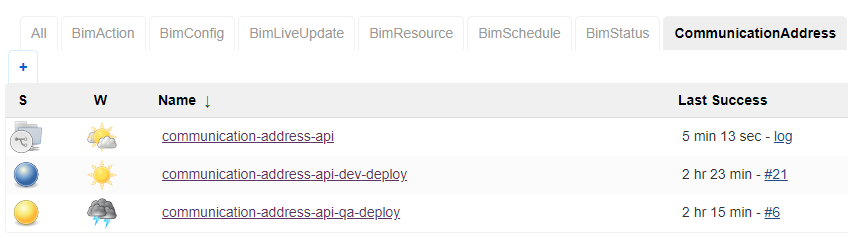
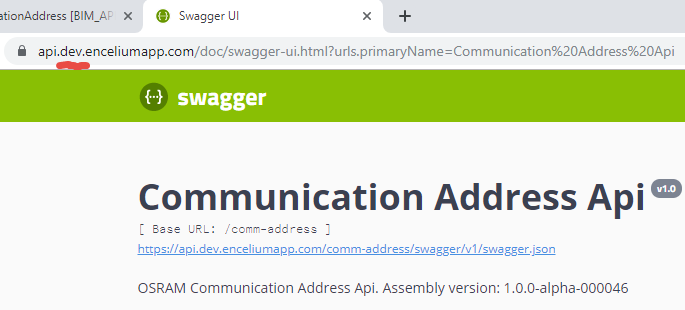
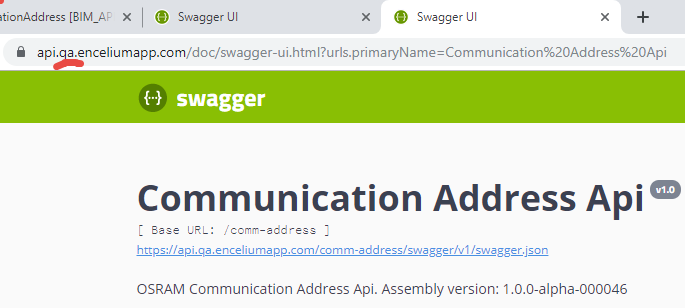
Run scripts against multiple environments with different credentials in CI/CD pipeline

# Requirements:

When QA scripts, such as postman scripts, are executed in Jenkins deployment projects, the scripts should be running against the environment URL with the environment credentials.



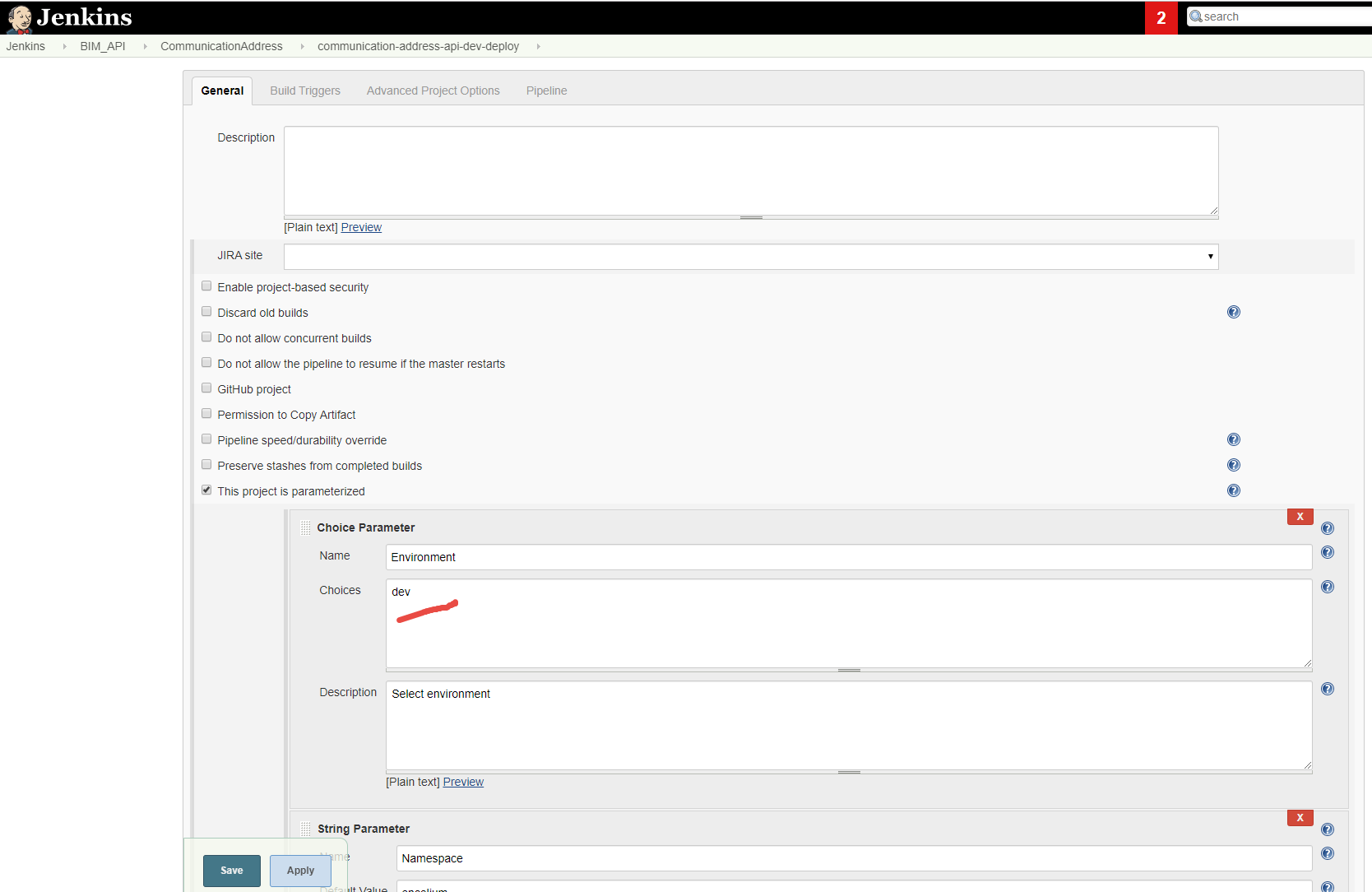
We currently only have 2 environments, dev and qa. In the future, we will also have staging and production environments. Each of the environment has its own deployment project as displayed above. “bim\_communication\_address\_api” is one of the API/microservice we have, inside the “bim\_communication\_address\_api” folder, you can see “Jenkinsfile” and “Deploy.Jenkinsfile”. The “Jenkinsfile” is in charge of the “communication-address-api” multi-branch project as displayed above, which create the Docker image after developers’ code is merged to dev branch (we use dev instead master branch). The “Deploy.Jenkinsfile” is in charge of all the deployment projects in Jenkins for communication address api, i.e., the “communication-address-api-dev-deploy” and “communication-address-api-qa-deploy” projects. Both “Jenkinsfile” and “Deploy.Jenkinsfile” are based on the functions that being defined in the “jenkins\_shared\_libraries” folder (this is an improvement since last time I created the “Version\_Control” folder in this repo). During the process of each of the Jenkins deployment project, after the docker file has been deployed by Kubernetes, the corresponding URL will be populated as displayed below, and our QA sanity and regression script will be running against these URLs.

As a result, in order to run QA scripts againest different URLs inside CICD pipeline, we need to generate the URL, URL credentials (currently is basic authentication of username and password), these 3 parameters from Jenkins GUI or script, and pass them into our QA script which is being managed by npm (see “package.json” file inside bim\_communication\_address\_api\qa\postman folder)

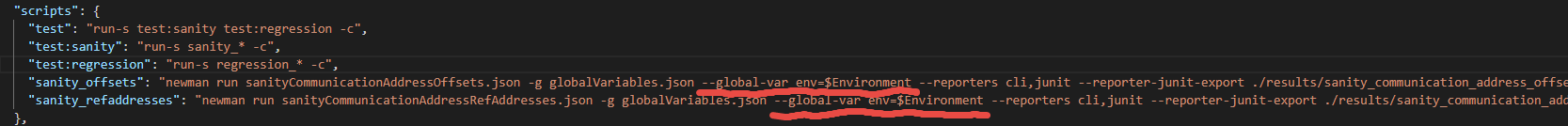
# Step 1: Generate the URL, URL credentials in Jenkins

In Jenkins GUI, if you click the “communication-address-api-dev-deploy” project, then click the “Configure” on the left hand menu, you will notice that this deployment project already has a paramter called “Environment”, and its value is “dev”. In the the “communication-address-api-qa-deploy” project, it also has this parameter which has the same name “Environment”, but its value is “qa”.



This Environment variable from Jenkins can be easily passed into newman command as --global-var env=$Environment

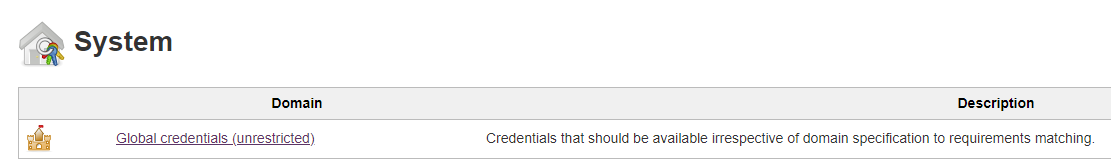
to overwrite our postman global variables “env” which was defined inside “globalVariables.json”. The “globalVariables.json” located under “bim\_communication\_address\_api\qa\postman”. Our API URLs are [https://api.dev.enceliumapp.com](https://api.dev.enceliumapp.com/) for dev environment and [https://api.qa.enceliumapp.com](https://api.qa.enceliumapp.com/) for qa enviroment, and we use “env” variable to construct the URL during test. The “env” variable should have value “dev” when testing dev URL, and should have value “qa” when testing qa URL.



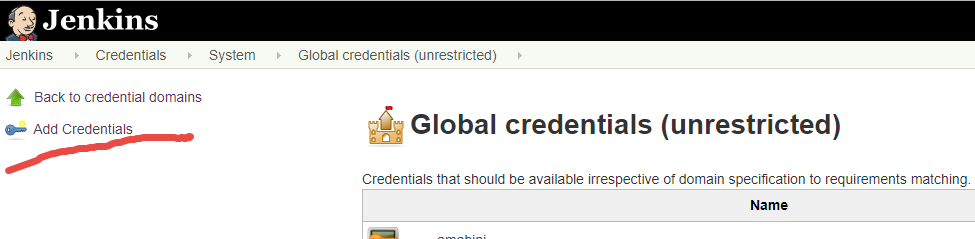
However, to execute npm run located will have a problem after above change. I tried to set system variable on my local PC to have Environment variable, but newman command cannot recognized it. Let us put this problem aside, we still need to overwrite the API username and password since under different environment, these credentials are different.



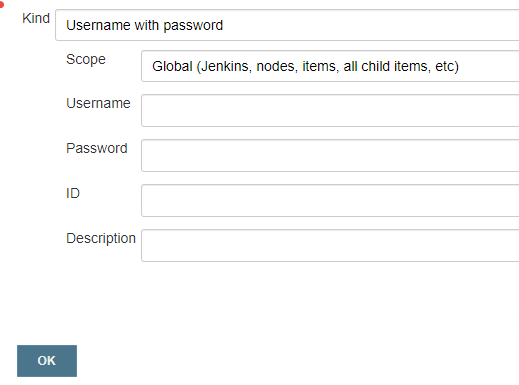
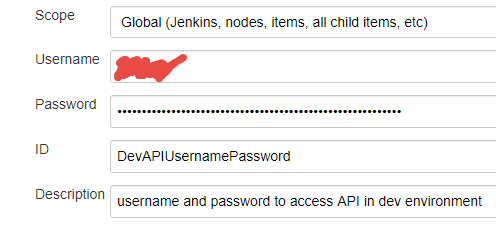
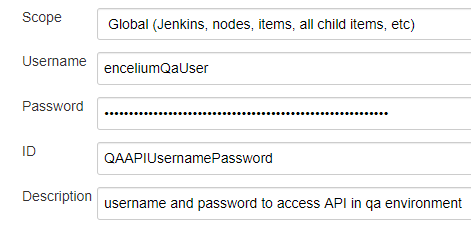
From Jenkins GUI, click the left-hand side “Credentials” selection from the menu, you will notice that there are many different types of credentials have been created by our DevOps person. I created 2 new set of credentials here, one set for dev environment username and password (named as DevAPIUsernamePassword), and one set for qa environment username and password (named as QAAPIUsernamePassword), as displayed above. The way to create credentials in Jenkins is, click on the “Jenkins” link (the 3rd column in the table) of any of the existing credentials in Jenkins – Credentials page.



Continue click on the “Global credentials (unrestricted)” link, after the “Global credentials (unrestricted)” page is open, click on the left-hand side “Add Credentials” link to add new credentials.



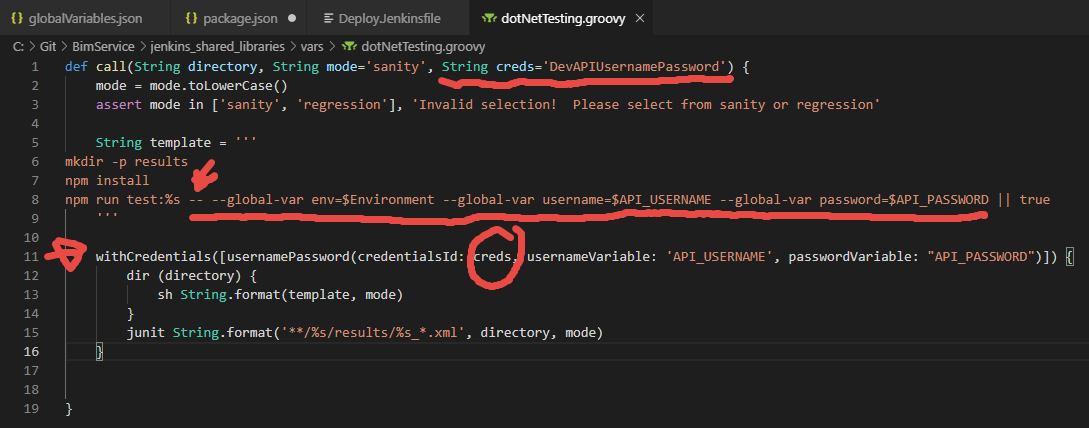
You can choose different kind of credentials, for my case, I picked “Username with password”. Fill in the Username and Password fields. The ID filed will be the name you pass in to your script. The end result for dev and qa credentials are in the middle and right below. In case you made any mistake, you can always pick the credential, and choose “Update” from the left-hand side menu to update the content.

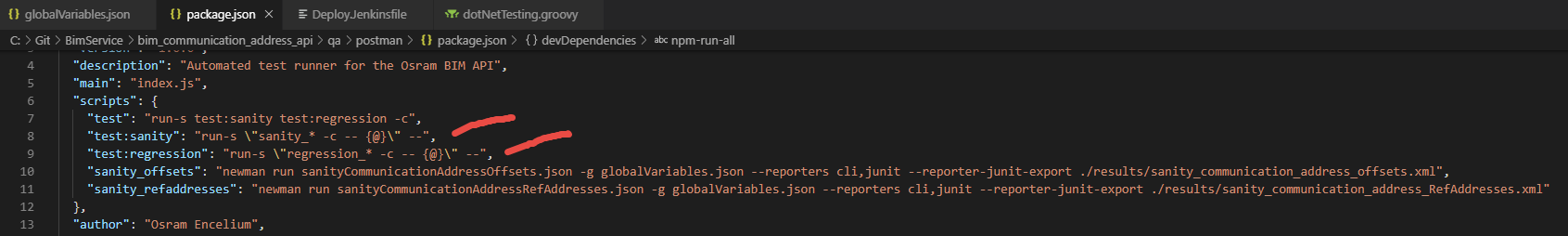
# Step 2: pass in Jenkins parameters to package.json

Let me just show you the end result.

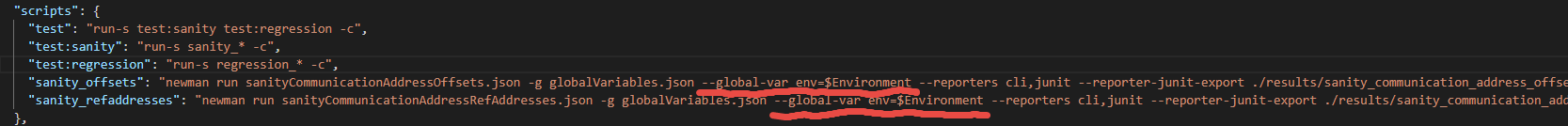
First, from “jenkins\_shared\_libraries\vars” directory, find the “dotNetTesting.groovy” file. Line 11 wrapped original code block from line 12 to line 15. The use of line 11 is based on <https://jenkins.io/doc/pipeline/steps/credentials-binding/>, search for “usernamePassword”. Basically, it take the credentialId which I just defined, either “DevAPIUsernamePassword” which is passed in as a default value in line 1, or “QAAPIUsernamePassword” which is passed in “Deploy.Jenkinsfile” line 67 or 79. The withCredentials function get the credentials from Jenkins by id, and created 2 parameters API\_USERNAME and API\_PASSWORD.



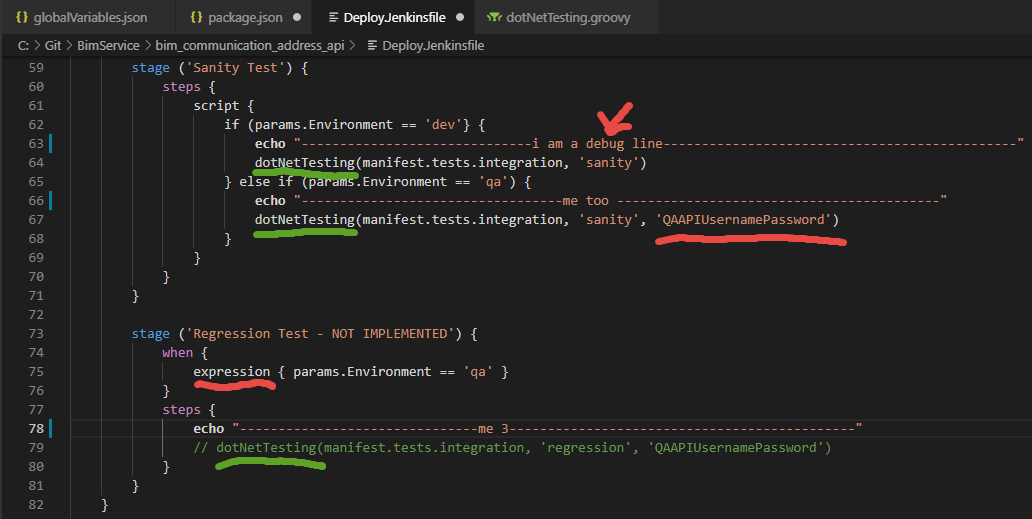
These 2 parameters are the same as $Environment parameter defined in the Jenkins deployment projects, so they are used in the same way in line 8 (note you need to write -- first before pass in --gloable-var). line 8 was originally as npm run test:%s || true



The benefit of line 8 is to solve the problem that I mentioned in step 1, if we pass in --global-var env=$Environment --global-var username=$API\_USERNAME --global-var password=$API\_PASSWORD inside the “package.json” line 10 and 11, then we cannot longer run the script locally. Line 8 of “dotNetTesting.groovy” file is calling line 7 of the “package.json” file, which aggregate the test of line 8 and line 9. Line 8 aggregated all the test which name starts with sanity\_, and line 9 aggregate all the tests which name start with regression\_. We changed line 8 and line 9 of “package.json”, so it can understand the arguments passed in from line 8 of “dotNetTesting.groovy”. You can compare the change of line 8 and line 9 of “package.json” by comparing the image above, and the image below. The change of line 8 and line 9 of “package.json” turned out to be the most difficult part of this task.



Let us come back to talk about where “dotNetTesting.groovy” file is being used. This file is being called by “Deploy.Jenkinsfile” under “bim\_communication\_address\_api” folder. dotNetTesting function is used under “Sanity Test” and “Regression Test” stages in the “Deploy.Jenkinsfile”.



Couple things to note:

1, you can not have multiple when statements in a stage, so in the Sanity Test stage, I used if else statement. However, please note, when you use when statement, you use expression, but when you use if statement, if you still uses expression, then it will always be true, and you can never falls into the else if branch.

2, the use of echo line is actually a debug line, it will show us in Jenkins result. In Jenkins, click 8s under “Sanity Test” column, a pop up with “Log” tag will show up, click on it, another pop up named “Stage Logs” are displayed on the top, you will see the echo line being printed (at the time the echo line was -----------------------000). Then you can know which if branch you actually falls into.

