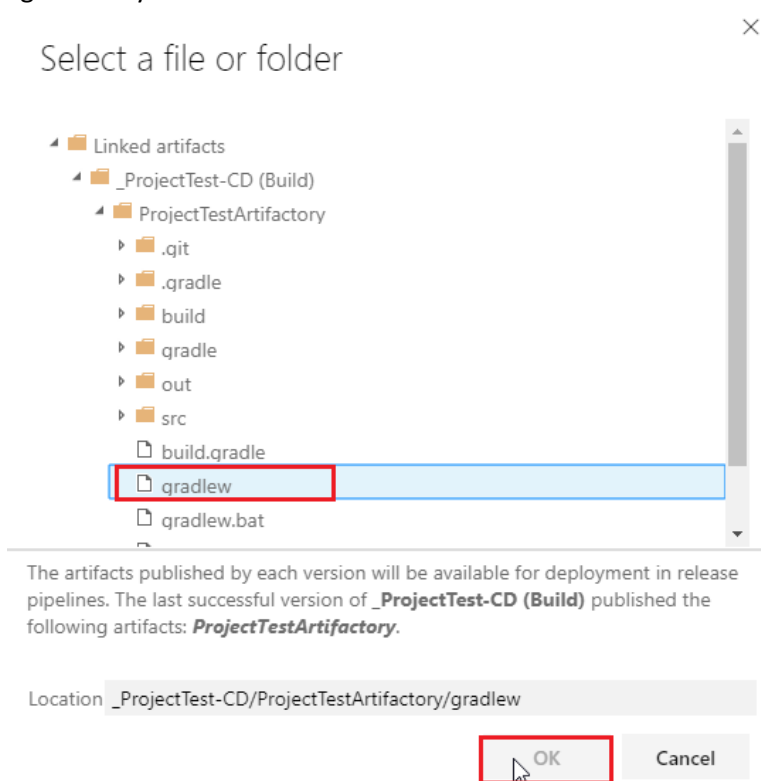
6. Agregamos una nueva tarea “Task” a nuestro “Agent job”



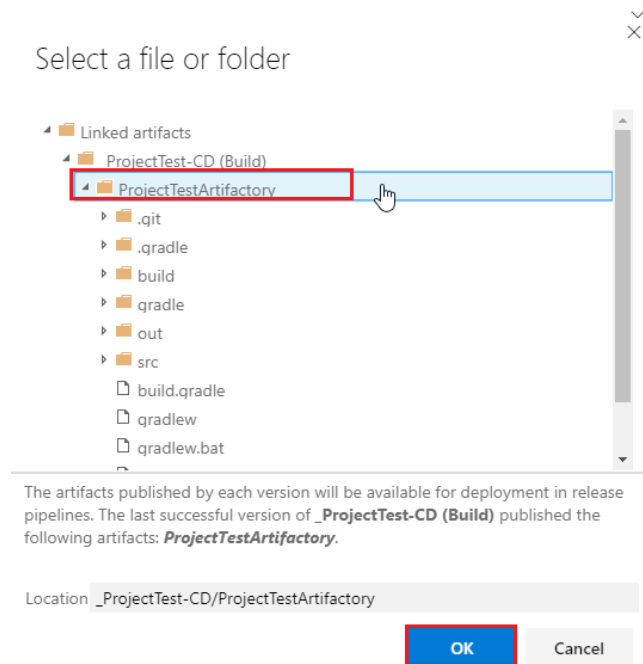
7. Parametrizamos la tarea de la siguiente manera:

Display name: **gradlew**

Para el campo “Gradle wrapper” damos clic en el botón para seleccionar el archivo “gradlew” y clic en el botón “OK”



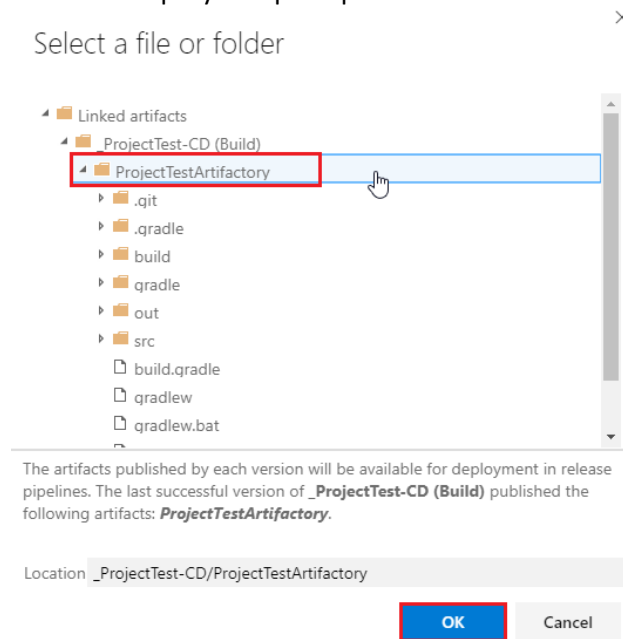
8. Para el campo “Working directory”, de igual manera damos clic en el botón para especificar ruta y seleccionamos el directorio raíz de nuestro repositorio de código y clic en el botón “OK”.



9. En los campos:

Task: **clean test --info aggregate**

10. Para el campo "Test results files" damos clic para seleccionar inicialmente como ubicación el proyecto principal:



Y a la ruta que queda agregada:

```
$(System.DefaultWorkingDirectory)/_ProjectTest-  
CD/ProjectTestArtifactory
```

Le complementamos con el siguiente segmento: **/target/site/serenity/SERENITY-*.xml**

Quedando como ruta definitiva: `$(System.DefaultWorkingDirectory)/_ProjectTest-CD/ProjectTestArtifact/target/site/serenity/SERENITY-*.xml`

Test results files *



`$(System.DefaultWorkingDirectory)/_ProjectTest-CD/ProjectTestArtifact/target/site/serenity/SERENITY-*.xml`



11. En el campo “Test run title” colocamos: “pruebas de aceptación” y finalizamos dando clic en SAVE



Create release ...

`$(System.DefaultWorkingDirectory)/_ProjectTest-CD/ProjectTestArtifact/gradlew`



Working directory ⓘ

`$(System.DefaultWorkingDirectory)/_ProjectTest-CD/ProjectTestArtifact`



Options

Tasks *

`clean test --info aggregate`

JUnit Test Results ^

☒ Publish to Azure Pipelines ⓘ

Test results files *



`$(System.DefaultWorkingDirectory)/_ProjectTest-CD/ProjectTestArtifact/target/site/serenity/SERENITY-*.xml`



Test run title ⓘ

`pruebas de aceptacion`

12. Para ejecutar el pipeline de RM. Damos clic en el botón “Create Release”

All pipelines > New release pipeline



Create release ...

Pipeline Tasks Variables Retention Options History

E2E

Deployment process



Gradle ⓘ

View YAML

Damos clic en el Stage “E2E” y luego clic en el botón “Crear”

Create a new release

New release pipeline

⚙ Pipeline ^

Click on a stage to change its trigger from automated to manual.

E2E

Stages for a trigger change from automated to manual. ⓘ

📦 Artifacts ^

Select the version for the artifact sources for this release

Source alias	Version	
_ProjectTest-CD	18	▼

Release description

Create Cancel

Para continuar con la ejecución del Pipeline damos clic en el link

[All pipelines](#) > [New release pipeline](#)

✓ Release **Release-4** has been created

Luego colocamos el cursor sobre el Stage “E2E” y damos clic en “Deploy”

New release pipeline > Release-5 ▼

Pipeline Variables History | + Deploy ▼ Cancel Refresh Edit ...

Release

Manually triggered
by Yelson Arias
9/10/2019, 3:28 PM

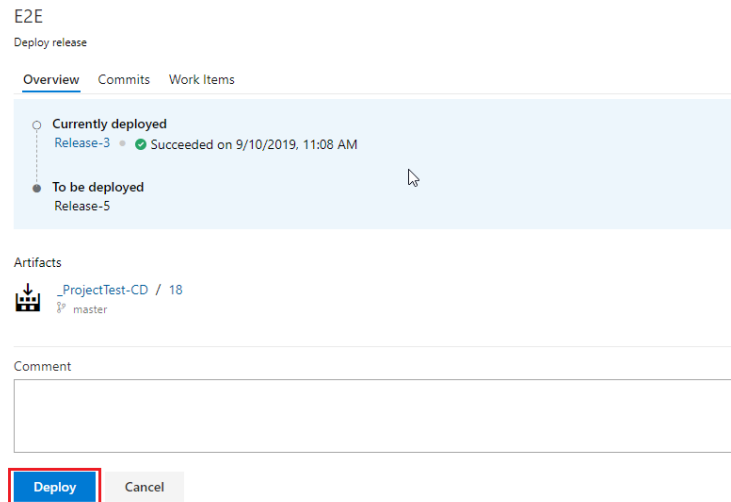
Artifacts
 _ProjectTest-CD
18
🔗 master

Stages

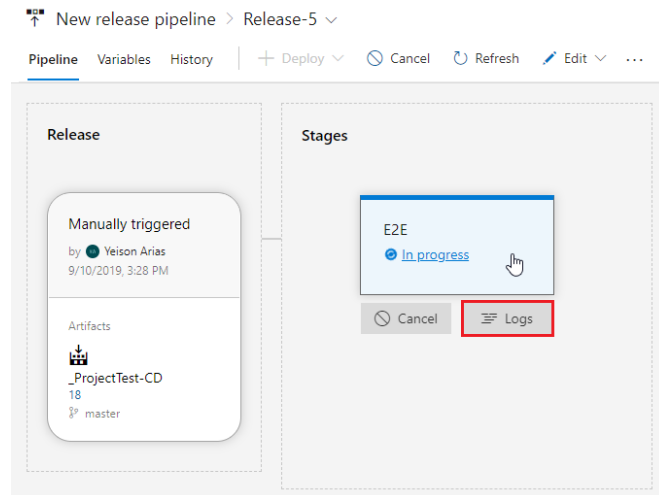
E2E
○ Not deployed

Deploy Logs

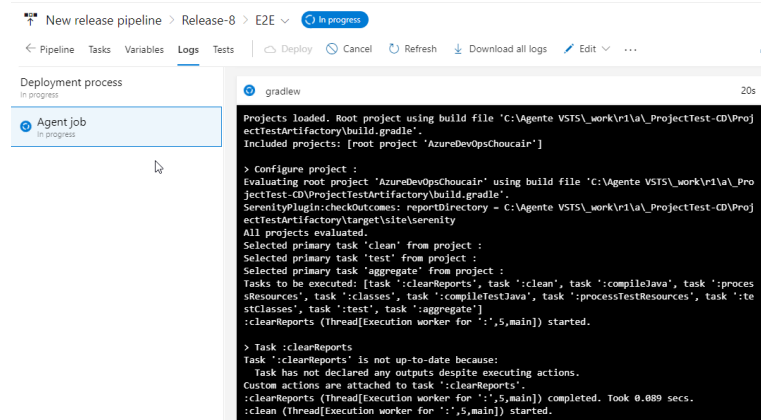
Nuevamente clic en el botón “Deploy”



Para hacer seguimiento damos clic en “Logs”



Y visualizamos el Log de la ejecución.



Aquí hemos terminado con la creación de los pipelines (CI, CD, RM), practícalo tantas veces sea necesario.