**Jenkins MS Build Nexus Upload**

This document explains about the how the Jenkins automates the Dotnet build and upload the artifacts to Nexus repository.

About DotNet Project:

----------------------------

The DotNet project has below folder structure and it has pom.xml file to define the configuration details about the artifact.

📁 docker-working

└──📂 App

├──DotNet.Docker.csproj

├──Program.cs

└──📂 obj

├── DotNet.Docker.csproj.nuget.dgspec.json

├── DotNet.Docker.csproj.nuget.g.props

├── DotNet.Docker.csproj.nuget.g.targets

├── project.assets.json

└── project.nuget.cache

DotNet commands:

--------------------------------

**Maven Phases**

dotnet restore command, the .NET CLI uses NuGet to look for these dependencies and download them if necessary. It also ensures that all the dependencies required by the project are compatible with each other and that there are no conflicts between them.

dotnet build - Builds a project and all of its dependencies.

dotnet publish - Publishes the application and its dependencies to a folder for deployment to a hosting system.

dotnet pack - Packs the code into a NuGet package.

Jenkins Pipeline contains the multiple below sections.

1. Configuration Section
2. Environment Variable Declaration
3. Jenkins Initialization stage
4. Upload artifact to nexus

Configuration Section

----------------------------

Here we have defined about the Agent details where our task is going to execute. Creating a pod with docker image and it has the java, maven installed in it.

Environment Variable Declaration:

------------------------------------------------

In this section we have defined the required environment variable and it hold the below details.

1. Nexus url
2. Dotnet nexus repo name
3. Jenkins Credentials to connect to Nexus repository.

Jenkins Initialization Stage:

-----------------------------------

It will clone our java repository into the docker container to use for further stages.

Restore Dependencies Stage:

--------------------------------------

The below command used to restore the dependencies to build the dotnet application.

dotnet restore --no-cache

Dotnet Build Stage:

-------------------------

Build the dotnet application with below command.

**dotnet build --configuration Release**

Dotnet Publish Stage:

----------------------------

Publish the dotnet package files into the folder publish.

dotnet publish --configuration Release --output ./publish

Package stage:

--------------------

Here we are packing the files which has been build and published and create a file extension “.nugpkg”.

Push to nexus stage:

--------------------------

To upload the build artifact to NEXUS we can use different methods available but we are using linux “CURL” command here because our nexus is exposed as https url.

The below command is used to upload the artifact to nexus.

for file in publish/\*.nupkg; do

curl -k -v -u ${NEXUS\_CREDENTIALS\_USR}:${NEXUS\_CREDENTIALS\_PSW} -F "nuget.asset=@$file" "https://nexusrepo-tools.apps.bld.cammis.medi-cal.ca.gov/service/rest/v1/components?repository=${NEXUS\_REPOSITORY}"