How to install?

1. Go to Jenkins Plugin Manager -> Advanced (http://[Jenkins_URL]/pluginManager/advanced).
2. Upload the plugin (.hpi file).

UiPath Pack

Can be used within a build sequence or a pipeline and does the packaging of one or more UiPath projects given their source code.

Versioning Method

The versioning of the package or packages that are created in this task can be done in two ways:

- Using the Auto-generate **the package version** option that will pseudo-randomly generate a version following the same algorithm that UiPath Studio uses when publishing to Orchestrator.
- Using custom versioning like {MAJOR}.${MINOR}.${BUILD_NUMBER}. Semantic versioning is supported.

Project(s) Path

Here you can specify either a path to a single project or a path to a directory that contains multiple other projects. You can have the following situations:

1. path\to\invoice_processing\project.json - it packages invoice_processing;
2. path\to\invoice_processing - it packages invoice_processing;
3. path\to\invoicing_projects\ where this folder contains the following:
    - path\to\invoicing_projects\PROJECT_A\
    - path\to\invoicing_projects\PROJECT_B\
    - path\to\invoicing_projects\directory_with_other_projects\

   In this case, the plugin packages individually only PROJECT_A and PROJECT_B, the result being one NuGet package for each one of them.

Output Folder

This is where the NuGet packages that were produced by the plugin are going to be stored.
One possible path could be:
`${JENKINS_HOME}\jobs\${JOB_NAME}\builds\${BUILD_NUMBER}\.`
In this case, each build folder contains the NuGet package(s) that the build yelds.

UiPath Deploy

Enables you to deploy a process automation package or multiple to a specific Orchestrator tenant. The task requires the following information: the path from where to take the packages and of course Orchestrator related information: URL, tenant name, and user credentials to perform the API calls.

An example of **Package(s) path** is:
`${JENKINS_HOME}\jobs\${JOB_NAME}\builds\${BUILD_NUMBER}\.`

The UiPath Deploy task takes the latest version of each NuGet package from the specified folder and deletes the older ones. keeping in the folder only the ones with the latest versions.

Using Environment Variables

You can use environment variables in all text boxes of both **Pack** and **Deploy** tasks. Use the following format: `${WORKSPACE}`, `${JENKINS_HOME}`, etc.

Prerequisites for the Build Agent Machine

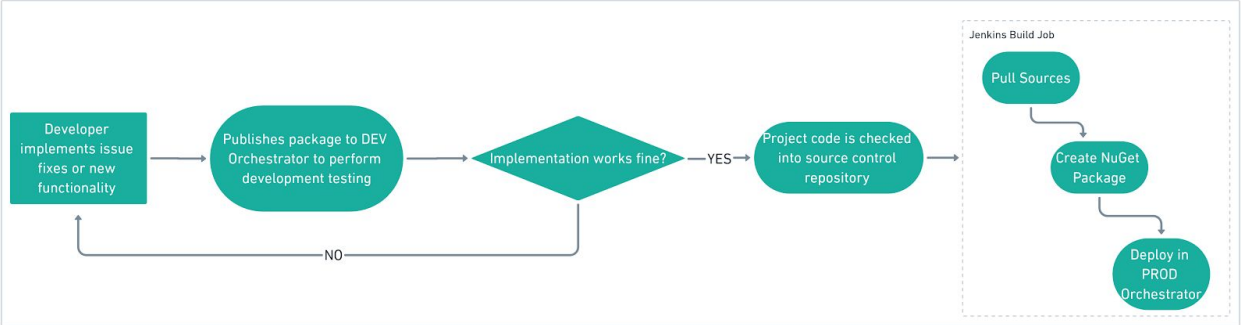
a. A reachable Orchestrator instance at which the packages are to be deployed;
b. SSL Certificate should be imported so the HTTPS calls to Orchestrator can be trusted;
c. UiPath Robot, minimum version 18.3.2, installed in C:\Program Files (x86)\UiPath\Studio.

Known Limitations

- Blue Ocean pipelines

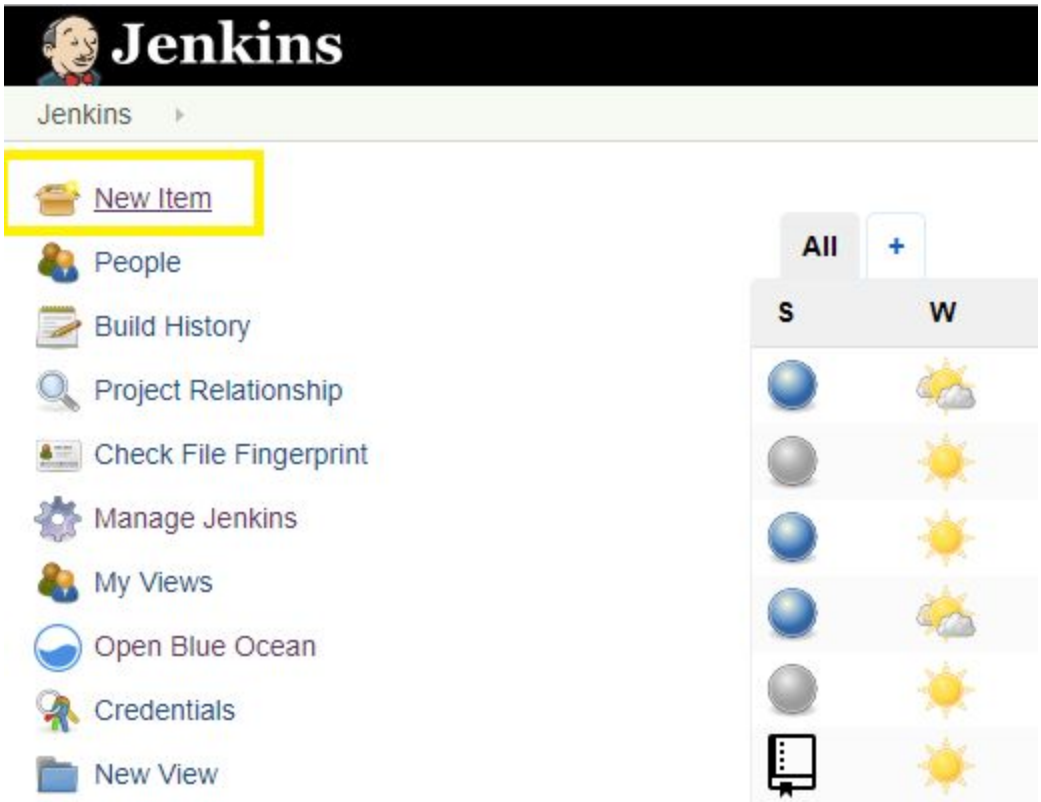
Example of usage #1: Packing and deploying a build job

This example is meant to showcase the UiPath Jenkins Plugin capabilities but also the scenario can be replicated for the following use-case:

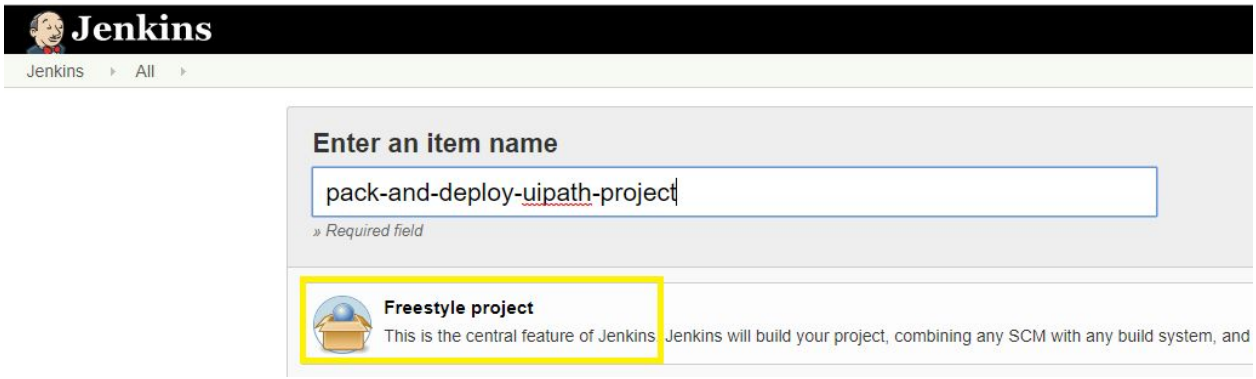


These are the steps to set up a simple pack and deploy build job:

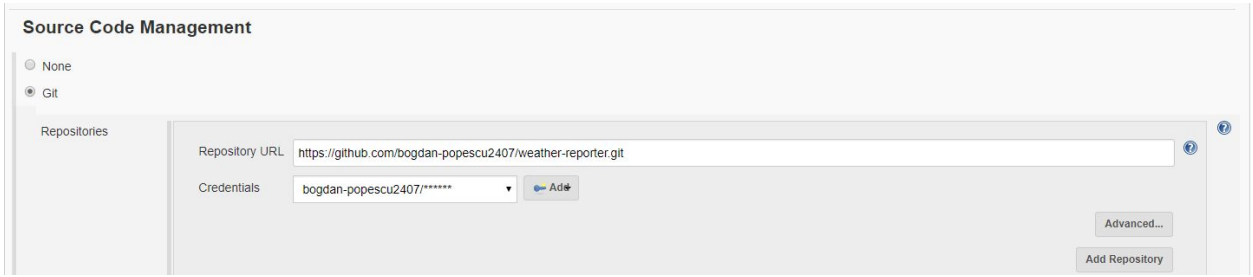
1. Create a new item.



2. Choose Freestyle Project.

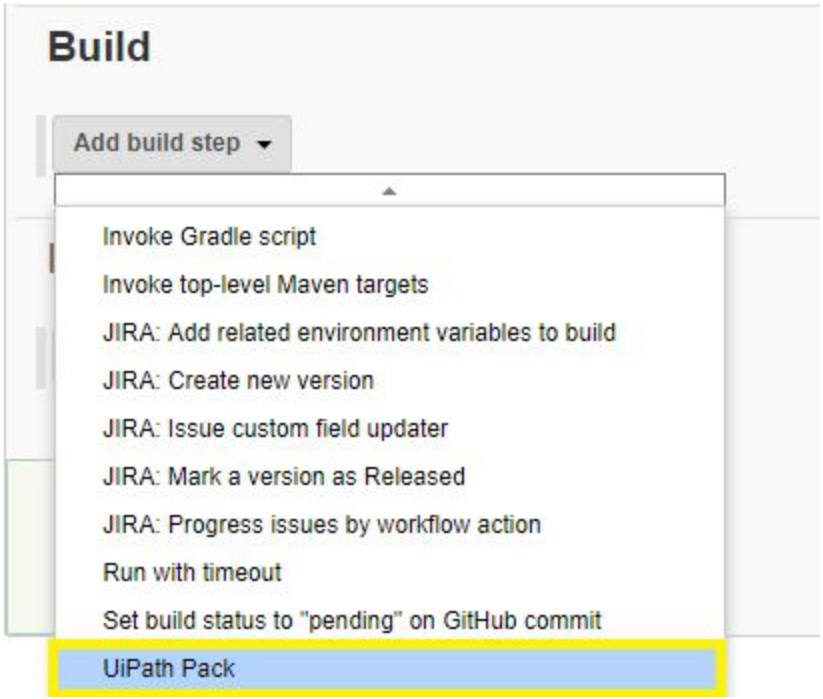


3. Specify the source code repository of the project you want to pack and deploy. This UiPath project has its own repository.

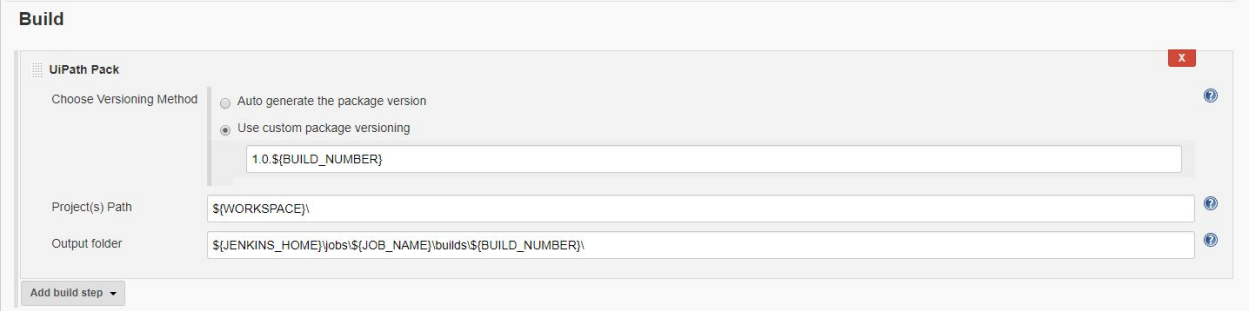


You need to create upfront your set of credentials for this operation. For configuring credentials in Jenkins please follow [this guide.]

4. Add UiPath Pack as a build step.

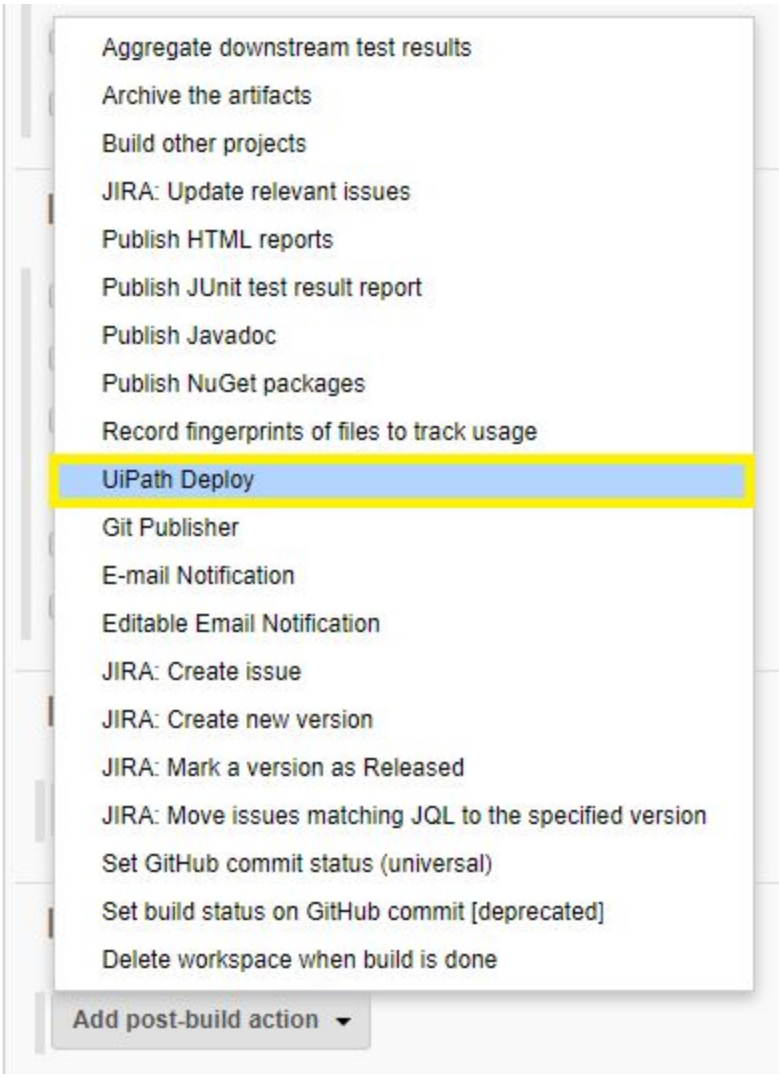


5. Configure UiPath Pack as follows:

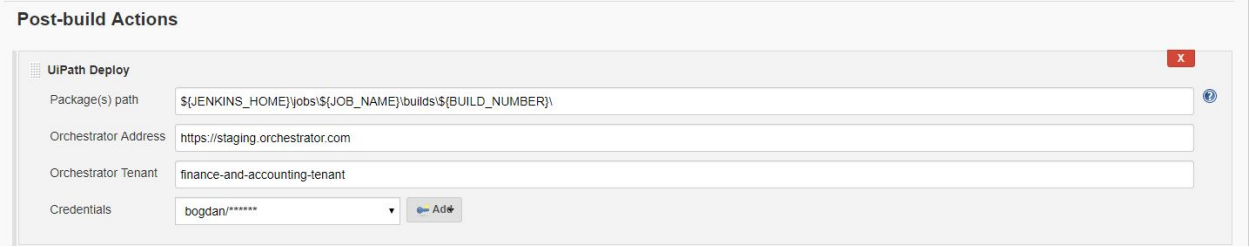


- Specify the package versioning pattern to help track packages back to their build jobs and sources. In this example, `${BUILD_NUMBER}` is a Jenkins environment variable which gets incremented with each build for this project. The first two numbers of the version, MAJOR and MINOR, can also be configured depending on the project status.
- Sources from the GIT repository are checked-out in the `${WORKSPACE}` folder, so that should be the **Project path**. Remember, in this example we're packaging and deploying only one project.
- The result of this operation is stored in the specified **Output folder**. In Jenkins, each build has its own folder so it's best to have the results placed in: `${JENKINS_HOME}\jobs\${JOB_NAME}\builds\${BUILD_NUMBER}\.`

6. Add UiPath Deploy as a post-build step.



7. Configure UiPath Deploy as shown below.

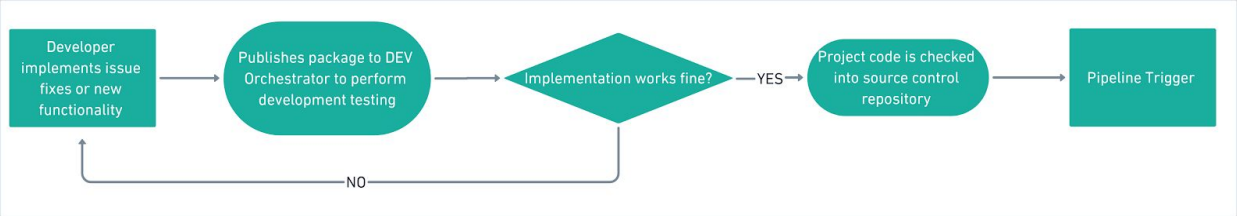


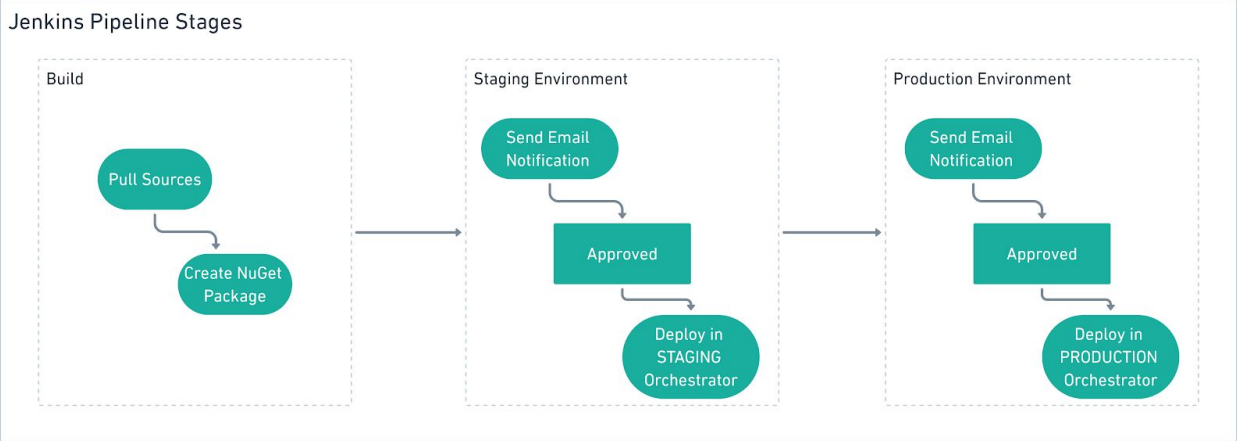
For **Package(s) path** it is recommended to use the same folder given as Output for the Pack task. Also, the credentials for Orchestrator should be defined separately in **Jenkins Credentials Manager**.

8. Run the build job.
   As a result, the latest version of the project is taken from source control, a NuGet package is created and deployed to the specified Orchestrator.

Example of usage #2: Creating a Jenkins pipeline that packages a UiPath project and deploys it across multiple stages

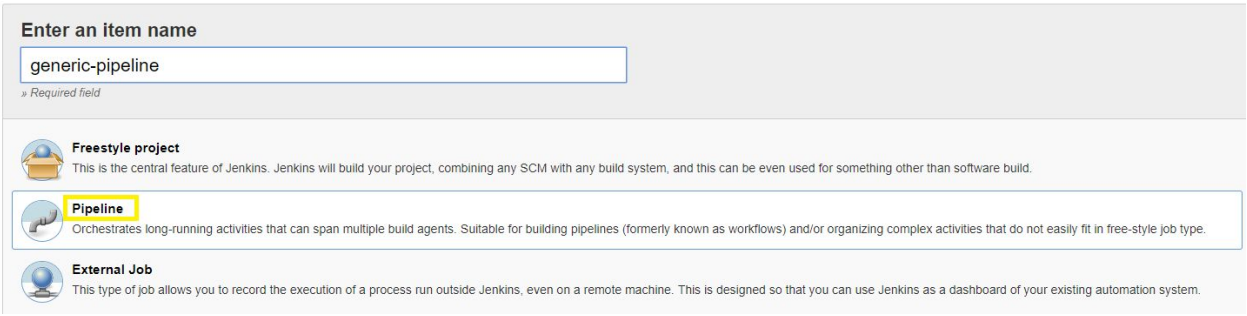
This example showcases how the UiPath Jenkins Plugin works with Jenkins Pipelines. However, please note that it can also be replicated for an RPA development context that involves multiple stages with approvals for package promotion.



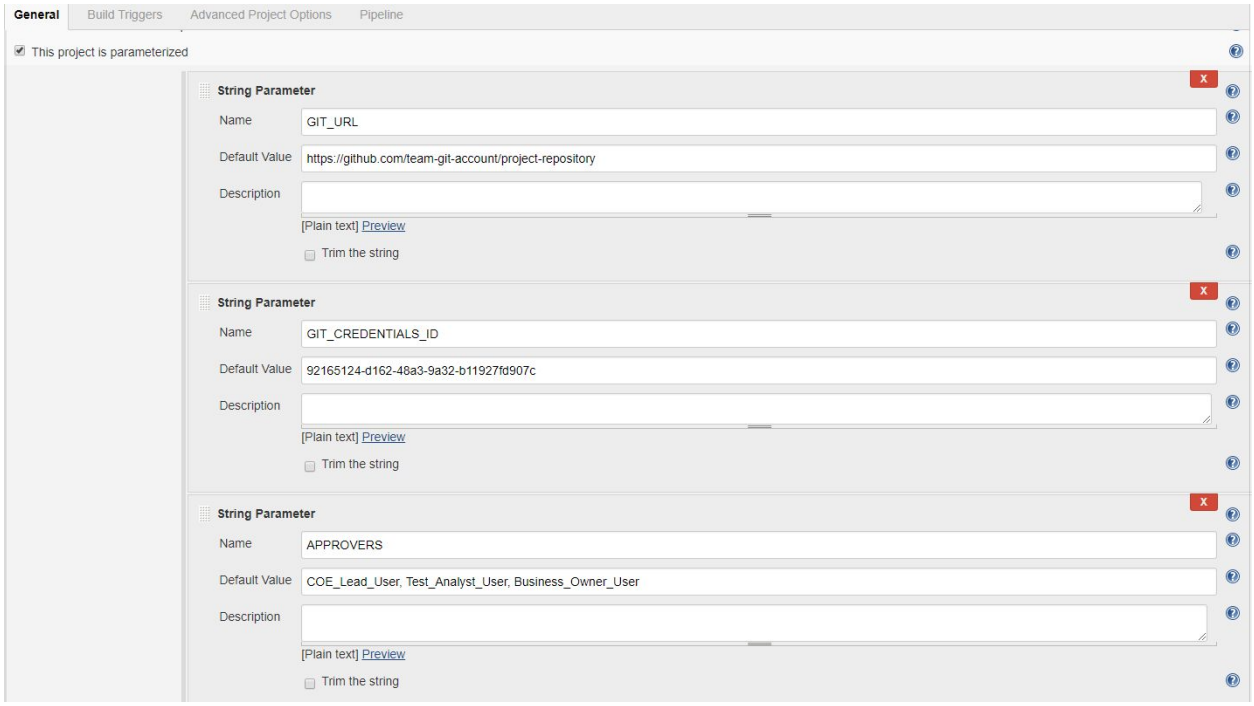


Here are the steps for creating such a pipeline:

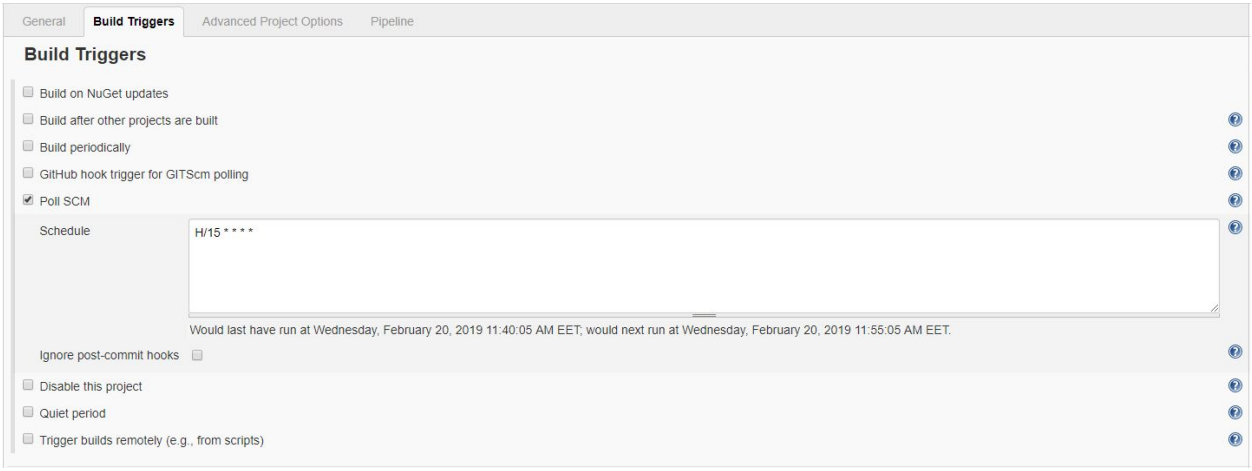
1. Create a new pipeline.



2. Parametrize the pipeline so that you can repurpose it later on with other projects, as displayed below:



3. Set-up the best-fit build triggers for your project depending on your governance model and project life-cycle:



4. Configure the stages in the pipeline and what should be executed in each of them. This is done by writing code. The code can be defined at pipeline level in Jenkins (Pipeline script) or kept in the source control repository (Pipeline script from SCM). In this example we're gonna use Pipeline script option:

And here's the code for it:

```
pipeline {
    agent any
    environment {
        MAJOR = '1'
        MINOR = '0'
    }
    stages {
        stage('Build') {
            steps {
                checkout([$class: 'GitSCM', branches: [[name:
'*/master']], doGenerateSubmoduleConfigurations: false,
extensions: [], submoduleCfg: [], userRemoteConfigs:
[[credentialsId: env.GIT_CREDENTIALS_ID, url: env.GIT_URL]]])
```

```
                UiPathPack (outputPath:
"${env.JENKINS_HOME}\\jobs\\${env.JOB_NAME}\\builds\\${env.BUIL
D_NUMBER}", projectJsonPath: "${env.WORKSPACE}", version:
[$class: 'ManualEntry', text:
"${MAJOR}.${MINOR}.${env.BUILD_NUMBER}"])
            }
        }
        stage('Staging Environment') {
            steps {
                //mail bcc: '', body: "Please check
${env.JOB_NAME} for approving deployment into Staging En
stage.", cc: '', from: '', replyTo: '', subject: 'Jenkins
Pipeline Approval Required', to: 'user.name@company.com'

                timeout(time: 14, unit: 'DAYS') {
                    input message: 'Please approve the
deployment of this package into Staging', submitter:
env.APPROVERS
                }

                build job: 'deploy-in-staging', parameters:
[string(name: 'PACKAGE_PATH', value:
"${env.JENKINS_HOME}\\jobs\\${env.JOB_NAME}\\builds\\${env.BUIL
D_NUMBER}")]
            }
        }
        stage('Production Environment') {
            steps {
                //mail bcc: '', body: 'Please check
${env.JOB_NAME} for approving deployment into Test stage.', cc:
'', from: '', replyTo: '', subject: 'Jenkins Pipeline Approval
Required', to: 'user.name@company.com'

                timeout(time: 14, unit: 'DAYS') {
                    input message: 'Please approve the
deployment of this package into Production', submitter:
env.APPROVERS
                }

                build job: 'deploy-in-production', parameters:
[string(name: 'PACKAGE_PATH', value:
"${env.JENKINS_HOME}\\jobs\\${env.JOB_NAME}\\builds\\${env.BUIL
D_NUMBER}")]
```

```
            }
        }
    }
    post {
        success {
            echo "Process ${env.GIT_URL} with version
${MAJOR}.${MINOR}.${env.BUILD_NUMBER} was successfully deployed
into Production."
        }
     }
}
```

5. Save this generic pipeline. It you want to use it for a different project, just clone the generic one and customize the following information: pipeline name, git project repository, git credentials id, and the list of approvers.

Please note that within the pipeline script, `UiPathPack` is directly invoked to create the NuGet package after checking-out the sources from SCM. However, the Orchestrator publishing part is done by invoking dedicated jobs:
`build job: 'deploy-in-staging'` or `'deploy-in-production'`

This job needs to be created and configured properly for interacting with Orchestrator instances/tenants:

The advantage for externalizing the publishing part is that the Orchestrator Address, Tenant, and Credentials are handled separately from the build pipeline, reducing the risk of unauthorized users getting this information.