Building Angular in Jenkins Pipelines

Here are some things that need to be done, for a Jenkins build server to build/pack/publish Angular libraries, and apps.

> NOTE: This page assumes the library or app source code is being built in a monorepo workspace.

When using the same Jenkins build server to build Angular apps that target differing versions of Angular, it is necessary to swap in the required Node.js version, compatible with that version of Angular.

> See this article for setting up NVM: Jenkins: Add Angular Build Support

With NVM installed on the build server, we need to make sure that the build job configuration and pipeline steps are setup.

Build Job Config

These differ slightly, depending if we are building an app or a library.

Libraries get published to an NPM repository, whereas apps get published to Artifactory.

Library Job Config

When compiling an Angular library, the job's Properties need to include these:

```
ENV_SolutionName=oga.webui.runtimesecurity
ENV_WorkspaceName=workspace
ENV_projectName=lib-oga-webui-runtimesecurity
ENV_gitrepo=git@github.com:ogauto/OGA.WebUI.RuntimeSecurity.Lib.git
ENV_angularVersion=15
```

Here's a description of each property:

- ENV_Solution - folder name inside the same folder where the jenkinsfile sits.
- ENV_WorkspaceName - folder name insides the solution folder, that is the monorepo workspace root.
- ENV_projectName - name of the library, as listed in npm.
  This is also the folder name, inside the workspace's projects folder.
- ENV_gitrepo - full path to the git repository.
- ENV_angularVersion - version of Angular being targeted.
  Will be: 14, 15, 17.0.7, 17.3.0.

App Job Config

When compiling an Angular application, the job's Properties need to include these:

```
ENV_artifactoryUploadRepo=oga-built-dev
ENV_artifactoryRepoFolder=OGA.Backups.WebUI
ENV_SolutionName=OGA.Backups.WebUI
ENV_WorkspaceName=backupsui-workspace
ENV_projectName=mgmtui
ENV_globalProjectName=oga.backups.webui
ENV_artifactZipFileName=OGA.Backups.WebUI
ENV_gitrepo=git@github.com:LeeWhite187/OGA.Backups.WebUI.git
ENV_angularVersion=17.0.7
```

Here's a description of each property:

- ENV_artifactoryUploadRepo - repository name on the Artifactory server.
- ENV_artifactoryRepoFolder - folder name, inside the Artifactory repository.
- ENV_Solution - folder name inside the same folder where the jenkinsfile sits.
- ENV_WorkspaceName - folder name insides the solution folder, that is the monorepo workspace root.
- ENV_projectName - folder name, under the projects folder, where the app's source is located.
- ENV_globalProjectName - application full name, used by the post-action notification step, when publishing job status.
- ENV_artifactZipFileName - Name of the published zip file, containing the built application.
- ENV_gitrepo - full path to the git repository.
- ENV_angularVersion - version of Angular being targeted.
  Will be: 14, 15, 17.

Pipeline Steps

Here's a description of necessary details in the pipeline job for an Angular app or library.

Common Pipeline Steps

Here are common elements for both Angular app and library pipelines.

Environment Block

Both pipeline types lead off with an environment step, to populate the jenkins user session with necessary variables.

```
pipeline
{
    agent any
    environment
□{
□□// Define where NVM is located (in the user's home), so we can call its scripts...
        NVM_DIR = "${HOME}/.nvm"
    }


    ...
}
```

Place the environment block before all scripts blocks and after the agent block, like above.

Angular Version Step

This step figures out what specific version of Node.js and Angular will be needed.

Place this step above others that run NVM.

```
□□// Determine the actual version of Angular and Node.js that we need to compile with...
        stage('Angular Version')
□□{
            steps
□□□{
                script
□□□□{
□□□□□echo "angularVersion: ${angularVersion}"
                    if (angularVersion == '14')
□□□□□{
                        env.NODE_VERSION = "16"
                        env.NG_CLI_VERSION = "14.2.0"
                    }
```

```
    else if (angularVersion == '15.2.10')
    {
                env.NODE_VERSION = "18"
                env.NG_CLI_VERSION = "15.2.10"
            }
    else if (angularVersion == '17.0.7')
    {
                env.NODE_VERSION = "20"
                env.NG_CLI_VERSION = "17.0.7"
            }
    else if (angularVersion == '17.3.0')
    {
                env.NODE_VERSION = "20"
                env.NG_CLI_VERSION = "17.3.0"
            }
    else if (angularVersion == '17.3.12')
    {
                env.NODE_VERSION = "20"
                env.NG_CLI_VERSION = "17.3.12"
            }
    else if (angularVersion == '19.2.3')
    {
                env.NODE_VERSION = "22"
                env.NG_CLI_VERSION = "19.2.3"
            }
    else
    {
                error "Unsupported Angular version: ${angularVersion}"
            }
    echo "env.NG_CLI_VERSION is: $env.NG_CLI_VERSION"
    echo "env.NODE_VERSION is: $env.NODE_VERSION"
          }
        }
      }
```

NVM Swap-In Step

The next step will install the Node.js and Angular CLI version for the project.

It looks like this:

```
        stage('Install Node.js & Angular CLI')
    {
            steps
    {
    //    // Adapted from here: https://www.baeldung.com/ops/jenkins-pipeline-change-to-another-folder
    //    // NOTE: This 'dir' command will switch the directory only for the steps inside it. After the block, the working directory reverts to the jenkinsfile folder.
    //    dir('Libraries/oga.webui.sharedkernel/workspace')
    //    {
    //        /* execute commands in the workspace directory */
        // NOTE: We wrapped the NVM and NPX calls in a single shell call that first loads the .bashrc profile, because the shell is noninteractive.
        // NOTE: The '.' before the $HOME/.bashrc call is the Ubuntu equivalent of the command 'source'.
        sh "whoami"
        sh '''
            . $NVM_DIR/nvm.sh
            nvm install $NODE_VERSION
            nvm use $NODE_VERSION
            npx -p @angular/cli@$NG_CLI_VERSION ng version
        '''
    //    }
                // sh '''
                //     source $NVM_DIR/nvm.sh
                //     nvm install $NODE_VERSION
                //     nvm use $NODE_VERSION
                //     npx -p @angular/cli@$NG_CLI_VERSION ng version
                // '''
            }
        }
```

It employs a 'source' command (a '.' in Ubuntu Linux) to load the jenkins user environment with NVM variables.

It will swap in the Node.js version, and install Angular.

NPM Install Step

The next step will install packages, via npm.

It uses 'npm ci', instead of 'npm install'.
This is because 'ci' will stick to package versions that have been defined in your package.json lock file, instead of installing the latest version.

So, using 'npm ci' ensures a consistent build, that matches dependency versions to what you last developed in.

```
        stage('Install Packages')
    {
    steps
    {
    // Adapted from here: https://www.baeldung.com/ops/jenkins-pipeline-change-to-another-folder
    // NOTE: This 'dir' command will switch the directory only for the steps inside it. After the block, the working directory reverts to the jenkinsfile folder.
    dir("${solutionName}/${workspaceName}")
    {
    /* execute commands in the workspace directory */
    // See this for these commented out lines: https://wiki.galaxydump.com/books/howto/page/jenkins-fails-to-inst packages-for-angular-builds
    //    sh "ng update @angular/cli@15 --allow-dirty"
    //    sh "ng update @angular/core@15 --allow-dirty"
       //sh "npm install"
    sh '''
    . $NVM_DIR/nvm.sh
    nvm use $NODE_VERSION
    npm ci
    '''
    }
    }
    }
```

The above common pipeline steps are necessary to swap in libraries and load packages required for building.

App Build and Publish Steps

Here's the steps to build and publish an application:

```
        stage('Build App')
    {
    steps
    {
    // Adapted from here: https://www.baeldung.com/ops/jenkins-pipeline-change-to-another-folder
    // NOTE: This 'dir' command will switch the directory only for the steps inside it. After the block, the working directory reverts to the jenkinsfile folder.
```

```
        dir("${solutionName}/${workspaceName}")
        {
          /* execute commands in the workspace directory */
          //sh "npm run build-library"
          //sh "npx ng build ${projectName} --configuration=production"
          sh '''
            . $NVM_DIR/nvm.sh
            nvm use $NODE_VERSION
            npx ng build ''' + "${projectName}" + ''' --configuration=production
          '''
        }
      }
    }
        stage('Create Publish Folder')
    {
      steps
      {
        fileOperations([folderCreateOperation("${solutionName}/publish")])
      }
    }
        stage('Zip-App')
    {
      steps
      {
        script
        {
          zip zipFile: "${solutionName}/publish/${artifactZipFileName}-angular.zip",
          archive: false,
          dir: "${solutionName}/${workspaceName}/dist/${projectName}/browser",
          overwrite: true
        }
      }
    }
```

After the above zip step has executed, just use a normal Artifactory publish step to push the zip file to your bin repository for later deployment.

Library Build, Pack, and Publish Steps

The steps for a library are a little different.
It includes a pack step, and publishes to a different repository type (npm repo):

```
        stage('Build Library')
    {
    steps
    {
    // Adapted from here: https://www.baeldung.com/ops/jenkins-pipeline-change-to-another-folder
    // NOTE: This 'dir' command will switch the directory only for the steps inside it. After the block, the working directory reverts to the jenkinsfile folder.
    dir("${solutionName}/${workspaceName}")
    {
    /* execute commands in the workspace directory */
      //sh "npm run build-library"
      //sh "npx ng build ${projectName} --configuration=production"
    sh '''
    . $NVM_DIR/nvm.sh
    nvm use $NODE_VERSION
    npx ng build ''' + "${projectName}"
    }
    }
    }
        stage('Pack Library')
    {
    steps
    {
    /* Execute the pack command from the library's dist folder */
    // Adapted from here: https://www.baeldung.com/ops/jenkins-pipeline-change-to-another-folder
    // NOTE: This 'dir' command will switch the directory only for the steps inside it. After the block, the working directory reverts to the jenkinsfile folder.
    dir("${solutionName}/${workspaceName}/dist/${projectName}")
    {
    /* execute commands in the workspace directory */
    sh '''
    . $NVM_DIR/nvm.sh
    nvm use $NODE_VERSION
    npm pack
    '''
    }
    }
    }
        stage('Publish Library')
    {
```

```
    steps
    {
        /* Execute the publish command from the library's dist folder */

        // Adapted from here: https://www.baeldung.com/ops/jenkins-pipeline-change-to-another-folder
        // NOTE: This 'dir' command will switch the directory only for the steps inside it. After the block, the working directory reverts to the jenkinsfile folder.
        dir("${solutionName}/${workspaceName}/dist/${projectName}")
        {
            sh '''
                . $NVM_DIR/nvm.sh
                nvm use $NODE_VERSION
                npm publish --registry https://npmrepo.ogsofttech.com:4873/
            '''
        }
    }
}
```

Revision #5
Created 1 March 2025 19:00:31 by glwhite
Updated 6 August 2025 10:13:00 by glwhite